Rowe, Francisco (Ed.); Maier, Gunther (Ed.); Arribas-Bel, Daniel (Ed.); Rey, Sergio J. (Ed.)

Periodical Part

Special Issue: The potential of notebooks for scientific publication: Reproducibility, and dissemination

REGION

Provided in Cooperation with:
European Regional Science Association (ERSA)

Suggested Citation: Rowe, Francisco (Ed.); Maier, Gunther (Ed.); Arribas-Bel, Daniel (Ed.); Rey, Sergio J. (Ed.) (2020) : Special Issue: The potential of notebooks for scientific publication: Reproducibility, and dissemination, REGION, ISSN 2409-5370, European Regional Science Association (ERSA), Louvain-la-Neuve, Vol. 7, Iss. 3, https://openjournals.wu.ac.at/ojs/index.php/region/issue/view/24

Volume 7, Number 3, 2020

REGION

The Journal of ERSA
Powered by WU

The Potential of Notebooks for Scientific Publication: Reproducibility and Dissemination

Special Issue edited by **Francisco Rowe**, **Gunther Maier**, **Daniel Arribas-Bel** and **Sergio J. Rey**

Table of Contents

Funded by

This special issue on "The Potential of Notebooks for Scientific Publication, Reproducibility and Dissemination" is edited by Francisco Rowe (University of Liverpool, Liverpool, UK), Gunther Maier (Modul University, Vienna, Austria), Daniel Arribas-Bel (University of Liverpool, Liverpool, UK), Sergio J. Rey (University of California, Riverside CA, USA). With the exception of the editorial, all contributions to this special issue have already been published in earlier issues of REGION, for the sake of immediate exposure of the content.

- *Urban Street Network Analysis in a Computational Notebook*by Geoff Boeing was originally published in vol. 6, no. 3, 39–51.
- *Random Parameters and Spatial Heterogeneity using Rchoice in R* by Mauricio Sarrias was originally published in vol. 7, no. 1, 1–19.
- *REAT: A Regional Economic Analysis Toolbox for R* by Thomas Wieland was originally published in vol. 6, no. 3, R1–R57.
- *Demonstrating the utility of machine learning innovations in address matching to spatial socio-economic applications* by Sam Comber was originally published in vol. 6, no. 3, 17–37.
- *A reproducible notebook to acquire, process and analyse satellite imagery: Exploring long-term urban changes* by Meixu Chen, Dominik Fahrner, Daniel Arribas-Bel and Francisco Rowe was originally published in vol. 7, no. 2, R15–R46.
- *Exploring long-term youth unemployment in Europe using sequence analysis: A reproducible notebook approach* by Nikos Patias was originally published in vol. 6, no. 3, 53–69.
- *Teaching on Jupyter – Using notebooks to accelerate learning and curriculum development* by Jonathan Reades was originally published in vol. 7, no. 1, 21–34.

Editorials

REGION
The Journal of ERSA
Powered by WU

Volume 7, Number 3, 2020, E1–E5
DOI: 10.18335/region.v7i3.357

journal homepage: region.ersa.org

The Potential of Notebooks for Scientific Publication: Reproducibility, and Dissemination

Francisco Rowe[1], Gunther Maier[2], Daniel Arribas-Bel[1] and Sergio J. Rey[3]

[1] University of Liverpool, Liverpool, UK
[2] Modul University, Vienna, Austria
[3] University of California, Riverside CA, USA

Received: 25 December 2020/Accepted: 25 December 2020

1 Background

Recent developments and discussions concerning the SARS-Cov-2 virus and the development of a vaccine illustrate once again the necessity to assume "that scientific claims are supported by solid evidence" (Branco et al. 2017). In recent years, however, we see increasing evidence that casts doubts on this assumption (e.g. Fanelli 2009, Ioannidis 2011, Prinz et al. 2011, Begley, Ellis 2012, Fokkens et al. 2013, Open Science Collaboration 2015). "As a result, there is an increasingly urgent call for validation and verification of published research results, both within the academic community and the public at large (e.g. Naik 2011, Zimmer 2012, Begley 2012, Editorial 2013a,b, Branco 2012)" (Branco et al. 2017, p. 1). This is particularly important at a time when the scale and complexity of scientific studies grow, and replicability and reproducibility of scientific research has gained salience (Peng 2011).

Recent developments in software and web browser technology may help in solving this problem. They have enabled exciting and fast-moving developments in many areas of research, among them tools that can help validate and verify research as well as stimulate knowledge transfer among researchers. Computational notebooks, particularly Jupyter Notebooks, represent a major advance for scientific research. A Jupyter Notebook is an open-source web application which enables creating and sharing documents containing live code, equations, visualisations and narrative text (Jupyter Project 2019). A Jupyter notebook comprises a series of 'cells' containing executable code, or markdown, along with the popular HTML markup language for prose descriptions and LaTeX for mathematical equation write up. Jupyter Notebooks have enabled a new type of programming which emphasises a prose-first approach where exposition with human-friendly narrative is threaded with code blocks. The Jupyter Notebook was originally developed by Fernando Perez and Brian Granger in the Python programme language in 2011 and known as IPython Notebooks. In 2013, the technology was expanded to allow for additional programming languages and renamed 'Jupyter'[1].

The interactive and narrative nature of computational notebooks provide unique opportunities for sharing computational results, enabling reproducibility and publishing scientific research. Traditionally, code, data, results, and their exposition are stored in separate files, which is a source of disconnect and easy misalignment. Computational

[1] 'Jupyter' is an acronym for Julia, Python and R, three of the main modern languages for scientific computing.

notebooks allow conducting analyses and integrating code, results and descriptive text into a single 'computational narrative' to be shared, read and executed by others (Pérez, Granger 2015, Kluyver et al. 2016). Attracted by the ability to combine executable code and descriptive text in a single document, an increasingly large community of researchers have adopted computational notebooks to document, publish and share their research via personal websites and GitHub (Parente 2019).

Yet publishers have not embraced this technology. We believe that there are great benefits for the scientific community and general public from publishing computational notebooks. The publication of computational notebooks, along with articles, enables reproducibility and replicability of data analysis and methods. Computational notebooks offer a valuable vehicle for teaching and demonstration of analytical tools. They can also augment the impact of research beyond its primary objectives by extending original analysis and by reaching non-academic communities (Arribas-Bel et al. 2020). The interactivity of notebooks can engage policy makers and the general public in ways that standard academic journal publications cannot. Notebooks can be used to engage policy, discipline-specific or local knowledge experts in the research process. In doing so, data and outcomes channeled through notebooks can enable the identification of new relevant patterns or uses that may have not been reported or explicitly discussed in the original publication.

In view of these potentials, REGION officially announced a new form of publication, computational notebooks, in 2019. In order to demonstrate the value of computational notebooks in regional research and to stimulate this means of publication, we organized this special issue. REGION will continue to accept submissions in computational notebooks (.ipynb and .Rmd files). When accepted, computational notebooks are published in three formats: R or Python notebook file extensions, HTML and pdf. Unlike the pdf format, R or Python notebook file and HTML file extensions will provide an interactive version of the code which can be fully reproduced. REGION encourages authors to make submissions in these formats. Two publication options are available publishing computation notebooks: (1) as a companion to a research article, or (2) as standalone piece in the Resource section. By publishing computational notebooks, REGION seeks to encourage appropriate recognition of the work dedicated to this form of document. Normally the use of computer code published on personal websites or GitHub does not receive appropriate recognition by the way of citation given unfamiliarity with this form of publication or lack of a referenceable identifier. In REGION, computational notebooks are published as regular papers, receive a digital object identifier (DOI), and hence will be referenceable and citable.

2 The Issue

This Special Issue aims to introduce the publication of computational notebooks in REGION. Seven articles make up this Special Issue and illustrate some of the the key benefits of computational notebooks.

The first three articles use notebooks to introduce advanced urban planning and statistical methods available through open software. Boeing (2019) illustrates the potential of computational notebooks in urban analytics and planning introducing 'OSMnx'. 'OSMnx' is a Python package for working with OpenStreetMap data and modelling, analysing and visualising street networks anywhere in the world. The notebook shows how to download and model street networks, compute network indicators, visualise street centrality, calculate routes, and work with other spatial data, including building footprints and points of interest.

Sarrias (2020) uses a computational notebook to introduce a R package called 'Rchoice'. Rchoice offers a statistical modelling framework to estimate spatial heterogeneity by estimating locally varying coefficients offering different latent structures in a discrete choice setting. Wieland (2019) introduces the R package 'REAT', a Regional Economic Analysis Toolbox for R, and extensively illustrates its capabilities with regional economic data for Germany. Together the computational notebooks by Boeing, Sarrias, and Wieland illustrate how methods can be introduced to new users and help researchers reach broader

audiences interested in learning from, adapting, and remixing their work.

The following two articles use computational notebooks to introduce the application of novel methods and data. Comber (2019) illustrates the use of machine learning to conduct address matching. Data often lack of unique identifiers to enable one-to-one address matching. Deterministic matching based hand-crafted rules that classify address matches and non-matches based on specialist domain knowledge are typically applied. Machine learning approaches can provide a faster and automatable way to match addresses with little human intervention. The notebook offers an end-to-end pipeline to conduct address match using machine learning.

Chen et al. (2020) contributes a computational notebook to acquire, process and analyse satellite imagery. While satellite imagery is often used to study and monitor changes in natural environments and the Earth surface, it has remained underutilised in Regional Science to study cities despite the open availability and extensive temporal coverage of data sets, like Landsat enabling monitoring long-term changes for a period of up to 46 years. The notebook offers a tool to demonstrate how to batch-download high-resolution satellite imagery; and enable the extraction, analysis and visualisation of features of the built environment to capture long-term urban changes.

Patias (2019) contributes a notebook illustrating its interactivity potential as an engaging resource for end users and researchers through a regional analysis of youth unemployment in Europe. Interactive maps and figures enable readers to explore the data in greater detail by dragging, brushing and zooming. The notebook also offers an end-to-end workflow from reading raw data via API, through the data processing, to the final publication outputs. The notebook demonstrates the existence of systematic pattern of youth unemployment across Europe. Four distinctive groups of regions are identified: 'stable low youth unemployment'; 'stable moderate youth unemployment', 'increasingly high youth unemployment', and 'stable high youth unemployment'.

The final article by Reades (2020) illustrates the use of computational notebooks to support teaching delivery and enhance student learning. Specifically, the notebook argues that given the proliferation of large and complex spatial data, there is a need not only for quantitative skills, but also for computational skills. The notebook also shows how computational notebooks can assist in developing and delivering a suite of geo-computational modules to enhance data science and analytics skills.

Together, the articles in the Special Issue demonstrate how notebooks can be used to introduce the operation of open software and application of novel methods, enhance student learning, increase interactivity and exploration of research outputs, and how to produce replicable, reproducible and transparent research. We encourage submissions to this new form of publication. We believe that computational notebooks offer an exciting new platform adhering to REGION's principles of open, reproducible and transparent science, and anticipate a change in the future of academic publishing in this direction.

Acknowledgments

Sergio J. Rey acknowledges the support of NSF-SES 1831615.

References

Arribas-Bel D, Green M, Rowe F, Singleton A (2020) Open data products – A framework for creating valuable analysis ready data. *Journal of Geographical Systems*. in review

Begley CG, Ellis LM (2012) Drug development: Raise standards for preclinical cancer research. *Nature* 483: 531–533. CrossRef.

Begley S (2012) In cancer science, many "discoveries" don't hold up. Reuters. http://www.reuters.com/article/us-science-cancer-idUSBRE82R12P20120328

Boeing G (2019) Urban street network analysis in a computational notebook. *REGION* 6: 39–51. CrossRef.

Branco A (2012) Reliability and meta-reliability of language resources: Ready to initiate the integrity debate? The 12th workshop on treebanks and linguistic theories (tlt12)

Branco A, Bretonnel Cohen K, Vossen P, Ide N, Calzolari N (2017) Replicability and reproducibility of research results for human language technology: Introducing an LRE special section. *Language Resources & Evaluation* 51: 1–5. CrossRef.

Chen M, Fahrner D, Arribas-Bel D, Rowe F (2020) A reproducible notebook to acquire, process and analyse satellite imagery: Exploring long-term urban changes. *REGION* 7: 15–46. CrossRef.

Comber S (2019) Demonstrating the utility of machine learning innovations in address matching to spatial socio-economic applications. *REGION* 6: 17–37. CrossRef.

Editorial (2013a) Announcement: Reducing our irreproducibility. Nature News Nature

Editorial (2013b) Unreliable research: Trouble at the lab. The Economist. http://-www.economist.com/news/briefing/21588057-scientists-think-science-self-correcting-alarming-degree-it-not-trouble

Fanelli D (2009) How many scientists fabricate and falsify research? A systematic review and metaanalysis of survey data. *PloS ONE* 4: e5738. CrossRef.

Fokkens A, van Erp M, Postma M, Pedersen T, Vossen P, Freire N (2013) Offspring from reproduction problems: What replication failure teaches us. *Proceedings of the 51st annual meeting of the association for computational linguistics* 1: 1691–1701

Ioannidis JP (2011) An epidemic of false claims. *Scientific American* 304: 16–16. CrossRef.

Jupyter Project (2019) Jupyer notebooks. Available at: https://jupyter.org (accessed: 1 September 2019)

Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, Jupyter Development Team (2016) Jupyter notebooks – A publishing format for reproducible computational workflows. In: Loizides F, Schmidt B (eds), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS-Press, Amsterdam, The Netherlands, 87–90. CrossRef.

Koster S, Rowe F (2019) Fueling research transparency: Computational notebooks and the discussion section. *REGION* 6: 1–2. CrossRef.

Naik G (2011) Scientists' elusive goal: Reproducing study results. Wall Street Journal, December 2 2011, a1

Open Science Collaboration (2015) PSYCHOLOGY. Estimating the reproducibility of psychological science. *Science* 349: 943–950. CrossRef.

Parente P (2019) Estimate of public Jupyter notebooks on GitHub. Github, available at: https://github.com/parente/nbestimate (accessed: 5 September 2019)

Patias N (2019) Exploring long-term youth unemployment in Europe using sequence analysis: A reproducible notebook approach. *REGION* 6: 53–69. CrossRef.

Peng R (2011) Reproducible research in computational science. *Science* 334: 1226–1228. CrossRef.

Pérez F, Granger B (2015) Computational narratives as the engine of collaborative data science. Blog. available at: https://blog.jupyter.org/project-jupyter-computationalnarratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58 (accessed: 10 September 2019)

Prinz F, Schlange T, Asadullah K (2011) Believe it or not: How much can we rely on published data on potential drug targets? *Nature Reviews Drug Discovery* 10: 712–712. CrossRef.

Reades J (2020) Teaching on Jupyter – Using notebooks to accelerate learning and curriculum development. *REGION* 7: 21–34. CrossRef.

Sarrias M (2020) Random parameters and spatial heterogeneity using Rchoice in R. *REGION* 7: 1–19. CrossRef.

Wieland T (2019) REAT: A regional economic analysis toolbox for R. *REGION* 6: R1–R57. CrossRef.

Zimmer C (2012) A sharp rise in retractions prompts calls for reform. The New York Times 16. http://www.nytimes.com/2012/04/17/science/rise-in-scientific-journal-retractions-prompts-calls-forreform.html

Articles

REGION
The Journal of ERSA
Powered by WU

Volume 6, Number 3, 2019, 39–51
DOI: 10.18335/region.v6i3.278

journal homepage: region.ersa.org

Urban Street Network Analysis in a Computational Notebook*

Geoff Boeing[1]

[1] University of Southern California, Los Angeles, USA

Received: 21 September 2019/Accepted: 20 December 2019

Abstract. Computational notebooks offer researchers, practitioners, students, and educators the ability to interactively conduct analytics and disseminate reproducible workflows that weave together code, visuals, and narratives. This article explores the potential of computational notebooks in urban analytics and planning, demonstrating their utility through a case study of OSMnx and its tutorials repository. OSMnx is a Python package for working with OpenStreetMap data and modeling, analyzing, and visualizing street networks anywhere in the world. Its official demos and tutorials are distributed as open-source Jupyter notebooks on GitHub. This article showcases this resource by documenting the repository and demonstrating OSMnx interactively through a synoptic tutorial adapted from the repository. It illustrates how to download urban data and model street networks for various study sites, compute network indicators, visualize street centrality, calculate routes, and work with other spatial data such as building footprints and points of interest. Computational notebooks help introduce methods to new users and help researchers reach broader audiences interested in learning from, adapting, and remixing their work. Due to their utility and versatility, the ongoing adoption of computational notebooks in urban planning, analytics, and related geocomputation disciplines should continue into the future.

Key words: Computational Notebook, Jupyter, OpenStreetMap, OSMnx, Python, Street Network, Urban Planning

1 Introduction

A traditional academic and professional divide has long existed between code creators and code users. The former would develop software tools and workflows for professional or research applications, which the latter would then use to conduct analyses or answer scientific questions. Today, however, these boundary lines increasingly blur as computation percolates throughout both the natural and social sciences. As quantitatively-oriented academics gradually shift away from monolithic, closed-source data analysis software systems like SPSS and ArcGIS, they increasingly embrace coding languages like R and Python to script and document their research workflows (Padgham et al. 2019). Developing shareable, reproducible, and recomputable scripts in R or Python to acquire, transform, describe, visualize, and model data, these researchers act as both code creators and code users.

*This paper is available as computational notebook on the REGION webpage.

An important trend in this methodological trajectory has been the widespread adoption of the computational notebook. A computational notebook is a computer file that replaces the traditional lab notebook and intersperses plain-language narrative, hyperlinks, and images with snippets of code in the paradigm of literate programming (Knuth 1992). These notebooks are easily distributed and integrate well with version control systems like Git because they are simply structured text files. They have pedagogical value in introducing students to computational thinking and coding techniques while thoroughly explaining each new programming language facet as it is introduced. They also offer research value in documenting data, questions, hypotheses, procedures, experiments, and results in detail alongside each's attendant computations (Pérez, Granger 2007, Kluyver et al. 2016).

Computational notebooks thus open up the world of analytics to a wider audience than was possible in the past. This particularly impacts disciplines that encompass diverse methodologies and skillsets. For example, urban planning, like many academic domains related or adjacent to regional science, comprises a broad set of scholars, students, and working professionals with a wide range of computational aptitude. Some urban planners focus on policymaking within the political constraints of city hall. Others employ qualitative methods to work in and with vulnerable communities. Others develop simulation models to forecast urbanization patterns and infrastructure needs. Others intermingle these, and many more, different approaches to understanding and shaping the city. Yet all urban planners benefit from basic quantitative literacy and an ability to reason critically with data. This scholarly and professional imperative aligns with the growing importance of computational thinking in the urban context and parallel trends in geocomputation (Harris et al. 2017), geographic data science (Kang et al. 2019, Poorthuis, Zook 2019, Singleton, Arribas-Bel 2019), and the open-source/open-science movements (Rey 2019).

Urban planning and its related disciplines benefit accordingly from the growing adoption of computational notebooks in pedagogy, research, and practice. Computation is increasingly central to the field and its practitioners benefit from open and reproducible approaches to analyzing urban data and predicting city futures (Kedron et al. 2019, Kontokosta 2018, Batty 2019). In the Python universe, for example, numerous new tools now exist to support urban analytics and planning processes, including data wrangling/analysis (pandas), visualization (matplotlib), geospatial wrangling/analysis (geopandas), spatial data science and econometrics (pySAL), mapping (cartopy), web mapping (folium), network analysis (NetworkX), land use modeling/simulation (UrbanSim), activity-based travel modeling (ActivitySim), and computational notebooks themselves (Jupyter).

Another Python tool useful for urban planning research and practice – and the primary focus of this article – is OSMnx, a package for street network analysis (Boeing 2017). OSMnx allows users to download spatial data (including street networks, other networked infrastructure, building footprints, and points of interest) from OpenStreetMap then model, analyze, and visualize them. To introduce new users to its functionality and capabilities, OSMnx's official demos and tutorials are developed and maintained in Jupyter notebook format. This repository in turn offers a compelling case study of the potential of computational notebooks to document and disseminate geospatial software tools.

This article introduces OSMnx as a computational tool for urban street network analysis by way of these computational notebooks. It describes their repository and highlights examples from them, inline here, to illustrate the use and value of computational notebooks. To do so, it demonstrates how to interactively execute the code in this article itself by using Docker to run a containerized computational environment including Jupyter Lab as an interactive web-based interface. The article is organized as follows. First, it presents the repository containing OSMnx's demo and tutorial notebooks. Then it describes how to run OSMnx's computational environment via Docker. Next it demonstrates the use of OSMnx interactively in the article itself through a synoptic tutorial adapted from this repository. Finally, it concludes by discussing the prospects of notebooks for facilitating the adoption of computational workflows in urban analytics and planning.

2 The OSMnx Examples Repository

OSMnx's official demos, tutorials, and examples are in Jupyter notebook format in a GitHub repository. The repository's root contains a license file, a readme file, an environment definition file, repository contributing guidelines, and a notebooks folder. Within that folder, the repository contains 19 thematically organized Jupyter notebook files that collectively provide a short self-directed tutorial-style course in using OSMnx. The following notebooks are included there:

1. An introductory survey of features
2. A more comprehensive overview of OSMnx's basic functionality
3. Using OSMnx to produce shapefiles
4. Modeling and visualizing street networks in different places at different scales
5. Using OSMnx's network topology cleaning and simplification features
6. Saving and loading data to/from disk with OSMnx
7. Conducting street network analyses with OSMnx and its NetworkX dependency
8. Visualizing street networks and study sites
9. Working with dual graphs of street networks
10. Producing figure-ground diagrams for urban form analysis
11. Working with building footprints
12. Interactive web mapping of street networks and routes
13. Attaching elevations to the network and calculating street grades
14. Working with isolines and isochrones
15. Cleaning complex street intersections
16. Calculating street bearings
17. Working with other types of spatial infrastructure
18. Visualizing street network orientation with polar histograms
19. Interfacing between OSMnx and igraph for fast algorithm implementations in the C language

This resource is useful for introducing users to the OSMnx software package, demonstrating how to download, model, analyze, and visualize street networks in Python, and illustrating several basic and intermediate spatial network analyses. To run the code examples in this resource repository, one must have access to a Python installation with the code dependencies installed, including Jupyter itself for running the notebook files. Two primary options exist for installing this computational environment. The first is installing Python locally, then configuring it and installing all the necessary packages and dependencies. This can be time-consuming and requires some prior experience beyond the scope of this article. The second, and easier, option is to simply run everything in a pre-built Docker container. This latter option is detailed in the following section.

3 The Computational Environment

The OSMnx project's reference Docker image contains a stable, consistent computational environment for running OSMnx on any computer. Docker is a virtualization tool that allows complex software stacks to be delivered as self-contained packages called images, allowing users to run software without having to compile or install a complex chain of dependencies. Instead, users install Docker on their computer then tell it to run a certain image as an instance called a container.

This article can be read in its static form (i.e., HTML or PDF) or it can be executed interactively (i.e., via its .ipynb Jupyter notebook file). For interactive execution, install Docker and run the official OSMnx container as follows. First, download and install Docker Desktop. Once it is installed and running on your computer, open Docker's settings/preferences and ensure that your local drives are shared with Docker so the container has access to the notebook file. Then run the OSMnx Docker container (which contains a Python installation and all the packages needed to run OSMnx, including Jupyter Lab) by following the platform-specific instructions below.

If you are on *Windows* open a command prompt, change directory to the location of this notebook file then run:

```
docker run --rm -it -p 8888:8888 -v "\%cd\%":/home/jovyan/work gboeing/osmnx:v10
```

If you are on *Mac/Linux* open a terminal window, change directory to the location of this notebook file then run:

```
docker run --rm -it -p 8888:8888 -v "$PWD":/home/jovyan/work gboeing/osmnx:v10
```

Once the container is running per these instructions, open your computer's web browser and visit http://localhost:8888 to access Jupyter Lab and open this article's notebook file.

4 Street Network Analysis with OSMnx

Here we showcase the resource repository inline to demonstrate potential applications. In particular, we highlight specific material from its notebooks (enumerated above), adapting their code into this interactive article to introduce OSMnx and illustrate some of the capabilities of a computational notebook.

First we import the necessary Python modules:

[1]:
```
import matplotlib.cm as cm
import matplotlib.colors as colors
import networkx as nx
from IPython.display import Image
from pprint import pprint
```

matplotlib is a package for data visualization and plotting. NetworkX is a package for generic network analysis. IPython provides interactive computing and underpins our Python-language Jupyter environment (Pérez, Granger 2007). pprint allows us to "pretty print" Python data structures to make them easier to read inline.

Next we import OSMnx itself, configure it, and display its version number:

[2]:
```
import osmnx as ox
ox.config(log_console=True, use_cache=True)
ox.__version__
```

[2]:
```
'0.10'
```

The configuration step tells OSMnx to log its actions to the terminal window and to use a cache. This cache saves a local copy of any data downloaded by OSMnx to prevent re-downloading the same data each time the code is run.

Next we use OSMnx to download the street network of Piedmont, California, construct a graph model of it (via NetworkX), then plot the network with the `plot_graph` function (which uses matplotlib under the hood):

[3]:
```
# create a graph of Piedmont's drivable street network then plot it
G = ox.graph_from_place('Piedmont, California, USA', network_type='drive')
fig, ax = ox.plot_graph(G)
```

[3]:
```
For the output see Figure 1
```

In the resulting Figure 1, the network's intersections and dead-ends (i.e., graph nodes) appear as light blue circles and its street segments (i.e., graph edges) appear as gray lines. This is the street network within the municipal boundaries of the city of Piedmont, California. We select this study site for pedagogical purposes as it is a relatively small, self-contained municipality and lends itself to convenient visualization and indicator calculation here. Note that we specified `network_type='drive'` so this is specifically the drivable network in the city. OSMnx can also automatically download and model walkable and bikeable street networks by changing this argument.

4.1 Calculating Network Indicators

Now that we have a model of the network, we can calculate some statistics and indicators. First, what area does our network cover in square meters? To calculate this, we project the graph, convert its projected nodes to a geopandas GeoDataFrame, then calculate the area of the convex hull of this set of node points in the Euclidean plane:

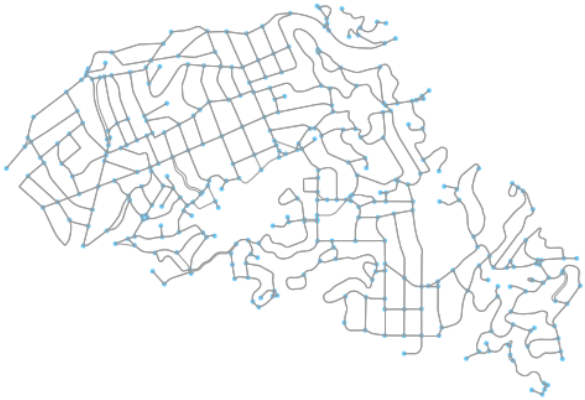


Figure 1: Output from codebox 3

[4]:
```
# project graph then calculate its nodes' convex hull area
G_proj = ox.project_graph(G)
nodes_proj = ox.graph_to_gdfs(G_proj, edges=False)
graph_area_m = nodes_proj.unary_union.convex_hull.area
graph_area_m
```

[4]:
```
4224782.349449131
```

Thus, this network covers approximately 4.2 square kilometers. When projecting graphs, OSMnx by default uses the Universal Transverse Mercator (UTM) coordinate system and automatically determines the UTM zone for projection based on the network's centroid. Other coordinate reference systems can be defined by the user to customize this projection behavior.

Next, we compute and inspect some basic stats about the network:

[5]:
```
# calculate and print basic network stats
stats = ox.basic_stats(G_proj, area=graph_area_m, clean_intersects=True,
        circuity_dist='euclidean')
pprint(stats)
```

[5]:
```
{'circuity_avg': 1.11354525174028,
 'clean_intersection_count': 271,
 'clean_intersection_density_km': 64.1453162753664,
 'edge_density_km': 26951.828421373437,
 'edge_length_avg': 121.39190724946685,
 'edge_length_total': 113865.60899999991,
 'intersection_count': 312,
 'intersection_density_km': 73.84995822108604,
 'k_avg': 5.421965317919075,
 'm': 938,
 'n': 346,
 'node_density_km': 81.89771007851208,
 'self_loop_proportion': 0.006396588486140725,
 'street_density_km': 14061.652905680734,
 'street_length_avg': 121.23963877551029,
 'street_length_total': 59407.42300000004,
 'street_segments_count': 490,
 'streets_per_node_avg': 2.953757225433526,
 'streets_per_node_counts': {0: 0, 1: 34, 2: 0, 3: 263, 4: 47, 5: 1, 6: 1},
 'streets_per_node_proportion': {0: 0.0,
                                 1: 0.09826589595375723,
                                 2: 0.0,
                                 3: 0.7601156069364162,
                                 4: 0.13583815028901733,
                                 5: 0.002890173410404624,
                                 6: 0.002890173410404624}}
```

For example, we can see that this network has 346 nodes (n) and 938 edges (m). The

streets in this network are 11% more circuitous (*circuity_avg*) than straight-line would be. The average street segment length is 121 meters (*street_length_avg*). We can inspect more stats, primarily topological in nature, with the `extended_stats` function. As the results of many of these indicators are verbose (i.e., calculated at the node-level), we print only the indicators' names here:

[6]:
```
# calculate and print extended network stats
more_stats = ox.extended_stats(G, ecc=True, bc=True, cc=True)
for key in sorted(more_stats.keys()):
    print(key)
```

[6]:
```
avg_neighbor_degree
avg_neighbor_degree_avg
avg_weighted_neighbor_degree
avg_weighted_neighbor_degree_avg
betweenness_centrality
betweenness_centrality_avg
center
closeness_centrality
closeness_centrality_avg
clustering_coefficient
clustering_coefficient_avg
clustering_coefficient_weighted
clustering_coefficient_weighted_avg
degree_centrality
degree_centrality_avg
diameter
eccentricity
pagerank
pagerank_max
pagerank_max_node
pagerank_min
pagerank_min_node
periphery
radius
```

The average neighborhood degree indicators refer to the mean degree of nodes in the neighborhood of each node. The centrality indicators (betweenness, closeness, degree, and PageRank) identify how "central" or important each node is to the network in terms of its topological structure. The clustering coefficient indicators represent the extent to which a node's neighborhood forms a complete graph. The extended stats also include the network's eccentricity (the maximum distance from each node to all other nodes), diameter (maximum eccentricity in the network), radius (minimum eccentricity in the network), center (set of all nodes whose eccentricity equals the radius), and periphery (set of all nodes whose eccentricity equals the diameter). Additional information about the various indicators is available online in OSMnx's documentation.

Now that we have modeled the street network and computed various indictors of its geometry and topology, we can finally save our graph to disk as an ESRI shapefile or a GraphML file (an open-source format for graph serialization), allowing easy re-use in other GIS or network analysis software:

[7]:
```
# save the network model to disk as a shapefile and graphml
ox.save_graph_shapefile(G, filename='mynetwork_shapefile')
ox.save_graphml(G, filename='mynetwork.graphml')
```

4.2 Visualizing Street Centrality

OSMnx is built on top of NetworkX, a powerful network analysis package developed at Los Alamos National Laboratory (Hagberg et al. 2008). We can use it to calculate and visualize the closeness centrality of different streets in the network. Closeness centrality measures how central a node or edge is in a network and is defined as the reciprocal of the sum of the distance-weighted shortest paths between the node/edge and every other node/edge in the network.

First, we convert our graph to its line graph (sometimes called the *dual graph*; see Porta et al. 2006) which inverts its topological definitions such that streets become nodes

Figure 2: Output from codebox 9

and intersections become edges. Then we calculate the closeness centrality of each node (i.e., street in the line graph):

[8]:
```
# calculate node closeness centrality of the line graph
edge_centrality = nx.closeness_centrality(nx.line_graph(G))
```

Now that we have calculated the centrality of each street in the network, we visualize it with matplotlib via OSMnx's `plot_graph` function, using the inferno color map to represent the most-central streets in bright yellow and the least-central streets in dark purple (see Figure 2):

[9]:
```
# make a list of graph edge centrality values
ev = [edge_centrality[edge  (0,)] for edge in G.edges()]

# create a color scale converted to list of colors for graph edges
norm = colors.Normalize(vmin=min(ev)*0.8, vmax=max(ev))
cmap = cm.ScalarMappable(norm=norm, cmap=cm.inferno)
ec = [cmap.to_rgba(cl) for cl in ev]

# color the edges in the original graph by closeness centrality in line graph
fig, ax = ox.plot_graph(G, bgcolor='black', axis_off=True, node_size=0,
                        edge_color=ec, edge_linewidth=2, edge_alpha=1)
```

[9]:
```
For the output see Figure 2
```

4.3 Network Routing

OSMnx allows researchers and practitioners to calculate routes and simulate trips along the network using various shortest-path algorithms, such as that by Dijkstra (1959). We demonstrate this here. First we use OSMnx to find the network nodes nearest to two latitude-longitude points:

[10]:
```
# find the network nodes nearest to two points
orig_node = ox.get_nearest_node(G, (37.825956, -122.242278))
dest_node = ox.get_nearest_node(G, (37.817180, -122.218078))
```

Next we compute the shortest path between these origin and destination nodes using Dijkstra's algorithm weighted by length (i.e., geometric distance along the street network). Then we use OSMnx to plot this route along the network:

[11]:
```
# calculate the shortest path between these nodes then plot it
route = nx.shortest_path(G, orig_node, dest_node, weight='length',
        method='dijkstra')
fig, ax = ox.plot_graph_route(G, route, node_size=0)
```

[11]:
```
For the output see Figure 3
```

Finally, we can calculate some statistics of our route, including its total length, in meters:

Figure 3: Output from codebox 11

[12]:
```
# what is the network distance of this route?
net_dist = nx.shortest_path_length(G, orig_node, dest_node, weight='length',
           method='dijkstra')
net_dist
```

[12]:
```
3284.0989999999997
```

Thus, this trip would travel approximately 3.3 kilometers along the network. We can also calculate the straight-line distance between these two network nodes as-the-crow-flies, using OSMnx's vectorized great-circle calculator:

[13]:
```
# what is the straight-line distance from origin to destination?
sl_dist = ox.great_circle_vec(G.node[orig_node]['y'], G.node[orig_node]['x'],
                              G.node[dest_node]['y'], G.node[dest_node]['x'])
sl_dist
```

[13]:
```
2340.8766018171827
```

Comparing these two distance values, we can compute an indicator of trip circuity: that is, how much greater the network-constrained distance is between two nodes compared to the straight-line distance between them. In this case, we can see that the network distance is approximately 40% longer than the straight-line distance:

[14]:
```
# how much longer is the network distance than the straight-line?
net_dist / sl_dist
```

[14]:
```
1.4029355487814306
```

4.4 Downloading/Modeling Networks in Other Ways

So far, we have modeled and analyzed the street network of Piedmont, California. However, we are not constrained to study sites in the United States. OpenStreetMap is a global mapping project and OSMnx can model networks anywhere in the world, such as Modena, Italy:

[15]:
```
# create a graph of Modena's drivable street network then plot it
G = ox.graph_from_place('Modena, Italy', retain_all=True)
fig, ax = ox.plot_graph(G, fig_height=8, node_size=0, edge_linewidth=0.5)
```

[15]:
```
For the output see Figure 4
```

We have seen how to download street network data and turn it into a graph-based model using OSMnx's `graph_from_place` function. This function geocodes the place name using OpenStreetMap's Nominatim web service, identifies its bounding polygon, then downloads all the network data within this polygon from OpenStreetMap's Overpass API. This workflow easily handles well-defined place names. However, OSMnx offers additional functionality to download and model networks for other study sites as well.

For example, if OpenStreetMap does not have a bounding polygon for a specific study site, we can acquire its street network anyway by passing a polygon directly into the

Figure 4: Output from codebox 15

`graph_from_polygon` function. Or we can pass in latitude-longitude coordinates and a distance into the `graph_from_point` function as demonstrated here, where we visualize the network within a bounding box around the University of California, Berkeley's Wurster Hall:

[16]:
```
# create a graph around UC Berkeley then plot it
wurster_hall = (37.870605, -122.254830)
one_mile = 1609 #one mile in meters
G = ox.graph_from_point(wurster_hall, distance=one_mile, network_type='drive')
fig, ax = ox.plot_graph(G, node_size=0)
```

[16]:
```
For the output see Figure 5
```

OSMnx also accepts place queries as unambiguous Python dictionaries to help the geocoder find a specific matching study site when several names might approximately overlap. In this example, we download the street network of San Francisco, California by defining the query with such a dictionary:

[17]:
```
# create a graph of San Francisco's drivable street network then plot it
place = {'city'   : 'San Francisco',
         'state'  : 'California',
         'country': 'USA'}
G = ox.graph_from_place(place, network_type='drive')
```

4.5 Downloading Other Infrastructure Types

All of the preceding examples have focused on urban and suburban street networks. However, OSMnx can also download and model other networked infrastructure types by passing in custom queries via the `infrastructure` argument. Such networked infrastructure could include power lines, the canal systems of Venice or Amsterdam, or the New York City subway's rail infrastructure as illustrated in this example:

[18]:
```
# create a graph of NYC's subway rail infrastructure then plot it
G = ox.graph_from_place('New York City, New York, USA',
            retain_all=False, truncate_by_edge=True, simplify=True,
            network_type='none', infrastructure='way["railway"~"subway"]')

fig, ax = ox.plot_graph(G, node_size=0)
```

[18]:
```
For the output see Figure 6
```

Figure 5: Output from codebox 16

Note that the preceding code snippet modeled subway rail *infrastructure* which thus includes crossovers, sidings, spurs, yards, and the like. For a station-based train network model, the analyst would be best-served downloading and modeling a station adjacency matrix.

Beyond networked infrastructure, OSMnx can also work with OpenStreetMap building footprint and points of interest data. For example, we can download and visualize the building footprints near New York's Empire State Building:

[19]:
```
# download and visualize the building footprints around the empire state bldg
point = (40.748482, -73.985402) #empire state bldg coordinates
dist = 812 #meters
gdf = ox.footprints_from_point(point=point, distance=dist)
gdf_proj = ox.project_gdf(gdf)
bbox_proj = ox.bbox_from_point(point=point, distance=dist, project_utm=True)
fig, ax = ox.plot_footprints(gdf_proj, bbox=bbox_proj, bgcolor='#333333',
        color='w', figsize=(6,6))
```

[19]:
```
For the output see Figure 7
```

Finally, we can download and inspect the amenities matching the tag "restaurants" near the Empire State Building and then display the five most common cuisine types

[20]:
```
# download restaurants near the empire state bldg then display them
gdf = ox.pois_from_point(point=point, distance=dist, amenities=['restaurant'])
gdf[['name', 'cuisine']].dropna().head()
```

[20]:
```
                              name        cuisine
357620442 Dolcino Trattoria Toscana        italian
419359995              Little Alley        chinese
419367625              Ramen Takumi japanese;ramen
561042187                Les Halles         french
663104998           Tick Tock Diner          diner
```

[21]:
```
# show the five most common cuisine types among these restaurants
gdf['cuisine'].value_counts().head()
```

[21]:
```
indian      22
korean      15
italian     14
japanese    13
pizza        9
Name: cuisine, dtype: int64
```

Figure 6: Output from codebox 18

Figure 7: Output from codebox 19

5 Conclusion

This article argued that computational notebooks underpin an important emerging pillar in urban analytics and planning research, pedagogy, and practice. To demonstrate this, it presented the official repository of computational notebooks that the OSMnx project uses for tutorials, demos, and guides. It illustrated the use of these notebooks by highlighting specific examples from them, inline and interactively within this article, as an introduction to this modeling and analysis software. OSMnx itself is a Python package for downloading, modeling, analyzing, and visualizing data from OpenStreetMap. It lets users analyze networked infrastructure like street networks as well as building footprints, points of interest, elevation data, and more. This article demonstrated how computational notebooks can provide a tutorial-style introduction to scientific software such as this.

The OSMnx project uses computational notebooks because they offer several advantages. First, they empower scientific reproducibility, replication, sharing, and remixing. Second, they allow researchers to intermingle data analyses with visualizations and narratives to ask and answer research questions. Third, they offer "follow-along" guides for introducing software and methods to new users, such as in this repository for OSMnx or even in the university classroom. Finally, they help researchers reach a wider community

of interest by making their methodologies and analyses more legible to a broad audience potentially interested in adapting and remixing their work. For these reasons and more, we expect to see growing adoption of computational notebooks in the urban planning discipline and related analytics fields.

References

Batty M (2019) Urban analytics defined. *Environment and Planning B: Urban Analytics and City Science* 46[3]: 403–405. CrossRef.

Boeing G (2017) Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65: 126–139. CrossRef.

Dijkstra E (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1[1]: 269–271. CrossRef.

Hagberg A, Schult D, Swart P (2008) Exploring network structure, dynamics, and function using networkx. In: Varoquaux G, Vaught T, Millman J (eds), *Proceedings of the 7th Python in Science Conference*. SciPy, Pasadena, CA, 11–16

Harris R, O'Sullivan D, Gahegan M, Charlton M, Comber L, Longley P, Brunsdon C, Malleson N, Heppenstall A, Singleton A, Arribas-Bel D, Evans A (2017) More bark than bytes? reflections on 21+ years of geocomputation. *Environment and Planning B: Urban Analytics and City Science* 44[4]: 598–617. CrossRef.

Kang W, Oshan T, Wolf L, Boeing G, Frias-Martinez V, Gao S, Poorthuis A, Xu W (2019) A roundtable discussion: Defining urban data science. *Environment and Planning B: Urban Analytics and City Science* 46[9]: 1756–1768. CrossRef.

Kedron P, Frazier A, Trgovac A, Nelson T, Fotheringham A (2019) Reproducibility and replicability in geographical analysis. *Geographical Analysis*. CrossRef.

Kluyver T, Ragan-Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Jupyter Development Team (2016) Jupyter notebooks: A publishing format for reproducible computational workflows. In: Loizides F, Schmidt B (eds), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, Amsterdam, The Netherlands, 87–90. CrossRef.

Knuth D (1992) *Literate Programming*. Center for the Study of Language and Information, Stanford, CA

Kontokosta C (2018) Urban informatics in the science and practice of planning. *Journal of Planning Education and Research*. CrossRef.

Padgham M, Boeing G, Cooley D, Tierney N, Sumner M, Phan T, Beare R (2019) An introduction to software tools, data, and services for geospatial analysis of stroke services. *Frontiers in Neurology* 10[743]. CrossRef.

Pérez F, Granger B (2007) Ipython: A system for interactive scientific computing. *Computing in Science & Engineering* 9[3]: 21–29. CrossRef.

Poorthuis A, Zook M (2019) Being smarter about space: Drawing lessons from spatial science. *Annals of the American Association of Geographers*. CrossRef.

Porta S, Crucitti P, Latora V (2006) The network analysis of urban streets: A dual approach. *Physica A: Statistical Mechanics and Its Applications* 369[2]: 853–866. CrossRef.

Rey S (2019) Pysal: The first 10 years. *Spatial Economic Analysis* 14[3]: 273–282. CrossRef.

Singleton A, Arribas-Bel D (2019) Geographic data science. *Geographical Analysis*. CrossRef.

A Appendix

The interested reader may consult the following web sites for more information and resources as discussed in the article:

- OSMnx examples repository: https://github.com/gboeing/osmnx-examples
- OSMnx documentation: https://osmnx.readthedocs.io/
- Docker Desktop is available at: https://www.docker.com/products/docker-desktop
- The OSMnx Docker image is available at: https://hub.docker.com/r/gboeing/osmnx

REGION
The Journal of ERSA
Powered by WU

Volume 7, Number 1, 2020, 1–19
DOI: 10.18335/region.v7i1.279

journal homepage: region.ersa.org

Random Parameters and Spatial Heterogeneity using *Rchoice* package in *R*

Mauricio Sarrias[1]

[1] Universidad Católica del Norte, Antofagasta, Chile

Received: 17 September 2019/Accepted: 4 February 2020

Abstract. This document provides a brief introduction to models with spatial heterogeneity using a random parameter approach. Specifically, this paper shows how this modelling strategy can be used to capture and model spatial heterogeneity and locally varying coefficients for different latent structure. To show the main advantages of this modeling strategy, the `Rchoice` package (Sarrias 2016) in `R` is used. The examples will be focused on the ordered probit model with spatially varying coefficients using self-assessed health status as the dependent variable.

1 Introduction

Regional scientists, as well as many social researchers concerned on spatial relationships, analyze how the reciprocal geographical interaction of social agents generates spatial autocorrelation, affecting the bias and efficiency of standard econometric estimators. After Anselin (1988), a large number of papers dealt with the spatial autocorrelation using spatial versions of standard linear regression models, namely Spatial Autoregresive Regression (SAR) or Spatial Error Model (SEM) and even recent contributions extend the analysis toward Spatial Panel Data (Kelejian, Prucha 1998, 1999, Elhorst 2014). However, spatial interaction also is manifested through spatially varying coefficients referred to as: "structural instability over space, in the form of different response functions or systematically varying parameters" (Anselin 1988). In spite of the relevance of the concept, the evolution and development of econometric models that attempt to capture and model spatial heterogeneity has not been as euphoric as those focused on the spatial autocorrelation. The few attempts to capture this heterogeneity can be summarized by the spatial expansion method (SEM) (Casetti 1972), Geographically Weighted Regression (GWR) (Brunsdon et al. 1998) or assuming that the local relationship varies randomly over geographical space, a method also known as the Random Coefficient Model (RCM) (Swamy 1971). Each one of these methods enable estimation of model parameters locally, or they allow model parameter to vary as a function of location[1].

The three methods presented above share an important limitation: they require aggregating the variables at the location level. Therefore, we are prevented from using data at the individual level and capturing the spatial heterogeneity, simultaneously. This raises concerns about the misleading conclusions that can be derived at the individual level by using aggregate variables known as the ecological fallacy problem (Robinson 1950). A potential solution for this constraint is provided by multilevel modeling[2]. This

[1] For further review see for example Fotheringham, Brunsdon (1999).
[2] For other quantitative methods that avoid the ecological fallacy problem see Withers (2001).

approach separates the effect of personal and place characteristics to investigate the extent and nature of spatial variation in individual outcome measures (Goldstein 1987). The main drawback of multilevel modeling is that usually the random coefficients are assumed to be normally distributed. This makes the estimation process easier, but creates other problems. For example, this assumption implies that some locations might have positive or negative coefficients, whether or not this is true. In practice, this implies that occasionally researchers find sign reversals that are counterintuitive and difficult to explain. Furthermore, the domain of the normal distribution is $(-\infty, +\infty)$, which results in unreliable extreme coefficients and high coefficient variability. Those problems have also been found when applying the GWR approach (Jetz et al. 2005)[3].

This study focusses on models with spatially varying coefficients using simulation as in Sarrias (2019) and Train (2009). This modeling strategy is intended to complement the existing approaches by using variables at the micro level – overcoming the problem associated with spatial aggregation – and by adding flexibility and realism to the potential domain of the coefficient on the geographical space. Spatial heterogeneity is modelled by allowing the parameters associated with each observed variable to vary "randomly" across space according to some distribution $g(\cdot)$. However, it is not known how the parameters vary across space. All that is known is that they vary locally with population probability density function (pdf) $g(\cdot)$, which is assumed to be well behaved and continuous.

To show the main advantages of this modeling strategy, the `Rchoice` package (Sarrias 2016) in `R` is used. The examples will be focused on the ordered probit model with spatially varying coefficients using self-assessed health status as the dependent variable.

The remainder of this paper is organized as follows. Section 2 discusses the modelling approach for incorpo-rating continuous spatial heterogeneity using a random parameter approach. The main R packages needed for the examples are described in Section 3. The example using `Rchoice` package in `R` is presented in Section 4. Finally, Section 5 concludes.

2 Modelling approach

2.1 Continuous spatial heterogeneity

Consider the following structural model:

$$\begin{aligned} y_{ci}^{*} &= \boldsymbol{x}_{ci}'\boldsymbol{\beta}_c + \epsilon_{ci} \qquad c = 1, \ldots, C; \quad i = 1, \ldots, n_c \\ \boldsymbol{\beta}_c &\sim g(\boldsymbol{\beta}_c) \end{aligned} \tag{1}$$

where y_{ci}^{*} is a latent (unobserved) process for individual i in geographical area c (e.g, region, city, country, census track) that we are trying to explain; $\boldsymbol{x}_{ci}$ is a $K \times 1$ vector of individual and regional variables; and ϵ_i is the error term[4]. It is assumed that the vector $(y_{ci}, \boldsymbol{x}_{ci}', \boldsymbol{\beta}_c')'$ is independently and identically distributed. The conditional probability density function of the latent process, $f^{*}(y_{ic}|\boldsymbol{x}_{ci}, \boldsymbol{\epsilon}_c)$, is determined once the nature of the observed y_{ci} and the population pdf of ϵ_i is known. For example, if the observed y_{ci} is binary and ϵ_i is normal distributed, we obtain the traditional probit model. But if ϵ_i is distributed as logistic, then we obtain the binary logit model. Due to space restrictions, the applied example in this study will focus on the ordered model.

The key element in the structural model is $\boldsymbol{\beta}_c$. The notation implies that coefficients are associated with region c, representing those region-specific partial correlations on the latent dependent variable. Thus, all individuals located in the same region have the same coefficient, but there exists inter-spatial heterogeneity, i.e., the coefficients vary across regions but not within the region.

[3] There are some interesting extensions that have been recently developed. For example, Dong et al. (2015) extend the traditional multilevel models to incorporate spatial interaction effects at different level units. Dong et al. (2018) extend the GWR for ordinal categorical responses. Bayesian spatially varying coefficient models have been also suggested by Finley (2011) and Gelfand et al. (2003).

[4] Throughout this work I will use location unit, region, or geographical area interchangeably to refer to the subindex c.

However, we do not know how these parameters vary across regions. All we know is that they vary locally with population pdf $g(\boldsymbol{\beta}_c)$. Once $g(\boldsymbol{\beta}_c)$ is specified, we might have a fully parametric or a semi-parametric spatially random parameter model.

2.2 Choosing the distribution

Continuous spatial heterogeneity is introduced by assuming that the parameters vary "randomly" across regions according to some pre-specified "continuous" distribution. The pdf of the spatially random coefficients in the population is $g(\boldsymbol{\beta}_c|\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents, for example, the mean and variance of $\boldsymbol{\beta}_c$. The goal for the researcher is to estimate $\boldsymbol{\theta}$.

The distribution of the spatially random parameters can in principle take any shape. The researcher has to choose a priori the distribution according to his beliefs of the domain and boundedness of the coefficients.

Therefore, some prior theoretical knowledge of the spatial structure being modeled may lead to a more appropriate choice of the distribution. Below, some continuous distributions and their implications are discussed.

Normal Distribution: The normal distribution is by far the most widely used distribution for the spatially random parameters. The density of the normal parameter has mean β and standard deviation σ_β, so that $\boldsymbol{\theta} = (\beta, \sigma_\beta)'$. Thus, the coefficient for each region can be written as $\beta_c = \beta + \sigma_\beta \eta_c$, where $\eta_c \sim N(0,1)$. An important feature of the normal density is its unboundedness. This implies that every real number has a positive probability of being drawn. Thus, specifying a given coefficient to follow a normal distribution is equivalent to making the a priori assumption that there is a proportion of regions with positive coefficients and another proportion with negative ones. As an illustration, consider a normally distributed coefficient with population parameters $\beta = 0.5$ and $\sigma_\beta = 1$. The proportion of regions with positive coefficients is approximately $\Phi(\beta/\sigma_\beta) \cdot 100 \approx 70\%$. This last fact makes this distribution quite suitable when the researcher assumes that the effect of x_k on y^* can have both signs depending in the local context of each region. For example, there exists an extensive literature that uses the city population as a proxy for urbanization economies (see for example Duranton, Puga 2004). However, in some regions, a large population may suggest agglomeration economies, while in others, it may suggest congestion effects (Ali et al. 2007). In other words, β_c for the population density can take positive or negative values across space. The normal distribution can be also used as an initial exploratory analysis to determine the domain of a coefficient. For example, if the estimated parameters are $\hat{\beta} = 2$ and $\hat{\sigma}_\beta = 1$, this implies that approximately $\Phi(\hat{\beta}/\hat{\sigma}_\beta) \cdot 100 \approx 98\%$ of the regions in the sample have a positive coefficient. Therefore, the researcher may be more inclined to choose a distribution with just a positive real domain. One major disadvantage of the normal distribution is that it has infinite tails, which might result in some regions having implausible extreme coefficient values.

Triangular Distribution: This is a continuous probability distribution with probability density function shaped like a triangle. The advantage of this distribution is that it has a definite upper and lower limit, so its tails are shorter than the normal distribution and we avoid extreme coefficients that may result for some regions. The density of a triangular distribution with mean β and spread s_β is zero beyond the range $(\beta - s_\beta, \beta + s_\beta)$, rises linearly from $\beta - s_\beta$ to β, and drops linearly to $\beta + s_\beta$. The parameters $\boldsymbol{\theta} = (\beta, s_\beta)'$ are estimated.

Uniform Distribution: In this case the parameter for each location is equally likely to take on any value in some interval. Suppose that the spread of the uniform distribution is s_β, such that the parameter is uniformly distributed from $\beta - s_\beta$ to $\beta + s_\beta$. Then the parameter can be constructed as $\beta_c = \beta + s_\beta(2u_c - 1)$ where $u_c \sim U[0,1]$ and the parameters $\boldsymbol{\theta} = (\beta, s_\beta)$ are estimated. The new random draw $(2u_c - 1)$ is distributed as $U[-1,+1]$, therefore multiplying by s_β gives a uniformly distributed parameter $\pm$ s (Train 2009, Hensher, Greene 2003). The standard deviation of the uniform distribution can be derived from the spread by dividing s_β

by $\sqrt{3}$. Note also that the uniform distribution with a [0, 1] bound is very suitable when there exists spatial heterogeneity in a dummy variable. For this case the restriction is $\beta = s_\beta = 1/2$.

The normal, triangular and uniform distributions permit positive and negative coefficients. However, as I discussed above, the coefficient may present spatial heterogeneity but only in the positive or negative domain. For example, we may be confident that the coefficient for xk is positive for all regions, but still there may exist spatial heterogeneity around the mean. Some widely used distributions with domain in the positive numbers are the log-normal, truncated normal, and Johnson Sb distribution[5].

Log-normal Distribution: The support of the log-normal distribution is $(0, \infty)$. Formally, the coefficient for each region is specified as $\beta_c = \exp(\beta + \sigma_\beta \eta_c)$ where $\eta_c \sim N(0, 1)$. The parameters β and σ_β, which represent the mean and standard deviation of $\log(\beta_c)$, are estimated. The median, mean, and standard deviation of β_c are $\exp(\beta_c)$, $\exp(\beta_c + \sigma_\beta^2/2)$ and mean $\times\sqrt{\exp(\sigma_\beta^2) - 1}$, respectively (Revelt, Train 1998, Train 2009). The main drawback of the log-normal distribution is that it has a very long right-hand tail. This means that we might find regions with unreasonable extreme positive coefficients.

Truncated Normal Distribution: The domain of this distribution is $(0, \infty)$ if the normal distribution is truncated below at zero. The parameter for each region is created as $\beta_c = max(0, \beta + \sigma_\beta \eta_c)$ where $\eta_c \sim N(0, 1)$ with the share below zero massed at zero equal to $\Phi(-\beta/\sigma_\beta)$. A normal distribution truncated at 0 can be useful when the researcher has a priori belief that for some regions the marginal latent effect of the variable is null. The parameters $\boldsymbol{\theta} = (\beta, \sigma_\beta)$ are estimated.

Johnson S_b Distribution: The S_b distribution gives coefficients between 0 and 1, which is also very suitable for dummy variables. The parameter for region c is computed as $\beta_c = \frac{\exp(\beta + \sigma_\beta \eta_c)}{1 + \exp(\beta + \sigma_\beta \eta_c)}$ where $\eta_c \sim N(0, 1)$ and the parameters β and σ_β are estimated. The mean, variance, and shape are determined by the mean and variance of $\beta + \sigma_\beta \eta_c$ which is a normal distributed parameter. If the analyst needs the coefficient to be between 0 and k, then the variable can be multiplied by k. The logic behind this is the following. Since $\beta_c \times x_{ic}$ ranges between [0, 1], then $\beta_c \times k \times x_{ic}$ is the same as $k[0, 1] = [0, k]$. The advantage of the Johnson S_b is that it can be shaped like log-normal distribution, but with thinner tails below the bound.

For any distribution, all the information about the unobserved spatial heterogeneity is captured by the spread or standard deviation parameter. For example, a significant standard deviation would reveal a spatially non-stationary relationship, and the higher the standard deviation the higher the unobserved spatial heterogeneity in the parameters. Finally, it is worth noting that if only the constant is assumed to be random, then the model is reduced to the random effect model also known as the spatially constant random parameter in the multilevel context (Jones 1991). If $n_c = 1$ for all C, then the model is reduced to the RCM.

2.3 Correlated spatially random parameters and observed variations around the mean

The random parameters can be generalized to include correlation across the parameters. For example, we may be interested in whether regions with greater (lower) β_1 have also greater (lower) values for β_2. If it is true, we would say that both effects are positively correlated within regions. Furthermore, it is likely that the association between y_{ci}^* and x_{ci} is modified by unmeasured regional effects or region-specific unobserved factors. Therefore, by allowing the constant and the slope parameter to be correlated we might be able to identify whether those unobserved factors and the effect of x_{ci} are positively or negatively associated.

[5] If some coefficient is expected a priori to be negative for all the regions, one might create the negative of the variable and then include this new variable in the estimation. This "trick" allows the coefficient to be negative without imposing a sign change in the estimation procedure (Train 2009).

As an illustration of the usefulness of the correlated parameters, Wheeler, Tiefelsdorf (2005) raise the awareness of the potential dependencies (correlation) among the local regression coefficients associated with different exogenous variables in the GWR context. They use a GWR approach to explain the white male bladder cancer mortality rates in the 508 States Economic Areas of the United States. Using the population density and smoking as covariates, they find that those regions with high smoking parameter also have a low population density parameter. As they state, the important question is whether this negative correlation is real or an artifact of the statistical method. By allowing the parameters to be explicitly correlated, we are able to test whether the correlation among the parameters is in fact significant[6].

For simplicity of the notation, consider that the coefficients are distributed across space following a multivariate normal distribution, $\boldsymbol{\beta}_c \sim \text{MVN}(\boldsymbol{\beta}, \boldsymbol{\Sigma})$. In this case, the coefficient can be written as:

$$\boldsymbol{\beta}_c = \boldsymbol{\beta} + \boldsymbol{L}\boldsymbol{\eta}_c,$$

where $\boldsymbol{\eta}_c \sim N(\mathbf{0}, \boldsymbol{I})$, and $\boldsymbol{L}$ is the lower-triangular Cholesky factor of $\boldsymbol{\Sigma}$ such that $\boldsymbol{L}\boldsymbol{L}' = \text{var}(\boldsymbol{\beta}_c) = \boldsymbol{\Sigma}$. When the off-diagonal elements of $\boldsymbol{L}$ are zero, the parameters are independently normal distributed. If we assume that the model has only one covariate and the constant, then the extended form of the spatially random coefficient vector is

$$\begin{aligned} \begin{pmatrix} \alpha_c \\ \beta_c \end{pmatrix} &= \begin{pmatrix} \alpha \\ \beta \end{pmatrix} + \begin{pmatrix} \sigma_{\alpha\alpha} & 0 \\ \sigma_{\beta\alpha} & \sigma_{\beta\beta} \end{pmatrix} \begin{pmatrix} \eta_{c\alpha} \\ \eta_{c\beta} \end{pmatrix} \\ \boldsymbol{\beta}_c &= \boldsymbol{\beta} + \boldsymbol{L}\boldsymbol{\eta}_c, \end{aligned}$$

where:

$$\boldsymbol{L}\boldsymbol{L}' = \begin{pmatrix} \sigma_{\alpha\alpha} & 0 \\ \sigma_{\beta\alpha} & \sigma_{\beta\beta} \end{pmatrix} \begin{pmatrix} \sigma_{\alpha\alpha} & \sigma_{\beta\alpha} \\ 0 & \sigma_{\beta\beta} \end{pmatrix} = \begin{pmatrix} \sigma_{\alpha\alpha}^2 & \sigma_{\alpha\alpha}\sigma_{\beta\alpha} \\ \sigma_{\beta\alpha}\sigma_{\alpha\alpha} & \sigma_{\beta\alpha}^2 + \sigma_{\beta\beta}^2 \end{pmatrix} = \boldsymbol{\Sigma}$$

If we need correlated parameters with positive domain, we might create a log-normal distributed parameter. For instance, if we need β_c to be log-normal distributed, then we can transform it in the following way:

$$\beta_c = \exp(\beta + \sigma_{\beta\alpha}\eta_{c\alpha} + \sigma_{\beta\beta}\eta_{c\beta})$$

Observed spatial heterogeneity – or deterministic spatial heterogeneity – can be also accommodated in the random parameters by including region-specific covariates. Specifically, the vector of random coefficient is:

$$\boldsymbol{\beta}_c = \boldsymbol{\beta} + \boldsymbol{\pi}\boldsymbol{z}_c + \boldsymbol{L}\boldsymbol{\eta}_c \tag{2}$$

where $\boldsymbol{z}_c$ is a set of M characteristics of region c that influences the mean of the spatial random coefficients, and $\boldsymbol{\Pi}$ is a $K \times M$ matrix of additional parameters. The conditional mean becomes $E(\boldsymbol{\beta}_c|\boldsymbol{z}_c) = \boldsymbol{\beta} + \boldsymbol{\Pi}\boldsymbol{z}_c$. The main drawback of this modeling strategy – and any type of spatial heterogeneity in the form of unobserved spatial heterogeneity – is that it assumes that the coefficients are drawn from some univariate or multivariate distribution and no attention is paid to the location of the regions (Fotheringham, Brunsdon 1999). However, the previous model can be very useful to consider regions' location explicitly in the random parameters if $\boldsymbol{z}_c$ includes any function of the geographical coordinates $(\boldsymbol{u}_c, \boldsymbol{v}_c)$. Thus, if $\boldsymbol{z}_c = h(\boldsymbol{u}_c, \boldsymbol{v}_c)$, where $h()$ is any function, and $\boldsymbol{\eta}_c = \mathbf{0}$, then the model collapses into the Casetti's spatial expansion method.

[6] Those readers interested in modelling both spatial dependence and spatial heterogeneity are referred to Dong et al. (2016). They develop a spatial random slope multilevel modeling approach to account for the within-group dependence among individuals in the same area and the spatial dependence between areas simultaneously.

2.4 Estimation

Let $\boldsymbol{y}_c = \{y_{i1}, y_{i2}, \ldots, y_{in}\}$ be the sequence of choices for all individuals in region c, where n_c is the total number of individuals in that region. Assuming that individuals are independent across regions, the joint probability density function, given $\boldsymbol{\beta}_c$, can be written as

$$Pr(\boldsymbol{y}_c|\boldsymbol{X}_c, \boldsymbol{\beta}_c) = \sum_{i=1}^{n_c} f^*(y_{ic}|\boldsymbol{x}_{ic}, \boldsymbol{\beta}_c), \tag{3}$$

because, conditional on $\boldsymbol{\beta}_c$, the observations are independent. Since $\boldsymbol{\beta}_c$ is common for individuals living in the region c, within each region individuals are not independent. Thus, the unconditional pdf of $\boldsymbol{y}_c$ given $\boldsymbol{X}_c$ will be the weighted average of the conditional probability evaluated over all possible values of $\boldsymbol{\beta}$, which depends on the parameters of the distribution of $\boldsymbol{\beta}_c$:

$$P_c(\boldsymbol{\theta}) = f(\boldsymbol{y}_c|\boldsymbol{X}_c, \boldsymbol{\theta}) = \int_{\boldsymbol{\beta}_c} \left[\prod_{i=1}^{N_c} f^*(y_{ic}|\boldsymbol{x}_{ic}, \boldsymbol{\beta}_c, \boldsymbol{\theta}) \right] g(\boldsymbol{\beta}_c) d\boldsymbol{\beta}_c, \tag{4}$$

The unconditional probability has no closed form solution, therefore the log-likelihood function is difficult to compute. However, we can simulate this probability and use the simulated maximum likelihood in order to estimate $\boldsymbol{\theta}$ (Gourieroux, Monfort 1997, Hajivassiliou, Ruud 1986, Stern 1997, Train 2009)[7]. In particular, $P_c(\boldsymbol{\theta})$ is approximated by a summation over randomly chosen values of $\boldsymbol{\beta}_c$. For a given value of the parameters $\boldsymbol{\theta}$, a value of $\boldsymbol{\beta}_c$ is drawn from its distribution. Using this draw of $\boldsymbol{\beta}_c$, $P_c(\boldsymbol{\theta})$ is calculated. This process is repeated for many draws, and the average over the draws is the simulated probability. Formally, the simulated probability for region c is

$$\tilde{P}_c(\boldsymbol{\theta}) = \frac{1}{R} \sum_{r=1}^{R} \prod_{i=1}^{N_c} \tilde{P}_{icr}(\boldsymbol{\theta}) \tag{5}$$

where $\tilde{P}_{icr}$ is the probability for individual i in region c evaluated at the r^{th} draw of $\boldsymbol{\beta}_c$, and R is the total number of draws. Then, the simulated log-likelihood function is:

$$\log L_s = \sum_{c=1}^{C} \log \left[\frac{1}{R} \sum_{r=1}^{R} \prod_{i=1}^{N_c} \tilde{P}_{icr}(\boldsymbol{\theta}) \right] \tag{6}$$

Lee (1992), Gourieroux, Monfort (1991) and Hajivassiliou, Ruud (1986) derive the asymptotic distribution of the simulated maximum likelihood (SML) estimator based on smooth probability simulators with the number of draws increasing with sample size. Under regularity conditions, the estimator is consistent and asymptotically normal. When the number of draws, R, rises faster than the square root of the number of observations, the estimator is asymptotically equivalent to the maximum likelihood estimator. It is worth noting that, even though the simulated probability is an unbiased estimate of the true probability, the log of the simulated probability with fixed number of repetitions is not an unbiased estimate of the log of the true probability. This bias in the SML decreases as the number of draws increases (see for example Gourieroux, Monfort 1997, Revelt, Train 1998).

One main limitation of these modeling strategies is that the performance of the maximum likelihood estimators may not be accurate or satisfactory when the number of individuals per region is large. The problem is that the log-likelihood function involves the integration or summation over a term involving the product of the probabilities for all the individuals in each location c. Borjas, Sueyoshi (1994) were the first in noticing this problem in the context of the probit model with random effects and using Gauss quadrature. Lee (2000) also gives more insights about this problem. For example,

[7] Other methods can be used in order to approximate the integrals. For example, Gauss-Hermite quadrature procedure is another numerical method widely used. However, it has been documented that for models with more than 3 random parameters SML performs better. Bayesian estimation is also suitable for continuous spatial heterogeneity. See for example Hashiguchi, Tanaka (2014).

assuming a sample of 500 individuals per group – or regions in our case – with a likelihood contribution of 0.5 per observation, Borjas, Sueyoshi (1994) show that the value of the integrand can be as small as $\exp(500 \times \ln(0.5)) \approx \exp(-346.6)$, which is below the existing absolute value for a computer. A consequence of this might be larger standard errors, explosive estimates and/or a singular Hessian. In the worst scenario, the computation will overflow, that is, it will exceed the computer's capacity to compute the value and the maximization procedure will stop. This issue should be borne in mind when applying these methods[8].

3 Packages and dependencies

The main R packages used in the examples are the following:

Rchoice: This is the main package to estimate Binary, Poisson, and Ordered Models with Random parameters.

foreign: This package is used to read data in different formats (Stata, SPSS, etc).

car: This package will allow us to perform linear hypotheses.

lmtest: This package has generic functions that allow to perform likelihood ratio tests for nested models.

All these packages can be installed using the `install.packages()` function.

4 Application using Rchoice in R: Self-assessed health status

4.1 Ordered Probit model with spatially homogeneous parameters

Suppose we are interested in the determinants of individuals' subjective evaluation of health. We assume that the health status of individual i in municipality c, h_{ic}, follows an underlying continuous but latent health process h^*_{ic} based on a linear combination of individual and municipal covariates given by:

$$h^*_{ic} = \boldsymbol{x}'_{ic}\boldsymbol{\beta} + \epsilon_{ic} \tag{7}$$

where $\boldsymbol{x}_{ic}$ is a vector of individual and municipal characteristics; $\epsilon_{ic} \sim N(0, \sigma)$ is the error term, but since the scale of h^*_{ic} is not identified, we normalized $\sigma = 1$. Note that this model assumes that the partial correlation between the latent health status and the covariates follows a spatially stationary process.

As typical in ordered models, we do not observe h^*_{ci}, but we instead observe the self-assessed health status (SAH) for each individual, h_{ic}, which ranges between 1 (very bad health) and 5 (very good health) in our sample. The link between h_{ic} and h^*_{ic} is the following:

$$h_{ic} = \begin{cases} 1 & \text{if } \kappa_0 < h^*_{ic} < \kappa_1 \\ 2 & \text{if } \kappa_1 < h^*_{ic} < \kappa_2 \\ \vdots & \\ 5 & \text{if } \kappa_4 < h^*_{ic} < \kappa_5 \end{cases}$$

where it is assumed that $\kappa_0 = -\infty$ and $\kappa_5 = \infty$ to cover the entire real line. Since having a constant is useful in our model to accommodate random effects, we set $x_{1ic} \equiv 1$ for all $i = 1, \ldots, N$. Therefore, for identification we set $\kappa_1 \equiv 0$.

To estimate an ordered probit model with spatially homogeneous coefficients, we will use the `Rchoice` package which is loaded using the `library()` function:

[1]:
```
> # Load package
> library("Rchoice")
```

[8] For other estimation methods, such as Bayesian estimation of multi-level models, see for example Bürkner (2018).

Now, we load the dataset `sah.chile`. This dataset comes from the 2013 National Socioeconomic Characteri-zation Survey (CASEN) from Chile. CASEN is a national, population-based survey which is representative at the municipal level and is carried out by the Ministry of Planning (MIDEPLAN) to describe the socioeconomic situation as well as the impact of social programs on the living conditions of the Chilean Population[9].

In the following lines, the dataset in Stata format is downloaded. Then, the SAH variable (dependent variable) is recoded into 5 categories:

[2]:
```
> # Load data set and recode SAH variable
> library("foreign") # package to load datasets
> library("car") # package with recode function
> data <- read.dta("https://msarrias.weebly.com/uploads/3/7/7/8/37783629/sah.chile.dta")
> data$sah2 <- recode(data$shealth, "1= 1; 2 = 2; 3 = 3; 5:6 = 4; 7 = 5")
```

The vector $\boldsymbol{x}_{ic}$ includes the following controls at the individual level:

linch: log of household income.

agen: age in years / 10.

hsizen: household size / 10.

edun: years of schooling / 10.

male: =1 for men.

dcivil1: =1 if the individual is married.

dlstatus2: =1 if the individual is unemployed.

Some continuous variables are divided by 10 to improve convergence speed of the SML process and avoid singularities in the Hessian matrix.

In addition, a set of dummy variables indicating the self-perception of pollution and environmental problems is used. The dummy variables are obtained from the response to the question: *"What problems related to pollution and environmental degradation do you identify in your neighbourhood or location"*. Based on the answer, dummy variables were created for the following problems:

noise: noise pollution.

airpol: air pollution.

watpol: water pollution.

vispol: visual pollution.

waspol: garbage (rubbish) in the neighborhood.

The variables at the municipality level are:

lmdinc: log of median income (proxy for development).

lpop: log of population (size effect).

The following command lines show how to estimate the traditional ordered probit model. For other models such as the Binary (Logit and Probit) and Poisson model see Sarrias (2016)Sarrias (2016).

[3]:
```
> # Ordered probit model
> oprobit <- Rchoice(sah2 ~ linch + agen + hsizen + edun + male +
+   dcivil1 + dlstatus2 +
+        noise + airpol + watpol + vispol + waspol +
+        lmdinc + lpop,
+        family = ordinal("probit"),
+        data = data)
> summary(oprobit)
```

[9] Chile has 346 municipalities of which 324 are representative in CASEN 2013.

[3]:
```
##
## Model: ordinal
## Model estimated on: Tue Jan 07 10:31:58 2020
##
## Call:
## Rchoice(formula = sah2 ~ linch + agen + hsizen + edun + male +
## dcivil1 + dlstatus2 + noise + airpol + watpol + vispol +
## waspol + lmdinc + lpop, data = data, family = ordinal("probit"),
## method = "bfgs")
##
##
## Frequencies of categories:
## y
##       1       2       3       4       5
## 0.01087 0.01359 0.02897 0.64523 0.30133
## The estimation took: 0h:0m:5s
##
## Coefficients:
##             Estimate Std. Error z-value Pr(>|z|)
## kappa.1     0.341698   0.022434  15.231  < 2e-16 ***
## kappa.2     0.720961   0.027012  26.690  < 2e-16 ***
## kappa.3     3.015915   0.031564  95.548  < 2e-16 ***
## constant  -0.290407   0.481404  -0.603 0.546342
## linch       0.132595   0.014377   9.223  < 2e-16 ***
## agen       -0.212654   0.008214 -25.889  < 2e-16 ***
## hsizen      0.242645   0.060359   4.020 5.82e-05 ***
## edun        0.211808   0.028537   7.422 1.15e-13 ***
## male        0.154095   0.019058   8.086 6.66e-16 ***
## dcivil1    -0.028142   0.021246  -1.325 0.185316
## dlstatus2 -0.093728   0.046169  -2.030 0.042347 *
## noise      -0.139891   0.026745  -5.231 1.69e-07 ***
## airpol     -0.084563   0.026219  -3.225 0.001259 **
## watpol     -0.120485   0.037756  -3.191 0.001417 **
## vispol     -0.069064   0.060747  -1.137 0.255573
## waspol     -0.041040   0.027421  -1.497 0.134482
## lmdinc      0.147852   0.043400   3.407 0.000658 ***
## lpop       -0.016493   0.009016  -1.829 0.067359 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Optimization of log-likelihood by BFGS maximization
## Log Likelihood: -13020
## Number of observations: 16188
## Number of iterations: 190
## Exit of MLE: successful convergence
```

The argument `family = ordinal("probit")` indicates that an ordered probit model will be estimated. If the user wants an ordered logit model the argument should be `family = ordinal("logit")`. For other models, see `help(Rchoice)`.

The results show that household income and education increase the probability of reporting better health status, whereas age decreases it. Men are more likely to report better health than women and being unemployed is detrimental for health. At the neighborhood level, noise, air, and water pollution reduce health perception and vispol and waspol apparently do not matter for health. The coefficient for the logarithm of population for each municipality, which is intended to capture agglomeration effects, is negative but weakly significant, whereas the level of development is positively correlated with individuals' health evaluation.

4.2 Ordered Probit models with spatial random coefficients

The standard ordered probit model does not allow for spatial heterogeneity in the coefficients. In this section, we estimate an Ordered Probit with Spatial Random Parameters (OPSRP) model. To reduce excessive computing time, we will only assume that the variables at the level of municipalities and neighborhood vary across space.

The first and more difficult task is to choose the distribution for each of them. As explained by Hensher, Greene (2003), distributions are essentially arbitrary approximations to the real behavioral profile. The researcher chooses a specific distribution because he has

a sense that the "empirical truth" is somewhere in their domain. The most widely used distribution in the empirical literature is the normal distribution due to its properties. If unobserved spatial heterogeneity is viewed as the sum of small random influences, then the central limit theorem can be invoked to justify the normality assumption Greene, Hensher (2010). Moreover, the normal distribution is unbounded, and therefore every real number has a positive probability of being drawn. Thus, specifying a given coefficient to follow a normal distribution is equivalent to making a priori assumption that both positive and negative coefficients exits across space (Sarrias 2019).

This last property is very appealing in our case, since theoretically we might observe municipalities with positive and negative sign for the population coefficient. For instance, municipalities with a positive coefficient might be characterized for having positive urban externalities that outweigh the negative ones. In those municipalities, inhabitants, on average, enjoy better health through local positive urban externalities. If the coefficient is negative, the opposite might be expected.

A OPSRP model with normally distributed parameters is estimated as follows:

[4]:
```
> # Spatial random parameter model
> ran_1 <- Rchoice(sah2 ~ linch + agen + hsizen + edun + male + dcivil1 + dlstatus2 +
+        noise + airpol + watpol + vispol + waspol +
+        lmdinc + lpop,
+        family = ordinal('probit'),
+        data = data,
+        ranp = c(noise = "n", airpol = "n", watpol = "n", vispol = "n",
+        waspol = "n", lmdinc = "n", lpop = "n"),
+        panel = TRUE,
+        index = "idc",
+        R=30,
+        method = "bfgs")
> summary(ran_1)
```

[4]:
```
##
## Model: ordinal
## Model estimated on: Tue Dec 31 09:11:29 2019
##
## Call:
## Rchoice(formula = sah2 ~ linch + agen + hsizen + edun + male +
## dcivil1 + dlstatus2 + noise + airpol + watpol + vispol +
## waspol + lmdinc + lpop, data = data, family = ordinal("probit"),
## ranp = c(noise = "n", airpol = "n", watpol = "n", vispol = "n",
## waspol = "n", lmdinc = "n", lpop = "n"), R = 30, panel = TRUE,
## index = "idc", method = "bfgs", iterlim = 2000)
##
##
## Frequencies of categories:
## y
##       1       2       3       4       5
## 0.01087 0.01359 0.02897 0.64523 0.30133
## The estimation took: 0h:9m:48s
##
## Coefficients:
##               Estimate Std. Error z-value Pr(>|z|)
## kappa.1      0.3417117  0.0162495  21.029  < 2e-16 ***
## kappa.2      0.7209893  0.0209411  34.429  < 2e-16 ***
## kappa.3      3.0170292  0.0268859 112.216  < 2e-16 ***
## constant    -0.2905344  0.4958543  -0.586 0.557925
## linch        0.1310569  0.0143865   9.110  < 2e-16 ***
## agen        -0.2132221  0.0082107 -25.969  < 2e-16 ***
## hsizen       0.2425766  0.0604009   4.016 5.92e-05 ***
## edun         0.2116421  0.0285525   7.412 1.24e-13 ***
## male         0.1540426  0.0190620   8.081 6.66e-16 ***
## dcivil1     -0.0281834  0.0212486  -1.326 0.184717
## dlstatus2   -0.0937187  0.0462089  -2.028 0.042545 *
## mean.noise  -0.1399619  0.0269286  -5.198 2.02e-07 ***
## mean.airpol -0.0845976  0.0264687  -3.196 0.001393 **
## mean.watpol -0.1205199  0.0379779  -3.173 0.001507 **
## mean.vispol -0.0690556  0.0611251  -1.130 0.258585
## mean.waspol -0.0410117  0.0276460  -1.483 0.137953
## mean.lmdinc  0.1463353  0.0445929   3.282 0.001032 **
## mean.lpop   -0.0180293  0.0091612  -1.968 0.049067 *
```

```
## sd.noise     0.0986997  0.0261224   3.778 0.000158 ***
## sd.airpol    0.0986787  0.0254373   3.879 0.000105 ***
## sd.watpol    0.0994208  0.0398379   2.496 0.012573 *
## sd.vispol    0.0998332  0.0688382   1.450 0.146987
## sd.waspol    0.0988309  0.0273009   3.620 0.000295 ***
## sd.lmdinc    0.0050337  0.0008714   5.776 7.64e-09 ***
## sd.lpop      0.0012177  0.0011066   1.100 0.271156
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Optimization of log-likelihood by BFGS maximization
## Log Likelihood: -11320
## Number of observations: 16188
## Number of iterations: 391
## Exit of MLE: successful convergence
## Simulation based on 30 Halton draws
```

The argument `ranp` indicates which variables are random in the formula and their distributions. In this example, all the random variables are assumed to be normally distributed using `"n"`. The remaining distribution discussed in Section 2.2 can be used using the following shorthands:

- Triangular = `"t"`,
- Uniform = `"u"`,
- Truncated normal = `"cn"`,
- Log-normal = `"ln"`,
- Johnson's Sb = = `"sb"`.

The number of draws for the simulation of the probability is set using the argument `R`. To keep the estimation time manageable, we use 30 draws for each individual. However, consistency of the SML requires a higher number of draws (see for example Train 2009).

The argument `index` is a string indicating the id for the municipalities in the data, whereas `panel=TRUE` allows for the spatial structure of the sample.

The previous model assumes that the coefficients has the following form:

$$\beta_k = \bar{\beta}_k + \sigma_k v_{ir}$$

where $v_{ir} \sim N(01)$. Thus, the coefficients with the `mean.` and `sd.` prefix represent the estimated mean, $\hat{\bar{\beta}}$, and standard deviation, $\hat{\sigma}$, for variable k, respectively. If $\sigma_k = 0$, then there is no evidence of systematical variation for regression coefficient over space. The output shows that there is evidence of spatial heterogeneity for most of the variables, except for `vispol` and `lpop`.

To test the joint hypothesis of coefficient homogeneity across space we can perform a Likelihood Ratio test using `lrtest` function from `lmtest` package.

[5]:
```
> # Testing spatial heterogeneity
> library("lmtest")
> lrtest(oprobit, ran_1)
```

[5]:
```
## Likelihood ratio test
##
## Model 1: sah2 ~ linch + agen + hsizen + edun + male + dcivil1 + dlstatus2 +
## noise + airpol + watpol + vispol + waspol + lmdinc + lpop
## Model 2: sah2 ~ linch + agen + hsizen + edun + male + dcivil1 + dlstatus2 +
## noise + airpol + watpol + vispol + waspol + lmdinc + lpop
## #Df LogLik Df  Chisq Pr(>Chisq)
## 1  18 -13018
## 2  25 -11318  7 3400.6  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The test rejects the null hypothesis providing empirical evidence of spatial heterogeneity for those variables.

Since the parameters are allowed to vary across space following a normal distribution, we can also compute the proportion of municipalities with positive coefficients using $\Phi(\hat{\beta}/\hat{\sigma})$. For example, for `noise` and `lmdinc` the results are:

[6]:
```
> # Computing proportions
> pnorm(coef(ran_1)["mean.noise"] / coef(ran_1)["sd.noise"])
```

[6]:
```
## mean.noise
## 0.07808686
```

[7]:
```
> pnorm(coef(ran_1)["mean.lmdinc"] / coef(ran_1)["sd.lmdinc"])
```

[7]:
```
## mean.lmdinc
##           1
```

Thus, we can say that for 100% of the municipalities development is positively correlated with individuals' health, whereas for around 8% of the municipalities, higher noise pollution increases health. This last result can be true or an artifact of the normality assumption.

4.3 Correlated parameters

The previous model specifies the coefficients to be independently distributed, while one would expect correlation. To show this, the model `ran_1` we will be estimated but assuming that the spatially random coefficients are correlated adding the argument `correlation = TRUE`:

[8]:
```
> # Spatially random parameters with correlated coefficients
> ran_2 <- Rchoice(sah2 ~ linch + agen + hsizen + edun + male + dcivil1 +
+        dlstatus2 + noise + airpol + watpol + vispol + waspol +
+        lmdinc + lpop,
+        family = ordinal("probit"),
+        data = data,
+        ranp = c(noise = "n", airpol = "n", watpol = "n", vispol = "n",
+        waspol = "n", lmdinc = "n", lpop = "n"),
+        panel = TRUE,
+        index = "idc",
+        R=30,
+        method = "bfgs",
+        correlation = TRUE)
> summary(ran_2)
```

[8]:
```
##
## Model: ordinal
## Model estimated on: Tue Dec 31 09:21:06 2019
##
## Call:
## Rchoice(formula = sah2 ~ linch + agen + hsizen + edun + male +
## dcivil1 + dlstatus2 + noise + airpol + watpol + vispol +
## waspol + lmdinc + lpop, data = data, family = ordinal("probit"),
## ranp = c(noise = "n", airpol = "n", watpol = "n", vispol = "n",
## waspol = "n", lmdinc = "n", lpop = "n"), R = 30, correlation = TRUE,
## panel = TRUE, index = "idc", method = "bfgs", iterlim = 2000)
##
##
## Frequencies of categories:
## y
##       1       2       3       4       5
## 0.01087 0.01359 0.02897 0.64523 0.30133
## The estimation took: 0h:9m:36s
##
## Coefficients:
##                    Estimate Std. Error z-value Pr(>|z|)
## kappa.1           0.3412626  0.0180952  18.859  < 2e-16 ***
## kappa.2           0.7195420  0.0226625  31.750  < 2e-16 ***
## kappa.3           3.4970489  0.0294076 118.917  < 2e-16 ***
## constant         -0.2901421  0.5322664  -0.545 0.585680
## linch             0.1457106  0.0149581   9.741  < 2e-16 ***
## agen             -0.2696671  0.0086621 -31.132  < 2e-16 ***
## hsizen            0.2440816  0.0629962   3.875 0.000107 ***
## edun              0.2202293  0.0300012   7.341 2.12e-13 ***
```

```
## male              0.1579018  0.0199386   7.919 2.44e-15 ***
## dcivil1          -0.0361313  0.0222373  -1.625 0.104204
## dlstatus2        -0.0930049  0.0482324  -1.928 0.053822 .
## mean.noise       -0.1446350  0.0281478  -5.138 2.77e-07 ***
## mean.airpol      -0.0855360  0.0275906  -3.100 0.001934 **
## mean.watpol      -0.1204861  0.0399076  -3.019 0.002535 **
## mean.vispol      -0.0689847  0.0636902  -1.083 0.278751
## mean.waspol      -0.0371335  0.0288215  -1.288 0.197609
## mean.lmdinc       0.1607965  0.0478379   3.361 0.000776 ***
## mean.lpop         0.0069806  0.0101792   0.686 0.492860
## sd.noise.noise    0.0050271  0.0333140   0.151 0.880055
## sd.noise.airpol   0.0058448  0.0324393   0.180 0.857013
## sd.noise.watpol   0.0539982  0.0478928   1.127 0.259539
## sd.noise.vispol   0.0739318  0.0776348   0.952 0.340943
## sd.noise.waspol   0.0182864  0.0331744   0.551 0.581483
## sd.noise.lmdinc   0.0618425  0.0086254   7.170 7.51e-13 ***
## sd.noise.lpop    -0.0813180  0.0105174  -7.732 1.07e-14 ***
## sd.airpol.airpol  0.0160025  0.0319875   0.500 0.616881
## sd.airpol.watpol  0.0559519  0.0459907   1.217 0.223760
## sd.airpol.vispol  0.0816862  0.0775592   1.053 0.292245
## sd.airpol.waspol  0.0208368  0.0330971   0.630 0.528979
## sd.airpol.lmdinc  0.0434195  0.0084910   5.114 3.16e-07 ***
## sd.airpol.lpop   -0.0588680  0.0102640  -5.735 9.73e-09 ***
## sd.watpol.watpol  0.0628180  0.0456531   1.376 0.168826
## sd.watpol.vispol  0.0818528  0.0779919   1.050 0.293946
## sd.watpol.waspol  0.0233655  0.0330245   0.708 0.479244
## sd.watpol.lmdinc  0.0408594  0.0085428   4.783 1.73e-06 ***
## sd.watpol.lpop   -0.0505877  0.0103099  -4.907 9.26e-07 ***
## sd.vispol.vispol  0.0850337  0.0779709   1.091 0.275457
## sd.vispol.waspol  0.0187284  0.0330837   0.566 0.571331
## sd.vispol.lmdinc  0.0471032  0.0083197   5.662 1.50e-08 ***
## sd.vispol.lpop   -0.0644627  0.0100842  -6.392 1.63e-10 ***
## sd.waspol.waspol  0.0224287  0.0329135   0.681 0.495591
## sd.waspol.lmdinc  0.0491523  0.0082301   5.972 2.34e-09 ***
## sd.waspol.lpop   -0.0642418  0.0099655  -6.446 1.15e-10 ***
## sd.lmdinc.lmdinc  0.0353881  0.0082862   4.271 1.95e-05 ***
## sd.lmdinc.lpop   -0.0450126  0.0099427  -4.527 5.98e-06 ***
## sd.lpop.lpop      0.0002568  0.0011373   0.226 0.821375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Optimization of log-likelihood by BFGS maximization
## Log Likelihood: -11210
## Number of observations: 16188
## Number of iterations: 315
## Exit of MLE: successful convergence
## Simulation based on 30 Halton draws
```

It is important to note that the output prints the elements of the lower-triangular Cholesky factor $\boldsymbol{L}$. The variance-covariance matrix, $\boldsymbol{\Sigma}$, can be extracted using the `vcov` function in the following way:

[9]:
```
> # Obtain Sigma
> vcov(ran_2, what = "ranp", type = "cov", se = TRUE)
```

[9]:
```
##
## Elements of the variance-covariance matrix
##
##                    Estimate Std. Error z-value  Pr(>|z|)
## v.noise.noise    2.5271e-05 3.3494e-04  0.0754   0.93986
## v.noise.airpol   2.9382e-05 2.3810e-04  0.1234   0.90179
## v.noise.watpol   2.7145e-04 1.7934e-03  0.1514   0.87969
## v.noise.vispol   3.7166e-04 2.4696e-03  0.1505   0.88037
## v.noise.waspol   9.1927e-05 6.3478e-04  0.1448   0.88486
## v.noise.lmdinc   3.1089e-04 2.0673e-03  0.1504   0.88046
## v.noise.lpop    -4.0879e-04 2.7200e-03 -0.1503   0.88053
## v.airpol.airpol  2.9024e-04 1.0858e-03  0.2673   0.78923
## v.airpol.watpol  1.2110e-03 2.4984e-03  0.4847   0.62788
## v.airpol.vispol  1.7393e-03 3.7015e-03  0.4699   0.63843
## v.airpol.waspol  4.4032e-04 1.0616e-03  0.4148   0.67829
## v.airpol.lmdinc  1.0563e-03 2.4234e-03  0.4359   0.66293
## v.airpol.lpop   -1.4173e-03 3.2248e-03 -0.4395   0.66029
```

```
## v.watpol.watpol  9.9925e-03 9.2513e-03  1.0801   0.28009
## v.watpol.vispol  1.3705e-02 9.7826e-03  1.4009   0.16124
## v.watpol.waspol  3.6211e-03 3.6712e-03  0.9863   0.32397
## v.watpol.lmdinc  8.3355e-03 4.0026e-03  2.0825   0.03730 *
## v.watpol.lpop   -1.0863e-02 5.2619e-03 -2.0644   0.03898 *
## v.vispol.vispol  2.6069e-02 2.4122e-02  1.0807   0.27982
## v.vispol.waspol  6.5591e-03 5.8208e-03  1.1268   0.25981
## v.vispol.lmdinc  1.5469e-02 7.5381e-03  2.0521   0.04016 *
## v.vispol.lpop   -2.0443e-02 9.9442e-03 -2.0558   0.03981 *
## v.waspol.waspol  2.1683e-03 2.9746e-03  0.7289   0.46604
## v.waspol.lmdinc  4.9749e-03 3.5556e-03  1.3992   0.16176
## v.waspol.lpop   -6.5438e-03 4.7027e-03 -1.3915   0.16407
## v.lmdinc.lmdinc  1.3266e-02 1.8869e-03  7.0309 2.052e-12 ***
## v.lmdinc.lpop   -1.7439e-02 2.3778e-03 -7.3341 2.232e-13 ***
## v.lpop.lpop      2.2946e-02 3.0048e-03  7.6365 2.243e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated coefficients represent the variance and covariance of the randomly distributed parameters. Their standard errors are estimated using the Delta Method. To obtain the standard deviations for the random parameters, one might use the following code:

[10]:
```
> # Obtain standard deviations
> vcov(ran_2, what = "ranp", type = "sd", se = TRUE)
```

[10]:
```
##
## Standard deviations of the random parameters
##
## Estimate Std. Error z-value Pr(>|z|)
## noise  0.0050271  0.0333140  0.1509  0.88006
## airpol 0.0170365  0.0318660  0.5346  0.59291
## watpol 0.0999627  0.0462737  2.1602  0.03075 *
## vispol 0.1614594  0.0746999  2.1614  0.03066 *
## waspol 0.0465651  0.0319403  1.4579  0.14487
## lmdinc 0.1151790  0.0081909 14.0617  < 2e-16 ***
## lpop        0.1514789        0.0099181 15.2730        < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, the correlation matrix for the estimated coefficients is:

[11]:
```
> # Obtain correlation matrix of estimated coefficients
> vcov(ran_2, what = "ranp", type = "cor")
```

[11]:
```
##             noise     airpol     watpol     vispol     waspol     lmdinc
## noise   1.0000000  0.3430771  0.5401842  0.4578971  0.3927053  0.5369247
## airpol  0.3430771  1.0000000  0.7110814  0.6323119  0.5550469  0.5383007
## watpol  0.5401842  0.7110814  1.0000000  0.8491072  0.7779253  0.7239690
## vispol  0.4578971  0.6323119  0.8491072  1.0000000  0.8724090  0.8317965
## waspol  0.3927053  0.5550469  0.7779253  0.8724090  1.0000000  0.9275751
## lmdinc  0.5369247  0.5383007  0.7239690  0.8317965  0.9275751  1.0000000
## lpop   -0.5368276 -0.5492089 -0.7173734 -0.8358492 -0.9277189 -0.9995225

##              lpop
## noise  -0.5368276
## airpol -0.5492089
## watpol -0.7173734
## vispol -0.8358492
## waspol -0.9277189
## lmdinc -0.9995225
## lpop    1.0000000
```

The results show, for example, that noise pollution is positively correlated with other forms of pollution. Therefore, in those municipalities where noise pollution is detrimental to health, so are the other forms of pollution. It is also important to note that the municipalities where noise has a negative effect are also municipalities where lower development and higher population impact negatively the self-perception of health status.

4.4 Region-specific coefficients

In the applied literature it is very common to map the region-specific estimates to display the spatial heterogeneity for certain coefficients. This cannot be done using just the distribution of the parameters across regions, $g(\boldsymbol{\beta}_c|\boldsymbol{\theta})$. The population distributions give us just the average affect, $\boldsymbol{\beta}$, and the spatial variation around this mean, $\sigma_{\boldsymbol{\beta}}$, when in fact we would like to know where each region's $\boldsymbol{\beta}_c$ lies in $g(\boldsymbol{\beta}_c|\boldsymbol{\theta})$. We might be able to find the likely location of a given region on the heterogeneity distribution by moving from the conditional to the unconditional distribution (Revelt, Train 2000, Brunsdon et al. 1999, Sarrias 2019). Using Bayes' theorem, we obtain:

$$f(\boldsymbol{\beta}_c|\boldsymbol{y}_c,\boldsymbol{X}_c,\boldsymbol{\theta}) = \frac{f(\boldsymbol{y}_c|\boldsymbol{X}_c,\boldsymbol{\beta}_c)g(\boldsymbol{\beta}_c|\boldsymbol{\theta})}{f(\boldsymbol{y}_c|\boldsymbol{X}_c,\boldsymbol{\theta})} = \frac{f(\boldsymbol{y}_c|\boldsymbol{X}_c,\boldsymbol{\beta}_c)g(\boldsymbol{\beta}_c|\boldsymbol{\theta})}{\int_{\boldsymbol{\beta}_c} f(\boldsymbol{y}_c|\boldsymbol{X}_c,\boldsymbol{\beta}_c)g(\boldsymbol{\beta}_c|\boldsymbol{\theta})d\boldsymbol{\beta}_c} \tag{8}$$

where $f(\boldsymbol{\beta}_c|\boldsymbol{y}_c,\boldsymbol{X}_c,\boldsymbol{\theta})$ is the distribution of the regional parameters $\boldsymbol{\beta}_c$ conditional on the sequence of choices of all the individuals in region c, whereas $g(\boldsymbol{\beta}_c|\boldsymbol{\theta})$ is the unconditional distribution. The conditional expectation of $\boldsymbol{\beta}_c$ is given by

$$\bar{\boldsymbol{\beta}}_c = E\left[\boldsymbol{\beta}_c|\boldsymbol{y}_c,\boldsymbol{X}_c,\boldsymbol{\theta}\right] = \frac{\int_{\boldsymbol{\beta}_c}\boldsymbol{\beta}_c f(\boldsymbol{y}_c|\boldsymbol{X}_c,\boldsymbol{\beta}_c)g(\boldsymbol{\beta}_c|\boldsymbol{\theta})d\boldsymbol{\beta}_c}{\int_{\boldsymbol{\beta}_c} f(\boldsymbol{y}_c|\boldsymbol{X}_c,\boldsymbol{\beta}_c)g(\boldsymbol{\beta}_c|\boldsymbol{\theta})d\boldsymbol{\beta}_c} \tag{9}$$

This expectation gives us the conditional mean of the distribution of the spatially random parameter. The simulator for this expectation is:

$$\hat{\bar{\boldsymbol{\beta}}}_c = \hat{E}\left[\boldsymbol{\beta}_c|\boldsymbol{y}_c,\boldsymbol{X}_c,\hat{\boldsymbol{\theta}}\right] = \frac{\frac{1}{R}\sum_{r=1}^{R}\hat{\boldsymbol{\beta}}_{cr}\prod_{i=1}^{n_c} f^*(y_{ci}|\boldsymbol{x}_{ci},\hat{\boldsymbol{\beta}}_{cr})}{\frac{1}{R}\sum_{r=1}^{R}\prod_{i=1}^{n_c} f^*(y_{ci}|\boldsymbol{x}_{ci},\hat{\boldsymbol{\beta}}_{cr})} \tag{10}$$

This estimator is the region-specific estimate, and can be computed in `Rchoice` using `effect.Rchoice` function and plotted using the function `plot`. In the following lines the municipality's coefficient for all the random parameters is plotted using a kernel approximation:

[12]:
```
> # Plot municipality-specific coefficient
> par(mfrow = c(3, 3))
> plot(ran_2, par = "noise",  type = "density", main = "Noise Pol.")
> plot(ran_2, par = "airpol", type = "density", main = "Air Pol.")
> plot(ran_2, par = "watpol", type = "density", main = "Water Pol.")
> plot(ran_2, par = "vispol", type = "density", main = "Visual Pol.")
> plot(ran_2, par = "waspol", type = "density", main = "Garbage Pol.")
> plot(ran_2, par = "lmdinc", type = "density", main = "Development")
> plot(ran_2, par = "lpop",   type = "density", main = "Population")
```

[12]:
```
Output reproduced in Figure 1
```

The red area under the kernel distribution illustrates the proportion of municipalities with a positive conditional mean. The most relevant result is that size (`lpop`) seems to be a positive externality for almost 50% of the municipalities, evidencing substantial spatial heterogeneity. This important result is obscured by the traditional ordered probit model.

We might also plot the 95% confidence interval for the conditional means of airpol and noise for the first 50 municipalities by typing:

[13]:
```
> # Plot region-specific confidence intervals.
> par(mfrow = c(1, 2))
> plot(ran_2, par = "airpol", ind =  TRUE, id = 1:50, ylab = "Municipalities")
> plot(ran_2, par = "noise", ind =  TRUE, id = 1:50, ylab = "Municipalities")
```

[13]:
```
Output reproduced in Figure 2
```

In terms of consistency of the regional-specific estimates, it is expected that $\hat{\bar{\boldsymbol{\beta}}}_c \xrightarrow{p} \boldsymbol{\beta}_c$ as $n_c \to \infty$. That is, if we have more information about the choices made by the individuals in each region, then we are in better position to identify where each region coefficient lies on $g(\boldsymbol{\beta}_c)$ (see for example Train 2009, Revelt, Train 2000, Sarrias, Daziano 2018).

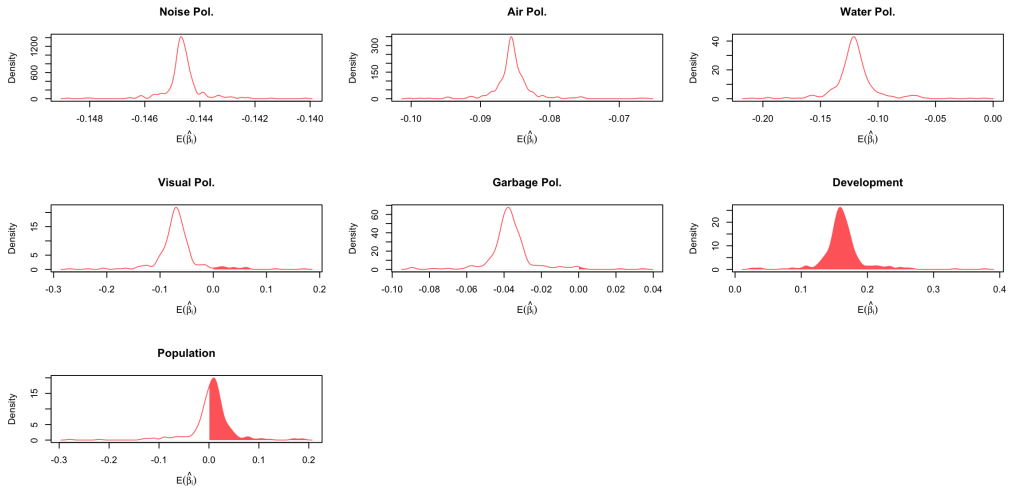


Figure 1: Plot of municipality-specific coefficients

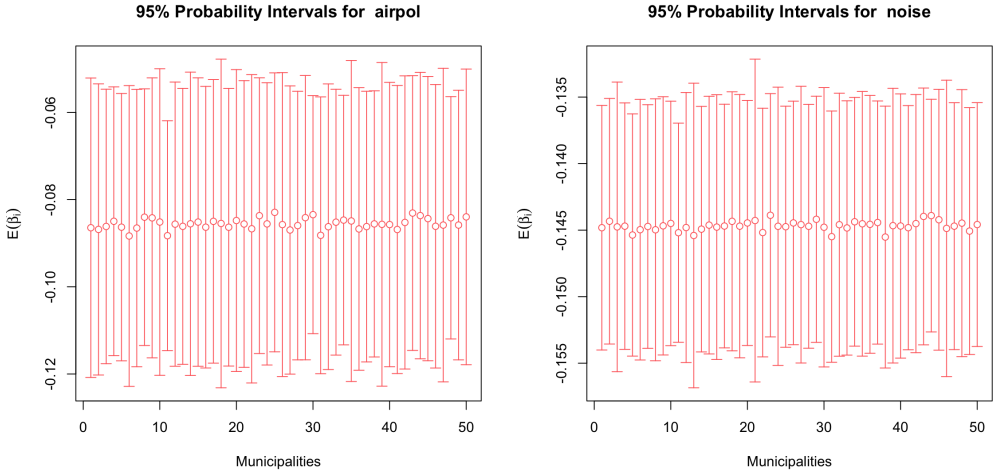


Figure 2: Plot of region-specific confidence intervals

5 Conclusion

This paper contributes to the literature of spatial econometric models that deal with spatially non-stationary process by assuming unobserved heterogeneity. This modelling approach has been widely used in discrete choice modeling, but it can also be implemented to capture and model observed and unobserved spatial heterogeneity. One of the main advantages of this modelling approach is that allows the analyst to include variables at the individual level, which mitigate the ecological fallacy problem, and to add more flexibility regarding the shape and boundedness of the coefficients.

Spatial heterogeneity is represented by some distribution $g(\boldsymbol{\beta}_c)$, which can take any continuous shape, and the analyst must choose the distribution a priori. The choice of the distribution may be guided by theoretical reasons regarding the domain and bound of the coefficients. It also discussed some extensions that can be useful in order to take into consideration the geographical location of the regions, as well as the spatial correlation of the parameters. Although the unobserved spatial heterogeneity using continuous distributions has very appealing features, the probability for each region does not have a closed form solution. Therefore, we need to simulate this probability and estimate the parameters using SML, which can be very costly in terms of computational time.

This study also shows how the `Rchoice` package can be used to estimate this type of models. To do so, we provide a simple example for ordered probit models, focusing on how the determinants of individuals' self-assessed health status might vary across space.

This work can be extended in different ways. First, one of the main concerns and limitations of the model is that the estimation requires computing the product of the probabilities for all individuals in a given region. Thus, if the number of individuals is too high, the estimation method may run into numerical difficulties. To overcome this problem some of the two methods proposed by Lee (2000) can be studied under the spatial context. These methods alleviate the numerical problems by interchanging the inner product with the outer summation. Another possible extension is to study the behavior of the parameters with small and large samples using Bayesian and EM algorithms. Finally, more empirical applications are needed in order to understand the strengths and weaknesses for estimating models with locally varying coefficients using unobserved heterogeneity.

References

Ali K, Partridge MD, Olfert MR (2007) Can geographically weighted regressions improve regional analysis and policy making? *International Regional Science Review* 30[3]: 300–329. CrossRef.

Anselin L (1988) *Spatial Econometrics: Methods and Models*, Volume 4. Springer. CrossRef.

Borjas GJ, Sueyoshi GT (1994) A two-stage estimator for probit models with structural group effects. *Journal of Econometrics* 64[1]: 165–182. CrossRef.

Bürkner P (2018) Advanced bayesian multilevel modeling with the r package brms. *R Journal* 10[1]. CrossRef.

Brunsdon C, Aitkin M, Fotheringham S, Charlton M (1999) A comparison of random coefficient modelling and geographically weighted regression for spatially non-stationary regression problems. *Geographical and Environmental Modelling* 3: 47–62

Brunsdon C, Fotheringham S, Charlton M (1998) Geographically weighted regression. *Journal of the Royal Statistical Society: Series D (the Statistician)* 47[3]: 431–443. CrossRef.

Casetti E (1972) Generating models by the expansion method: Applications to geographical research. *Geographical Analysis* 4[1]: 81–91. CrossRef.

Dong G, Harris R, Jones K, Yu J (2015) Multilevel modelling with spatial interaction effects with application to an emerging land market in beijing, china. *PloS One* 10[6]: e0130761. CrossRef.

Dong G, Ma J, Harris R, Pryce G (2016) Spatial random slope multilevel modeling using multivariate conditional autoregressive models: A case study of subjective travel satisfaction in beijing. *Annals of the American Association of Geographers* 106[1]: 19–35. CrossRef.

Dong G, Nakaya T, Brunsdon C (2018) Geographically weighted regression models for ordinal categorical response variables: An application to geo-referenced life satisfaction data. *Computers, Environment and Urban Systems* 70: 35–42. CrossRef.

Duranton G, Puga D (2004) Micro-foundations of urban agglomeration economies. In: Henderson V, Thisse J (eds), *Handbook of Regional and Urban Economics*, Volume 4. Elsevier, 2063–2117. CrossRef.

Elhorst JP (2014) *Spatial Panel Data Models*. Springer. CrossRef.

Finley AO (2011) Comparing spatially-varying coefficients models for analysis of ecological data with non-stationary and anisotropic residual dependence. *Methods in Ecology and Evolution* 2[2]: 143–154. CrossRef.

Fotheringham AS, Brunsdon C (1999) Local forms of spatial analysis. *Geographical Analysis* 31[4]: 340–358. CrossRef.

Gelfand AE, Kim HJ, Sirmans CF, Banerjee S (2003) Spatial modeling with spatially varying coefficient processes. *Journal of the American Statistical Association* 98[462]: 387–396. CrossRef.

Goldstein H (1987) *Multilevel Models in Education and Social Research*. Oxford University Press

Gourieroux C, Monfort A (1991) Simulation based inference in models with heterogeneity. *Annales d'Economie et de Statistique* 20-21: 69–107. CrossRef.

Gourieroux C, Monfort A (1997) *Simulation-Based Econometric Methods*. Oxford University Press. CrossRef.

Greene WH, Hensher DA (2010) *Modeling Ordered Choices: A Primer*. Cambridge University Press. CrossRef.

Hajivassiliou VA, Ruud PA (1986) Classical estimation methods for ldv models using simulation. In: Engle R, McFadden D (eds), *Handbook of Econometrics*, Volume 4. Elsevier. CrossRef.

Hashiguchi Y, Tanaka K (2014) Agglomeration and firm-level productivity: A bayesian spatial approach. *Papers in Regional Science* 94[S1]: S95–S114. CrossRef.

Hensher D, Greene WH (2003) The mixed logit model: The state of practice. *Transportation* 30[2]: 133–176. CrossRef.

Jetz W, Rahbek C, Lichstein JW (2005) Local and global approaches to spatial data analysis in ecology. *Global Ecology and Biogeography* 14[1]: 97–98. CrossRef.

Jones K (1991) Specifying and estimating multi-level models for geographical research. *Transactions of the Institute of British Geographers* 16: 148–159. CrossRef.

Kelejian HH, Prucha IR (1998) A generalized spatial two-stage least squares procedure for estimating a spatial autoregressive model with autoregressive disturbances. *The Journal of Real Estate Finance and Economics* 17[1]: 99–121

Kelejian HH, Prucha IR (1999) A generalized moments estimator for the autoregressive parameter in a spatial model. *International Economic Review* 40[2]: 509–533. CrossRef.

Lee L (2000) A numerically stable quadrature procedure for the one-factor random-component discrete choice model. *Journal of Econometrics* 95[1]: 117–129. CrossRef.

Lee LF (1992) On efficiency of methods of simulated moments and maximum simulated likelihood estimation of discrete response models. *Econometric Theory* 8[4]: 518–552. CrossRef.

Revelt D, Train K (1998) Mixed logit with repeated choices: Households' choices of appliance efficiency level. *Review of Economics and Statistics* 80[4]: 647–657. CrossRef.

Revelt D, Train K (2000) Customer-specific taste parameters and mixed logit: Households' choice of electricity supplier. Working Paper. Department of Economics, UCB

Robinson WS (1950) Ecological correlations and the behavior of individuals. *American Sociological Review* 15[3]: 351–357. CrossRef.

Sarrias M (2016) Discrete choice models with random parameters in R: The Rchoice package. *Journal of Statistical Software* 74[10]: 1–31. CrossRef.

Sarrias M (2019) Do monetary subjective well-being evaluations vary across space? comparing continuous and discrete spatial heterogeneity. *Spatial Economic Analysis* 14[1]: 53–87. CrossRef.

Sarrias M, Daziano RA (2018) Individual-specific point and interval conditional estimates of latent class logit parameters. *Journal of Choice Modelling* 27: 50–61. CrossRef.

Stern S (1997) Simulation-based estimation. *Journal of Economic Literature* 35[4]: 2006–2039

Swamy PAVB (1971) *Statistical Inference in Random Coefficient Regression Models*. Springer Berlin. CrossRef.

Train K (2009) *Discrete Choice Methods with Simulation*. Cambridge University Press. CrossRef.

Wheeler D, Tiefelsdorf M (2005) Multicollinearity and correlation among local regression coefficients in geographically weighted regression. *Journal of Geographical Systems* 7[2]: 161–187. CrossRef.

Withers SD (2001) Quantitative methods: Advancement in ecological inference. *Progress in Human Geography* 25[1]: 87–96. CrossRef.

REGION
The Journal of ERSA
Powered by WU

Volume 6, Number 3, 2019, R1–R57
DOI: 10.18335/region.v6i3.267

journal homepage: region.ersa.org

REAT: A Regional Economic Analysis Toolbox for R

Thomas Wieland[1]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany

Received: 7 June 2019/Accepted: 4 November 2019

Abstract. Methods of regional economic analysis are widely used in regional and urban economics as well as in economic geography. This paper introduces the REAT (Regional Economic Analysis Toolbox) package for the programming environment R, which provides a collection of mathematical regional analysis methods in a user-friendly way. The focus is on the identification of regional inequality, beta and sigma convergence, measurement of agglomerations, point-based measures of clustering and accessibility, as well as regional growth. The theoretical basics of the applications are briefly introduced, while the usage of the most important functions is presented and explained using real data.

1 Introduction

Methods of regional economic analysis (or regional analysis) are used frequently in theory-based, empirical studies from regional and urban economics as well as (quantitative) economic geography. These methods aim at analyzing some of the most important issues in the mentioned research fields, including (but not limited to) the existence and evolution of agglomerations, regional economic growth and regional disparities (Capello, Nijkamp, 2009; Dinc, 2015; Farhauer, Kröll, 2014; Schätzl, 2000). In any of the mentioned fields, a growing amount of quantitative data has to be processed when using traditional or novel methods and models of regional analysis. This paper introduces the package (add-on) `REAT` (Regional Economic Analysis Toolbox) (Wieland, 2019) for the programming environment `R` (R Core Team, 2018a). The package provides a collection of mathematical regional analysis applications, designed in a relatively user-friendly way.

The main topics in the regional analysis context can be summarized as follows, showing also the structure of the present paper with respect to the presented approaches and their application in `REAT`:

1. Identifying regional inequality (or regional disparities) using indicators of concentration and/or dispersion (Section 2)
2. Regional disparities over time leading to the concept of beta and sigma convergence (Section 3)
3. Measuring agglomerations, which means the specialization of regions and the spatial concentration of industries as well as more complex cluster indices (Section 4)
4. Point-based measures of clustering and accessibility (Section 5)
5. Regional growth, especially shift-share analysis (Section 6)

Note that, in its original form, the open source software `R` is a command-line environment including a lot of mathematical and statistical features. For the installation of `R` and its packages as well as the basics of navigation and implemented statistical functions, see the `R` documentations (R Core Team, 2018b). A good supplement for working with `R` is `RStudio` (RStudio Team, 2016). The `REAT` package deals with several `R` data types: The most functions require and calculate `numeric vectors`, but, in some cases, also objects of type `matrix`, `data frame` and `list`, depending on the complexity of calculation. For a quick introduction to the data types in `R` and their properties, see e.g. Kabacoff (2017).

2 Concentration, dispersion and regional disparities

2.1 Indicators of concentration and dispersion

Regional disparities are a frequent topic in economic geography and regional economics. The spatial inequality with respect to e.g. regional output, income or employment is an essential element of polarization theory (Myrdal, 1957) and "New Economic Geography" (Krugman, 1991; Fujita et al., 2001). Assessing regional disparities is possible using concentration and dispersion indicators, which belong to the univariate and descriptive analysis in statistics. Apart from regional economics, these measures are used in several contexts, such as competition economics (market concentration of firms) or welfare economics (income inequality). For a review of the most common indicators with respect to regional inequality, see Portnov, Felsenstein (2010), for studies comparing different indicators in the regional economic context using empirical data, see e.g. Gluschenko (2018); Habánik et al. (2013); Huang, Leung (2009); Palan (2017); Petrakos, Psycharis (2016).

Concentration is operationalized as the discrepancy between an empirical distribution of a variable x (e.g. annual turnover, income, gross domestic product [GDP]) with n observations or objects (e.g. competing firms, households, regions) and a (theoretical) equal distribution or a reference distribution (e.g. population distribution). Dispersion indicators aim at the deviation from the arithmetic mean of x, $\bar{x}$. In this context, Portnov, Felsenstein (2005, 2010) distinguish between measures of deprivation and variation.

Typical measures of regional disparities are the Gini coefficient, the Herfindahl-Hirschman index and the coefficient of variation (Lessmann, 2005). The most popular measure of concentration is the Gini coefficient (Gini, 1912) in combination with the Lorenz curve (Lorenz, 1905). There are several calculation approaches for the Gini coefficient, all producing the same result. The Lorenz curve is a graphical indicator, showing the deviation of the empirical shares of the regarded variable x from a (theoretical) equal distribution. Another well-known indicator is the Herfindahl-Hirschman index, which was developed independently by Hirschman (1945) and Herfindahl (1950), both in the context of competition economics. Several other concentration indicators are also applied in the fields of regional economics with respect to regional disparities, such as the Hoover coefficient (Hoover, 1936) and the Theil coefficient (Theil, 1967).

Except for the standard deviation, whose unit is equal to the unit of x, all common indicators are dimensionless. Most of them (except for standard deviation and coefficient of variation) have a fixed value range, normally between zero (indicating complete equality/dispersion) and one (indicating complete inequality/concentration).

Most of the common indicators are mathematically formulated in an unweighted and in a weighted form, while, in the context of regional disparities, the latter is mostly done using the regions' proportion of the total (e.g. national) population (Doran, Jordan 2013; Lessmann 2014; Mussini 2017; Petrakos, Psycharis 2016; for a critical discussion of weighting these coefficients, see Gluschenko 2018). In the literature, there are different formulations where the weighted coefficients also include a weighted arithmetic mean. Note that, in the case of the population-weighted Gini coefficient, a weighted arithmetic mean is mandatory to keep the indicators' value range.

Especially when dealing with GDP per capita as an indicator of regional economic output, several recent studies use dispersion measures rather than concentration measures, especially the (weighted) coefficient of variation (e.g. Lessmann 2005, 2014, 2016;

Lessmann, Seidel 2017; Petrakos, Psycharis 2016). This dispersion indicator is a dimensionless normalization of the standard deviation. Weighting the coefficient of variation with population shares was introduced by Williamson (1965), which has led to calling this coefficient the Williamson index. As regional incomes or outputs are not normally distributed in most cases, resulting in biased arithmetic means used in the calculation of dispersion measures, the regarded variable may be log-transformed, which means replacing x_i with $\log(x_i)$ in the calculations.

Table 1 shows the common indicators, including their (population-)weighted and their normalized form (if there exist any) and the corresponding value ranges. The formulae are shown in a way that includes several ways of application. The regarded variable is always named x_i, while the (population) weighting is called w_i. Some indicators, such as the Hoover or the Coulter coefficient, require a variable representing a reference distribution the shares of x_i are compared to. This reference is *not* a weighting. However, in many studies, the regional population is also used for the reference distribution. In these cases, reference and weighting are the same data. The reference distribution may also be equal to $1/n$.

Several indicators are also used for the analysis of regional specialization or the spatial concentration of industries, such as the Hoover coefficient or the Herfindahl-Hirschman index or its inverse ($1/HHI$; also known as the "equivalent number" in the competition context). Other coefficients of concentration and specialization are discussed in Section 4. The last coefficient in Table 1, the mean square successive difference (von Neumann et al., 1941) is a measure for time variability not originating from but also transferable to regional economics.

2.2 Application in REAT

2.2.1 REAT functions for concentration and dispersion indicators

Table 2 shows the functions for concentration and dispersion measures implemented in the `REAT` package. All functions require at least one argument, a `numeric vector` with a length equal to n, containing the regarded variable x (e.g. income) with i observations (e.g. regions), where $i = 1, ..., n$. This data may be a single `vector` or a column of a `data frame` or `matrix`.

An optional weighting of the vector `x` can be done using the function argument `weighting` which is also a `numeric vector` of length n. By default, the functions remove missing (`NA`) values. The `hoover()` function always needs a reference distribution (see the Hoover coefficient formula in Table 1), which is stated via the `ref` argument, also requiring a `numeric vector` of length n. If no reference variable is stated (`ref = NULL`), the reference is set to $1/n$.

All functions (except for `disp()`) return the single value of the computed coefficient. In the relevant cases (`gini()`, `gini2()`, `herf()` and `cv()`), a normalization of the coefficient is possible using the function argument `coefnorm = TRUE`, returning the normalized coefficient instead of the raw coefficient. The function `disp()` is a wrapper for all mentioned functions, calculating all coefficients (except for the *MSSD*) at once for one vector `x` or a set of variables/columns from a `data frame` or `matrix`.

Note that there are two functions for the Gini coefficient, `gini()` and `gini2()`, both producing the same result in the unweighted case. The former function is designed for income inequality, where the `weighting` option is designed for the calculation of the Gini coefficient for groups (e.g. income classes), where the weighting represents the group mean. The function `gini2()` is designed for the population-weighted analysis of regional inequality.

2.2.2 Application example: Small-scale regional disparities in health care provision

Regional inequality with respect to health care providers is a topic of high societal significance. In Germany, the health care planning system (*Kassenärztliche Bedarfsplanung*) attempts to flatten the disparities of local health care provision (Kassenärztliche Bundesvereinigung, 2013). Here, we analyze small-scale regional disparities in health care

Table 1: Indicators of concentration and dispersion for analyzing regional disparities

Indicator	Unweighted	Weighted	Normalized
Gini	$G = \frac{1}{2n^2\bar{x}} \sum_{i=1}^{n} \sum_{j=1}^{n} \lvert x_i - x_j \rvert$ $0 \leq G \leq 1 - \frac{1}{n}$	$G^w = \frac{1}{2\bar{x}^w} \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \lvert x_i - x_j \rvert$ $0 \leq G \leq 1 - \frac{1}{n}$	$G^* = \frac{n}{n-1} G$ $0 \leq G^* \leq 1$
HHI	$HHI = \sum_{i=1}^{n} (\frac{x_i}{\sum_{i=1}^{n} x_i})^2$ $\frac{1}{n} \leq HHI \leq 1$		$HHI^* = \frac{HHI - \frac{1}{n}}{1 - \frac{1}{n}}$ $0 \leq HHI^* \leq 1$
Hoover	$HC = \frac{1}{2} [\sum_{i=1}^{n} \lvert \frac{x_i}{\sum_{i=1}^{n} x_i} - \frac{r_i}{\sum_{i=1}^{n} r_i} \rvert]$ $0 \leq HC \leq 1$	$HC^w = \frac{1}{2} [\sum_{i=1}^{n} w_i \lvert \frac{x_i}{\sum_{i=1}^{n} x_i} - \frac{r_i}{\sum_{i=1}^{n} r_i} \rvert]$ $0 \leq HC \leq 1$	
Theil	$TC = \frac{1}{n} \sum_{i=1}^{n} \ln(\frac{\bar{x}}{x_i})$ $0 \leq TC \leq 1$	$TC^w = \frac{1}{n} \sum_{i=1}^{n} w_i \ln(\frac{\bar{x}}{x_i})$ $0 \leq TC^w \leq 1$	
Coulter		$CC = \sqrt{\frac{1}{2} [\sum_{i=1}^{n} w_i (\frac{x_i}{\sum_{i=1}^{n} x_i} - \frac{r_i}{\sum_{i=1}^{n} r_i})^2]}$ $0 \leq CC \leq 1$	
Atkinson	$AI = 1 - [\frac{1}{n} \sum_{i=1}^{n} x_i^{1-\epsilon}]^{\frac{1}{1-\epsilon}}$ $0 \leq AI \leq 1$		
Dalton	$\delta = \frac{\log(\frac{1}{n} \sum_{i=1}^{n} x_i)}{\log(\sqrt[n]{\pi_{i=1}^{n} x_i})}$ $0 \leq \delta \leq \infty$		
SD	$s = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$ $0 \leq s \leq \infty$	$s^w = \sqrt{\frac{1}{n} \sum_{i=1}^{n} w_i (x_i - \bar{x})^2}$ $0 \leq s \leq \infty$	see CV
CV	$v = \frac{1}{\lvert \bar{x} \rvert} \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2}$ $0 \leq v \leq \infty$	see Williamson	$v^* = \frac{v}{\sqrt{n}}$ $0 \leq v^* \leq 1$
Williamson		$WI = \frac{1}{\lvert \bar{x} \rvert} \sqrt{\frac{1}{n} \sum_{i=1}^{n} w_i (x_i - \bar{x})^2}$ $0 \leq v \leq \infty$	
MSSD	$MSSD = \frac{\sum_{t=1}^{T-1} (x_{t+1} - x_t)^2}{T-1}$		

Notes: x_i is the i-th observation of the regarded variable x (e.g. GDP [per capita] in region i), x_j is the value of the same variable with respect to object j, r_i is the i-th observation of a reference variable (e.g. population), n is the number of objects (e.g. regions), $\bar{x}$ is the arithmetic mean of x: $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$, $\bar{x}^w$ is the weighted arithmetic mean of x: $\bar{x}^w = \frac{1}{n} \sum_{i=1}^{n} w_i x_i$, w_i and w_j are the population weightings: $P_i / \sum_{i=1}^{n} P_i$ and $P_j / \sum_{j=1}^{n} P_j$, where P_i and P_j are the population sizes of regions i and j, respectively, ϵ is an inequality aversion parameter ($0 < \epsilon < \infty$) for the Atkinson index, t is a given time period and T is the number all regarded time periods.
Compiled from: Charles-Coll (2011); Cracau, Durán Lima (2016); Damgaard, Weiner (2000); Gluschenko (2018); Heinemann (2008); Kohn, Öztürk (2013); Portnov, Felsenstein (2005, 2010); Taylor, Cihon (2004); Schätzl (2000); Störmann (2009)

provision in two neighboring German counties (Göttingen and Northeim) using the data on medical practices and local population from Wieland, Dittrich (2016). The data is stored in the datasets `GoettingenHealth1` and `GoettingenHealth2`, both included as example datasets in the `REAT` package. The study area is segmented into 420 districts, representing either city districts of larger cities or villages and hamlets.

The dataset `GoettingenHealth2` contains these 420 regions with an individual ID

Table 2: REAT functions for concentration and dispersion indicators

Indicator	REAT function	Mandatory arguments	Optional arguments	Output
Gini/ Lorenz	gini()	vector x	weighting vector, remove NAs, Lorenz curve, normalization	value: G or G^* or G^w, optional: plot (LC)
	gini2()	vector x	weighting vector P_i, remove NAs, normalization	value: G or G^* or G^w,
	lorenz()	vector x	weighting vector, remove NAs,	plot LC, value: G or G^w and/or G^*
HHI	herf()	vector x	remove NAs, normalization	value: HHI or HHI^* or N_{HHI}
Hoover	hoover()	vector x reference vector r_i	weighting vector P_i, remove NAs	value: HC or HC^w
Theil	theil()	vector x	weighting vector P_i, remove NAs	value: TC or TC^w
Coulter	coulter()	vector x	weighting vector P_i, remove NAs	value: CC
Atkinson	atkinson()	vector x	remove NAs, epsilon	value: AI
Dalton	dalton()	vector x	remove NAs	value: δ
SD	sd2()	vector x	weighting vector, remove NAs, treating as sample	value: s or s^W
CV	cv()	vector x	weighting vector, remove NAs, normalization, treating as sample	value: v or v^W or v^*
Williamson	williamson()	vector x, weighting vector P_i	remove NAs	value: WI
MSSD	mssd()	vector x	remove NAs	value: $MSSD$
All indicators	disp()	vector x or vectors $x_1, x_2, ...$ from dataframe	weighting vector P_i, remove NAs	matrix with 13 (no weighting) or 19 indicators (incl. weighted)

Source: own compilation.

(column `district`) and geographic coordinates (columns `lat` and `lon`, respectively) and the number of general practitioners, psychotherapists and pharmacies located there (columns `phys_gen`, `psych` and `pharm`, respectively) as well as the local population (column `pop`). First, we load the dataset:

```
data(GoettingenHealth2)
```

Now, we investigate how the health care providers are dispersed over the whole area. In the first step, we calculate the Gini coefficient for the concentration of general practitioners using the `REAT` function `gini()`:

```
gini (GoettingenHealth2$phys_gen)
[1] 0.8386269
```

The empirical Gini coefficient is equal to 0.839, indicating a relatively strong concentration. If we want to calculate the normalized (unbiased) indicator instead, we use the same function with the optional argument `coefnorm = TRUE`:

```
gini (GoettingenHealth2$phys_gen, coefnorm = TRUE)
[1] 0.8406284
```

In the same way, we calculate e.g. the Herfindahl-Hirschman index, non-normalized and normalized:

```
herf (GoettingenHealth2$phys_gen)
[1] 0.01528053

herf (GoettingenHealth2$phys_gen, coefnorm = TRUE)
[1] 0.01293036
```

Remember that the minimum of HHI is $1/n$ (here: $1/420 \approx 0.00238$) and the minimum of HHI^* is equal to zero.

If we want to inspect the concentration graphically, we could use the Lorenz curve, which can be plotted using either the functions `gini()` or `lorenz()`. Here, we use `gini()`, tell the function to plot the curve (`lc = TRUE`), and include several graphical parameters (such as `lc.col` for the color of the Lorenz curve or `lcx` and `lcy` for the x/y axes labels). As we want to compare the population distribution to the location distribution, we start by plotting the Lorenz curve for the local population:

```
gini(GoettingenHealth2$pop, lc = TRUE, lsize = 1, le.col = "black",
lc.col = "orange", lcx = "Shares of districts", lcy = "Shares of
providers", lctitle = "Spatial concentration of health care
providers", lcg = TRUE, lcgn = TRUE, lcg.caption =
"Population 2016:", lcg.lab.x = 0, lcg.lab.y = 1)
# Gini coefficient and Lorenz curve for the no. of inhabitants
[1] 0.5840336
```

Now, we overlay the Lorenz curves of general practitioners and psychotherapists, which means adding two more curves (function argument `add.lc = TRUE`):

```
gini(GoettingenHealth2$phys_gen, lc = TRUE, lsize = 1, add.lc = TRUE,
lc.col = "red", lcg = TRUE, lcgn = TRUE, lcg.caption =
"Physicians 2016:", lcg.lab.x = 0, lcg.lab.y = 0.85)
# Adding Gini coefficient and Lorenz curve for the general practitioners
[1] 0.8386269

gini(GoettingenHealth2$psych, lsize = 1, lc = TRUE, add.lc = TRUE,
lc.col = "blue", lcg = TRUE, lcgn = TRUE, lcg.caption =
"Psychotherapists 2016:", lcg.lab.x = 0, lcg.lab.y = 0.7)
# Adding Gini coefficient and Lorenz curve for psychotherapists
[1] 0.9329298
```

Our commands result in the output of Figure 1, showing three Lorenz curves (population, general practitioners and psychotherapists) and the line of equality (diagonal). All three empirical distributions differ from an equal distribution. In about 72% of the regions, representing about 23% of the whole population (orange curve; $G \approx 0.584$), no general practitioner is located (red curve; $G \approx 0.839$). But the psychotherapists are more concentrated, as they are located only in about 13% of all districts (blue curve; $G \approx 0.933$). As we can see, the physicians are more concentrated than the inhabitants but the psychotherapists are more concentrated than the physicians.

Now, we calculate all mentionened concentration and dispersion coefficients at once for all three types of providers using the function `disp()`, including a population weighting:

```
disp(GoettingenHealth2[c(5,6,7)], weighting = GoettingenHealth2$pop)
# column 5 = general practitioners, column 6 = psychotherapists,
# column 7 = pharmacies, column "pop" = local population
```

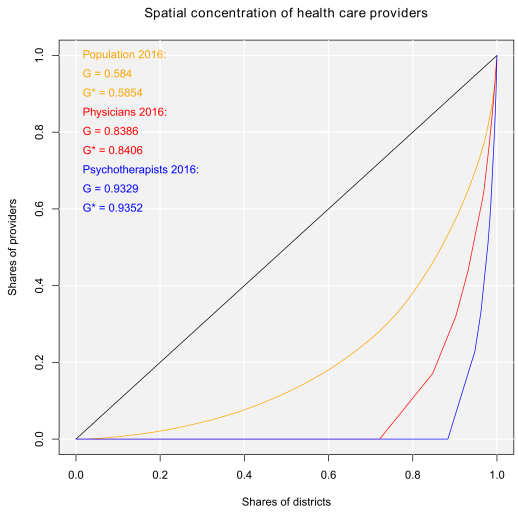


Figure 1: Lorenz curves for the spatial concentration of health care providers

Our output is:

```
Concentration and dispersion measures
Note: w = weighted, n = normalized, eq = equivalent number

                phys_gen         psych         pharm
Gini        0.838626907  0.932929782  0.891547619
Gini n      0.840628403  0.935156345  0.893675418
Gini w      0.629454516  0.770895945  0.705628058
Gini w n    0.630956794  0.772735792  0.707312135
HHI         0.015280527  0.038494685  0.024166667
HHI n       0.012930361  0.036199923  0.021837709
HHI eq     65.442769020 25.977611940 41.379310345
Hoover      0.721428571  0.883333333  0.838095238
Hoover w    0.001852337  0.003130602  0.003418787
Theil                NA           NA           NA
Theil w              NA           NA           NA
Coulter     0.049850824  0.123305927  0.065569205
Atkinson    0.761164110  0.900755425  0.854223763
Dalton               NA           NA           NA
SD          1.714506606  1.095496987  0.865286915
SD w        4.010246439  1.847716870  2.401476794
CV          2.330397328  3.899226565  3.028504203
CV n        0.113847359  0.190489683  0.147952112
Williamson  1.429449565  1.965446423  1.709288672
```

We conclude that any concentration/dispersion measure is the highest for psychotherapists and the lowest for the general practitioners, while the values for pharmacies lie between them. The regional disparities with respect to pharmacies are higher than those with respect to general practitioners, while the most unequal distribution is that of psychotherapists. In other words: The pharmacies are more spatially concentrated than the general practitioners and the psychotherapists are the most concentrated health locations here.

In most cases, population weighting reduces the coefficient values. That is, because districts with a large (small) population have a high (low) impact on the resulting coefficient and the districts without health service providers are also small districts. Furthermore, as the regarded variables contain zero values (which means no health service locations), the Theil coefficient (including the term $\ln(\bar{x}/x_i)$) and the Dalton coefficient (including the n-th root) cannot be computed, resulting in an output of `NA`.

The visible output of any function presented above can be saved in a new `R` object:

```
gini_phys <- gini (GoettingenHealth2$phys_gen)
# save as gini_phys (numeric vector of length = 1)
```

We can simply access our result:

```
gini_phys
[1] 0.8386269
```

The function `disp()` returns a `matrix` with 13 rows (when only unweighted coefficients are computed) or 19 rows (in the case of additional weighted coefficients) and one column for each regarded variable:

```
disp_Goettingen <- disp(GoettingenHealth2[c(5,6,7)],
weighting = GoettingenHealth2$pop)
# save as disp_Goettingen (matrix)
```

We call our results:

```
disp_Goettingen

                phys_gen       psych       pharm
Gini          0.83862691  0.93292978  0.89154762
Gini n        0.84062840  0.93515634  0.89367542
...
```

3 Regional convergence

3.1 The concept of beta and sigma convergence

Regional convergence is derived from (regional) growth theory (for an extensive survey, see Barro, Sala-i Martin 2004) and means the decline of regional disparities *over time*. The neoclassical growth model states that a region's economic output (e.g. GDP per capita) depends on its stock of factors of production, capital and labor (aggregate production function), on condition of constant returns to scale and diminishing marginal product of the factor inputs. As a consequence, regions with a high (low) initial level of factor input grow slower (faster) than "poor" ("rich") regions, what is called beta convergence. It is assumed that all regions converge to the same regional output level (steady-state). Sigma convergence means the decline of regional inequality with respect to regional output over time itself (Allington, McCombie, 2007; Capello, Nijkamp, 2009).

Both types of convergence can be tested empirically, as presented in Table 3. When testing for beta convergence, the natural logarithms of output growth over T time periods in i regions is regressed against the natural logarithms of the initial output values at time t. The original convergence formula was presented by Barro, Sala-i Martin (2004) using a nonlinear least squares (NLS) estimation approach. But in many cases, a linear transformation is used which allows for ordinary least squares (OLS) estimation (Allington, McCombie, 2007; Dapena et al., 2016; Schmidt, 1997; Young et al., 2008). The outcome variable of the convergence equation can be the regional growth between two years (e.g. Young et al. 2008) or the average growth rate per year (e.g. Goecke, Hüther 2016; Puente 2017; Weddige-Haaf, Kool 2017). Significance tests are carried out with t-tests for the regression coefficients and, in the OLS case, the F-test for the significance of R^2.

The estimated parameter of interest is the slope of the model, here denoted β (that is why the modeled process is called *beta* convergence): If $\beta < 0$ and statistically significant, there is *absolute* beta convergence. If additional variables (conditional variables) are included into the convergence equation, we have a test for *conditional* beta convergence. A further interpretation of the β coefficient is possible using the speed of convergence, λ, and H, the so-called half-life, which means the time (measured in the regarded time periods) to reduce the regional disparities by one half (Allington, McCombie, 2007; Schmidt, 1997).

Sigma convergence (which is named after the Greek letter for the standard deviation, σ) can be tested in two ways depending on the number of time periods: The regional

Table 3: Beta and sigma convergence

Type of convergence	Two time periods	More than two time periods
Beta convergence	absolute	
and estimation type	NLS	NLS
	$\frac{1}{T}\ln(\frac{Y_{i,t2}}{Y_{i,t1}}) = \alpha - [\frac{(1-e^{-\beta T})}{T}]\ln(Y_{i,t1}) + \epsilon$	$\frac{1}{T}\sum_{t=1}^{T}\ln(\frac{Y_{i,t+1}}{Y_{i,t}}) = \alpha - [\frac{(1-e^{-\beta T})}{T}]\ln(Y_{i,t1}) + \epsilon$
	OLS	OLS
	$\frac{1}{T}\ln(\frac{Y_{i,t2}}{Y_{i,t1}}) = \alpha + \beta\ln(Y_{i,t1}) + \epsilon$	$\frac{1}{T}\sum_{t=1}^{T}\ln(\frac{Y_{i,t+1}}{Y_{i,t}}) = \alpha + \beta\ln(Y_{i,t1}) + \epsilon$
	conditional	
	NLS	NLS
	$\frac{1}{T}\ln(\frac{Y_{i,t2}}{Y_{i,t1}}) = \alpha - [\frac{(1-e^{-\beta T})}{T}]\ln(Y_{i,t1}) + \theta X_i + \epsilon$	$\frac{1}{T}\sum_{t=1}^{T}\ln(\frac{Y_{i,t+1}}{Y_{i,t}}) = \alpha - [\frac{(1-e^{-\beta T})}{T}]\ln(Y_{i,t1}) + \theta X_i + \epsilon$
	OLS	OLS
	$\frac{1}{T}\ln(\frac{Y_{i,t2}}{Y_{i,t1}}) = \alpha + \beta\ln(Y_{i,t1}) + \theta X_i + \epsilon$	$\frac{1}{T}\sum_{t=1}^{T}\ln(\frac{Y_{i,t+1}}{Y_{i,t}}) = \alpha + \beta\ln(Y_{i,t1}) + \theta X_i + \epsilon$
	$\beta < 0$	$\beta < 0$
	Convergence speed: $\lambda = \frac{-\ln(1+\beta)}{T}$	
	Half-life: $H = \frac{\ln(2)}{\lambda}$	
Sigma convergence	$\sigma_t = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_{i,t} - \bar{Y}_t)^2}$ or	
	$cv_t = \frac{\sigma_t}{\lvert\bar{Y}_t\rvert}$	
	$\frac{\sigma_{t1}}{\sigma_{t2}} > 1$ or	$\sigma = a + bt + \epsilon$ or
	$\frac{cv_{t1}}{cv_{t2}} > 1$	$cv = a + bt + \epsilon$
	Test statistic: $\frac{\sigma^2_{t1}}{\sigma^2_{t2}}$	$b < 0$

Notes: $Y_{i,t}$ is the regional output (e.g. GDP per capita) of region i at time t, $\bar{Y}_t$ is the arithmetic mean of $Y_{i,t}$ for all regions at time t, T is the number of regarded time periods (e.g. years), X_i is a set of other variables (conditions), σ_t is the standard deviation of the regional output of all regions, cv_t is the corresponding coefficient of variation, α, β, θ, a and b are estimated coefficients, ϵ is an error term and n is the number of regions.
Compiled from: Allington, McCombie (2007); Barro, Sala-i Martin (2004); Furceri (2005); Schmidt (1997)

inequality between all regions at time t is measured using the standard deviation, σ_t, or the coefficient of variation, cv_t, for the GDP per capita in its original or natural-logged form. If only two years are regarded, the quotient of both parameters is computed. If e.g. $\sigma_{t1} > \sigma_{t2}$, the regional inequality has declined from $t1$ to $t2$. A significance test can be applied with a simple ANOVA (analysis of variance), where the test statistic is the quotient of the underlying variances (σ^2) (Furceri, 2005; Schmidt, 1997; Young et al., 2008). Within a time series, the dispersion parameter is regressed (and plotted) against time. If the slope coefficient of time is negative, there is sigma convergence (Goecke, Hüther, 2016; Huang, Leung, 2009; Schmidt, 1997).

3.2 Application in REAT

3.2.1 REAT functions for beta and sigma convergence

Table 4 shows the functions for beta and sigma convergence as implemented in `REAT`. The analysis of beta convergence is provided by the functions `betaconv.ols()` and `betaconv.nls()` for OLS and NLS estimation, respectively. Speed of convergence and

Table 4: REAT functions for beta and sigma convergence

Convergence	REAT function	Mandatory arguments	Optional arguments	Output
Beta convergence	betaconv.ols()	vectors $Y_{i,t1}$ and $Y_{i,t2}, ..., Y_{i,T}$, t_1 and t_T	Conditions, scatterplot	visible: model estimates, invisible: list with model estimates and regression data, optional: plot
	betaconv.nls()	vectors $Y_{i,t1}$ and $Y_{i,t2}, ..., Y_{i,T}$, t_1 and t_T	Conditions, scatterplot	visible: model estimates, invisible: list with model estimates and regression data, optional: plot
	betaconv.speed()	values β and T		matrix with λ and H
Sigma convergence	sigmaconv() (when $T = 2$)	vectors $Y_{i,t1}$ and $Y_{i,t2}$, t_1 and t_T	Sigma measure, log, weighting, normalization	visible: estimates, invisible: matrix with estimates
	sigmaconv.t() (when $T > 2$)	vectors $Y_{i,t1}$ and $Y_{i,t2}, ..., Y_{i,T}$, t_1 and t_T	Sigma measure, log, weighting, normalization, line plot	visible: model estimates, invisible: matrix with model estimates, optional: plot
All at once: Beta and sigma convergence	rca()	vectors $Y_{i,t1}$ and $Y_{i,t2}, ..., Y_{i,T}$, t_1 and t_T	Beta estimation, conditions, scatterplot, sigma measure, log, weighting, line plot	visible: model estimates, invisible: list with model estimates and regression data, optional: plot

Source: own compilation.

half-life can be computed with the function `betaconv.speed()`. The ratio test of sigma convergence for two time periods can be done using the function `sigmaconv()`, while a trend regression over time is implemented into the function `sigmaconv.t()`. Both convergence types can be analyzed at once with the function `rca()`, which is a wrapper for all functions mentioned above.

The functions require (at least) two `numeric vectors`, containing the regarded variable Y (e.g. GDP per capita) for at least two different time periods, e.g. from the same `data frame`. Also the start and end time periods (t_1 and t_T) have to be stated. Optionally, a graphical output can be generated (scatterplot for beta convergence, line plot for sigma convergence with respect to longitudinal data). Furthermore, when analyzing sigma convergence, the user can choose whether Y should be log-transformed or not and/or which sigma measure is computed (variance, standard deviation or coefficient of variation; weighted or non-weighted).

Note that, unlike the functions for regional inequality indicators (Section 2), the `REAT` functions for regional convergence distinguish between a *visible* and an *invisible* output. The latter can be saved as a new `R` object. While the visible output shows the main results, the invisible output goes beyond that: `betaconv.ols()`, `betaconv.nls()` and `rca()` return a `list`, which is the most flexible data type in `R`, because it consists of a non-predetermined number of different data objects. Apart from the model results, e.g. the (transformed) regression data is returned in this invisible output.

3.2.2 Application example: Beta and sigma convergence in Germany on the county level

In this example, we look at regional convergence in Germany. The `REAT` package includes the example dataset `G.counties.gdp` with the GDP (gross domestic product), the population and the GDP per capita for the 402 counties ("Kreise") in Germany 1992 to 2014

(complete data only for 2000-2014). First, we load the dataset:

```
data (G.counties.gdp)
```

In our case, we prevent scientific notation of numbers in R and set a limit of 4 digits:

```
options(scipen = 100, digits = 4)
```

We need the columns named gdppcxxxx, containing the GDP per capita for each year, e.g. G.counties.gdp$gdppc2010 contains the GDP per capita for 2010. In the first step, we test absolute beta convergence comparing the years 2010 and 2014 with OLS estimation using the function betaconv.ols():.

```
betaconv.ols (G.counties.gdp$gdppc2010, 2010, G.counties.gdp$gdppc2014,
2014, print.results = TRUE)
# Two years, no conditions (Absolute beta convergence)
```

The output is:

```
Absolute Beta Convergence
Model coefficients (Estimation method: OLS)
           Estimate Std. Error t value     Pr (>|t|)
Alpha      0.104159   0.018934   5.501 0.00000006743
Beta      -0.007373   0.001848  -3.990 0.00007867475
Lambda     0.001850         NA      NA            NA
Halflife 374.640507         NA      NA            NA
Model summary
          Estimate F value df 1 df 2    Pr (>F)
R-Squared  0.03827   15.92    1  400 0.00007867
```

We see that both regression coefficients, α and β, are statistically significant ($t \approx 5.50$ and -3.99, respectively, both $p < 0.001$) and the linear regression model is significant as a whole ($F \approx 15.92$, $p < 0.001$). The negative sign of β shows that, on average, the higher the initial GDP per capita, the lower its growth, which indicates absolute beta convergence. However, the convergence process is very slow: The speed of convergence, represented by λ, shows a harmonization by 0.185% per year. This implies that the output gap will be reduced by 50% in approximately 375 years.

Now we check sigma convergence for the same time using the function sigmaconv(). We choose the coefficient of variation as measure, while using the GDP per capita values in their original form:

```
sigmaconv (G.counties.gdp$gdppc2010, 2010, G.counties.gdp$gdppc2014,
2014, sigma.measure = "cv", print.results = TRUE)
# Using the coefficient of variation
```

The output is:

```
Sigma convergence for two periods (ANOVA)
         Estimate F value df1 df2 Pr (>F)
CV 2010   0.03416      NA  NA  NA      NA
CV 2014   0.03316      NA  NA  NA      NA
Quotient  1.03004   1.038 401 401  0.7117
```

The coefficient of variation is a little smaller in 2014, which means the spatial inequality declined between 2010 and 2014. The quotient of the variances is slightly above one ($F = \sigma^2_{2010}/\sigma^2_{2014} \approx 1.04$), but not statistically significant ($p \approx 0.71$).

When analyzing regional convergence with REAT, it is preferable (and more convenient) to use the wrapper function rca(). Instead of repeating the results above, we test for (absolute) beta and sigma convergence between 2000 and 2014. The analysis of sigma convergence uses trend regression (function argument sigma.type = "trend") for the coefficient of variation (sigma.measure = "cv"). We also want plots for both convergence types (beta.plot = TRUE and sigma.plot = TRUE, respectively) with specific axis labels (e.g. beta.plotX = "Ln (initial GDP p.c.)"). Our code is:

```
rca (G.counties.gdp$gdppc2000, 2000, G.counties.gdp[55:68], 2014,
conditions = NULL, sigma.type = "trend", sigma.measure = "cv",
beta.plot = TRUE, beta.plotLine = TRUE, beta.plotX =
"Ln (initial GDP p.c.)", beta.plotY = "Ln (av. growth GDP p.c.)",
beta.plotTitle = "Beta convergence of German counties 2000-2014",
sigma.plot = TRUE, sigma.plotY = "cv of ln (GDP p.c.)",
sigma.plotTitle = "Sigma convergence of German counties 2000-2014")
# 14 years: 2000 (column 55) to 2014 (column 68)
# no conditions (Absolute beta convergence)
# with plots for both beta and sigma convergence
```

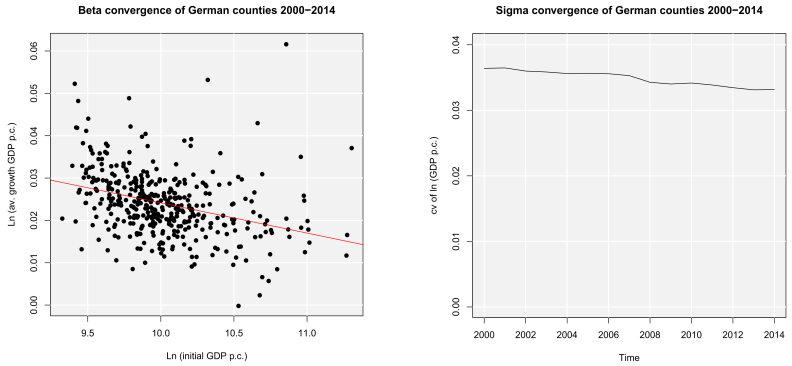
This results in the following output:

```
Regional Beta and Sigma Convergence

Absolute Beta Convergence
Model coefficients (Estimation method: OLS)
             Estimate Std. Error t value                  Pr (>|t|)
Alpha       0.0954564  0.0099087   9.634 0.000000000000000000006845
Beta       -0.0071323  0.0009885  -7.215 0.000000000002719258225550
Lambda      0.0005113         NA      NA                          NA
Halflife 1355.7282963         NA      NA                          NA
Model summary
          Estimate F value df 1 df 2           Pr (>F)
R-Squared   0.1152   52.06    1  400 0.000000000002719

Sigma convergence (Trend regression)
            Estimate Std. Error t value        Pr(>|t|)
Intercept  0.5523659 0.03084855   17.91 0.0000000001526
Time      -0.0002579 0.00001537  -16.78 0.0000000003446
Model summary
          Estimate F value df 1 df 2         Pr (>F)
R-Squared   0.9558   281.4    1   13 0.0000000003446
```

This function also produces the plots in Figures 2a and 2b, both showing a declining curve, which is a first indication of both beta and sigma convergence. The beta convergence model is statistically significant ($F \approx 52.06$, $p < 0.001$), as well as the coefficients α ($t \approx 9.63$, $p < 0.001$) and β ($t \approx -7.21$, $p < 0.001$). Again, we find evidence for absolute beta convergence because of a negative slope ($\beta \approx -0.007$). The trend regression model for sigma convergence is significant ($F \approx 281.4$, $p < 0.001$). The slope is significant and negative ($b \approx -0.00026$, $t \approx 17.91$, $p < 0.001$), which indicates sigma convergence. However, both types of convergence can be regarded as very slow processes: The half-life value shows that, resulting from the beta convergence model, the regional disparities in GDP per capita will be halved in approximately 1,356 years. When looking at the trend regression, we see that the coefficient of variation declines only by 0.00026 per year. Another aspect is that we only regarded absolute beta convergence, ignoring other spatial effects or the impact of regional policy. The latter is also not considered in neoclassical regional growth theory.

Remembering German reunification, we want to test if there are average growth differences between West Germany and East Germany (former German Democratic Republic), which leads to conditional beta convergence. The dataset `G.regions.emp` contains the column `regional`, where the counties are attributed either to West or East Germany, expressed as character string (`"West"` or `"East"`). We need to include our condition into the convergence equation. Thus, we use the `REAT` function `to.dummy()` to create dummy variables (1/0) out of (nominal scaled) variables, and add the indicator for West Germany (1, otherwise 0) to our data:

```
regionaldummies <- to.dummy(G.counties.gdp$regional)
# Creating dummy variables for West/East
# regionaldummies[,1] = East (1/0), regionaldummies[,2] = West (1/0)
G.counties.gdp$West <- regionaldummies[,2]
# Adding the dummy variable for West
```



(a) Absolute beta convergence (b) Sigma convergence

Figure 2: Regional convergence in Germany 2000-2014 (n = 402 counties)

Now, we test for conditional beta and sigma convergence, including the condition "West", again using the `rca()` function, but without plots and using the standard deviation (default setting) instead of the cv for sigma convergence. This time, we save the results in an object:

```
converg_results <- rca (G.counties.gdp$gdppc2000, 2000,
G.counties.gdp[55:68], 2014, conditions = G.counties.gdp[c(70)],
sigma.type = "trend")
# condition variable "West" in column 70
# Store results in "converg_results"
```

The output is:

```
Regional Beta and Sigma Convergence

Absolute Beta Convergence
Model coefficients (Estimation method: OLS)
             Estimate Std. Error t value                  Pr (>|t|)
Alpha       0.0954564  0.0099087   9.634 0.000000000000000000006845
Beta       -0.0071323  0.0009885  -7.215 0.000000000002719258225550
Lambda      0.0005113         NA      NA                          NA
Halflife 1355.7282963         NA      NA                          NA
Model summary
          Estimate F value df 1 df 2           Pr (>F)
R-Squared   0.1152   52.06    1  400 0.000000000002719

Conditional Beta Convergence
Model coefficients (Estimation method: OLS)
             Estimate Std. Error t value          Pr (>|t|)
Alpha       0.0754412  0.0102354   7.371 0.0000000000009872
Beta       -0.0047020  0.0010517  -4.471 0.0000101720129094
West       -0.0053559  0.0009745  -5.496 0.0000000693910790
Lambda      0.0003366         NA      NA                 NA
Halflife 2058.9555949         NA      NA                 NA
Model summary
          Estimate F value df 1 df 2                Pr (>F)
R-Squared   0.1774   43.04    2  399 0.00000000000000001192

Sigma convergence (Trend regression)
           Estimate Std. Error t value      Pr(>|t|)
Intercept  3.895236  0.3267817   11.92 0.00000002264
Time      -0.001764  0.0001628  -10.84 0.00000007041
Model summary
          Estimate F value df 1 df 2       Pr (>F)
R-Squared   0.9003   117.4    1   13 0.00000007041
```

In the `rca()` output, we can compare the results of absolute and conditional beta convergence. In the conditional model, the explained variance increases from $R^2 \approx 0.12$ to $R^2 \approx 0.18$, which indicates an increased explanatory power of the model due to the added condition variable. Both models are statistically significant, also the β values are negative and significant ($p < 0.001$ in both cases). The condition "West" is significant ($t \approx -5.50$, $p < 0.001$) and negative, which means that, on average, the GDP per capita in West German counties grew slower than in East Germany. These results *seem* to support the convergence hypothesis from growth theory, but one should not forget that e.g. political aspects (such as the German and/or EU regional policy) are not considered in this simple analysis.

As we have saved the invisible function output, we can access specific parts of our analysis, such as the regression data for the absolute convergence model:

```
converg_results$betaconv$regdata
# All results in list converg_results
# converg_results contains list betaconv (beta convergence results)
# betaconv contains data frame regdata (regression data)

    ln_initial   ln_growth
1       11.002  0.01997436
2       10.552  0.02980133
3       10.283  0.01794207
4       10.090  0.01763444
5       10.287  0.02361006
...
```

If we want to look at the single sigma values, we can address them via:

```
converg_results$sigmaconv$sigma.trend
# All results in list converg_results
# converg_results contains list sigmaconv (sigma convergence results)
# sigmaconv contains data frame sigma.trend (sigma values)

          years sigma.years
gdp1       2000      0.3646
gdppc2001  2001      0.3662
gdppc2002  2002      0.3618
gdppc2003  2003      0.3606
gdppc2004  2004      0.3592
...
```

4 Specialization of regions and spatial concentration of industries

4.1 Indicators of regional specialization and industry concentration

Specialization of regions or countries and the spatial concentration of industries or firms are phenomena linked to several research fields in regional economics and economic geography: Specialization is a key point in traditional theories of international trade with respect to comparative advantages (Ricardo, 1821) as well as in the generation of the "New Trade Theory" (introduced by Krugman 1979). Spatial clustering of firms or industries due to agglomeration economies is a perennial issue in all spatial economic fields. It especially reemerged in the context of the "New Economic Geography" (e.g. Krugman 1991; Fujita et al. 2001) as well as through the work of Porter (1990) regarding clusters. The common indicators are broadly discussed in Farhauer, Kröll (2014) or Nakamura, Morrison Paul (2009). For studies comparing some different indicators, see e.g. Goschin et al. (2009); Moga, Constantin (2011); Palan (2017).

When looking at the family of indicators of regional specialization and industry concentration, we have to distinguish between indicators for aggregate data, such as regional employment data, and those requiring individual firm data. The first group, compiled in Table 5, can be differentiated into indicators of specialization and indicators of spatial concentration. As both types of agglomeration are closely linked to each other, so are the

Table 5: Coefficients of regional specialization and industry concentration

Indicator	Specialization of region j	Spatial concentration of industry i
Hoover/Balassa	$LQ_{ij} = \frac{e_{ij}/e_i}{e_j/e} \equiv MRCA_{ij} = \frac{e_{ij}/e_j}{e_i/e}$	
	$\overline{LQ_j} = \frac{1}{I}\sum_{i=1}^{I} LQ_{ij}$	$\overline{LQ_i} = \frac{1}{J}\sum_{j=1}^{J} LQ_{ij}$
Extensions:		
O'Donoghue-Gleave	$SLQ_{ij} = \frac{LQ_{ij}-\overline{LQ_i}}{sd(LQ_i)}$	
Tian	$SLLQ_{ij} = \frac{\log(LQ_{ij})-\overline{\log(LQ_i)}}{sd(\log(LQ_i))}$	
Hoen-Oosterhaven	$ARCA_{ij} = \frac{e_{ij}}{e_j} - \frac{e_i}{e}$	
Hoover	$H_j = \frac{1}{2}[\sum_{i=1}^{I} \lvert\frac{e_{ij}}{e_j} - \frac{e_i}{e}\rvert]$ $0 \leq H_j \leq 1$	$H_i = \frac{1}{2}[\sum_{j=1}^{J} \lvert\frac{e_{ij}}{e_i} - \frac{e_j}{e}\rvert]$ $0 \leq H_i \leq 1$
Gini	$G_j = \frac{2}{I^2\bar{R}}\sum_{i=1}^{I} \lambda_i(R_i - \bar{R})$ $0 \leq G_j \leq 1$	$G_i = \frac{2}{J^2\bar{C}}\sum_{j=1}^{J} \lambda_j(C_j - \bar{C})$ $0 \leq G_i \leq 1$
	where: $R_i = \frac{e_{ij}/e_j}{e_i/e}$,	where: $C_j = \frac{e_{ij}/e_i}{e_j/e}$,
	$\bar{R} = \frac{1}{I}\sum_{i=1}^{I} R_i$ and	$\bar{C} = \frac{1}{J}\sum_{j=1}^{J} C_j$ and
	$\lambda_i = 1, ..., I$ $(\lambda_i < \lambda_{i+1})$	$\lambda_j = 1, ..., J$ $(\lambda_j < \lambda_{j+1})$
Krugman $(J = 2,\ I = 2)$	$K_{jl} = \sum_{i=1}^{I} \lvert s^s_{ij} - s^s_{il}\rvert$ $0 \leq K_{jl} \leq 2$	$K_{iu} = \sum_{j=1}^{J} \lvert s^c_{ij} - s^c_{uj}\rvert$ $0 \leq K_{iu} \leq 2$
	where: $s^s_{ij} = \frac{e_{ij}}{e_j}$ and $s^s_{il} = \frac{e_{il}}{e_l}$	where: $s^c_{ij} = \frac{e_{ij}}{e_i}$ and $s^c_{uj} = \frac{e_{uj}}{e_i}$
Extensions:		
Midelfart et al., Vogiatzoglou	$K_j = \sum_{i=1}^{I} \lvert s^s_{ij} - \bar{s}^s_{il}\rvert$ $0 \leq K_j \leq 2$	$K_i = \sum_{j=1}^{J} \lvert s^c_{ij} - \bar{s}^c_{uj}\rvert$ $0 \leq K_i \leq 2$
$(J > 2,\ I > 2)$	where: $s^s_{ij} = \frac{e_{ij}}{e_j}$ and	where: $s^c_{ij} = \frac{e_{ij}}{e_i}$ and
	$\bar{s}^s_{il} = \frac{1}{J-1}\sum_{i}^{J} s^s_{il}$, $l \neq j$	$\bar{s}^c_{uj} = \frac{1}{I-1}\sum_{u}^{I} s^c_{uj}$, $u \neq i$
Duranton-Puga	$RDI_j = \frac{1}{\sum_{i=1}^{I} \lvert\frac{s^s_{ij}}{s_i} - s_i\rvert}$ where: $s^s_{ij} = \frac{e_{ij}}{e_j}$ and $s_i = \frac{e_i}{e}$	
Litzenberger-Sternberg	$CI_{ij} = \frac{IS_{ij}ID_{ij}}{PS_{ij}}$ where $IS_{ij} = \frac{e_{ij}/a_j}{e_i/a}$, $ID_{ij} = \frac{e_{ij}/p_j}{e_i/p}$ and $PS_{ij} = \frac{e_{ij}/b_{ij}}{e_i/b_i}$	

Notes: e_{ij} and e_{il} equal the employment of industry i in regions j and l, respectively, e_i is the total employment in industry i, e_{uj} ist the employment of industry u in region j, e_j is the total employment in region j, e is the total employment in the whole economy, I is the number of industries, J is the number of regions, a_j is the area of region j, a is the total area in the whole economy, p_j is the population in region j, p is the total population, b_{ij} is the number of firms of industry i in region j and b_i is the number of firms in industry i.
Compiled from: Farhauer, Kröll (2014); Hoen, Oosterhaven (2006); Hoffmann et al. (2017); Nakamura, Morrison Paul (2009); O'Donoghue, Gleave (2004); Tian (2013); Schätzl (2000); Störmann (2009)

corresponding indicators. The empirical basis of all those measures is the employment e in industry i in region j, e_{ij}. This employment stock is compared to some reference, mostly including the total employment in region j, e_j, and/or the total employment in industry i, e_i, as well as the all-over employment e. The individual firm level indicators in Table 6 can be segmented into indicators for agglomeration of *one* industry due to localization economies and indicators for the coagglomeration of *different* industries due to urbanization economies.

Table 6: Coefficients of agglomeration and coagglomeration using individual firm data

Indicator	Agglomeration	Coagglomeration
Ellison-Glaeser	$\gamma_i = \frac{G_i - (1-\sum_{j=1}^{J} s_j^2) HHI_i}{(1-\sum_{j=1}^{J} s_j^2)(1-HHI_i)}$ where: $G_i = \sum_{j=1}^{J} (s_{ij}^c - s_j)^2$, $s_{ij}^c = \frac{e_{ij}}{e_i}$, $s_j = \frac{e_j}{e}$ and $HHI_i = \sum_{k=1}^{K} (\frac{e_{ik}}{e_i})^2$ z-standardization: $z_i = \frac{G_i - (1-\sum_{j=1}^{J} s_j^2) HHI_i}{\sqrt{var(G_i)}}$ where: $var(G_i) = 2\Big\{ HHI_i^2 \Big[\sum_{j=1}^{J} s_j^2$ $-2\sum_{j=1}^{J} s_j^3 + (\sum_{j=1}^{J} s_j^2)^2 \Big] -$ $\sum_{k=1}^{K} z_{ik}^4 \Big[\sum_{j=1}^{J} s_j^2 - 4\sum_{j=1}^{J} s_j^3$ $+3(\sum_{j=1}^{J} s_j^2)^2 \Big] \Big\}$	$\gamma^c = \frac{G/(1-\sum_{j=1}^{J} s_j^2) - HHI_U - \sum_{i=1}^{U} \gamma_i s_i^2 (1-HHI_i)}{1-\sum_{i=1}^{U} s_i^2}$ where: $G = \sum_{j=1}^{J} (x_j - s_j)^2$, $x_j = \sum_{i=1}^{U} \frac{e_{ij}}{e_i}$, $s_j = \frac{e_j}{e}$, $s_i = \frac{e_i}{e}$ and $HHI_U = \sum_{i=1}^{U} s_i^2 HHI_i$
Howard et al.		$CL_{ab} = \frac{\sum_{k=1}^{K_a} \sum_{l=1}^{K_b} C_{kl}}{K_a K_b}$ $XCL_{ab} = CL_{ab} - CL_{ab}^{RND}$ where: $C_{kl} = 1$ if firms k and l are located in the same region and $C_{kl} = 0$ otherwise

Notes: e_{ij} is the employment of industry i in region j, e_i is the total employment in industry i, e_j is the total employment in region j, e is the total employment in the whole economy, e_{ik} is the employment of firm k from industry i, k and l are indices for single firms, I is the number of industries, J is the number of regions, U is a subset of all I industries ($U \leq I$), K is the number of firms and K_a and K_b are the numbers of firms in industry a and b.

Compiled from: Farhauer, Kröll (2014); Howard et al. (2016); Nakamura, Morrison Paul (2009)

The most popular indicator is the Location Quotient (LQ), which is attributed to Hoover (1936) and mathematically equivalent to the Revealed Comparative Advantage (RCA) index, developed by Balassa (1965) in the context of international trade. The LQ is utilized in many studies (e.g. Bai et al. 2008; Kim 1995) as well as in the *OECD Territorial Reviews* (OECD, 2019). Following O'Donoghue, Gleave (2004) and Tian (2013), the original formulation can be extended: As the location quotient is not normalized, there is no cut-off value for defining a cluster, which leads to a standardization of the computed values via z-transformation. Hoen, Oosterhaven (2006) developed an additive alternative to the RCA index. The original LQ provides the main mathematical basis for several indicators developed later, such as the spatial Gini coefficients described below.

Some indicators which are known from the context of regional inequality (see Section 2) are also used for the analysis of agglomeration: A modification of the Gini coefficient is used for the spatial concentration of industries as well as regional specialization (e.g. Ceapraz 2008; Wieland, Fuchs 2018). As we can see in the calculation of R_i and C_j, respectively, the spatial Gini coefficient is based on the LQ. Another popular option for analyzing agglomeration is the Hoover coefficient, comparing the structure of an industry/a region to a reference structure of all industries/regions (e.g. Dixon, Freebairn 2009; Jiang et al. 2007). Both indicator types range between zero (no specialization/concentration) and one (total specialization/concentration). Also the Herfindahl-Hirschman index and its derivates are used to measure concentration, specialization and diversification (e.g. Duranton, Puga 2000; Goschin et al. 2009; Lehocký, Rusnák 2016).

Another type of specialization/concentration indicator was introduced by Krugman (1991), originally designed for comparing the specialization of *two* regions. An extension of this indicator was established by Midelfart-Knarvik et al. (2000) for the comparison of regional specialization/industry concentration with respect to the sum or mean of *all*

regions/industries (furthermore used e.g. by Haas, Südekum 2005; Vogiatzoglou 2006). Unlike the Gini- or Hoover-type measures, the Krugman coefficients range between zero (no specialization/concentration) and two (total specialization/concentration).

The cluster index developed by Litzenberger, Sternberg (2006) goes beyond employment data and includes additional information about the industry-specific firm size, population density and region size. It is composed of three parts: the relative industrial stock with respect to industry i and region j, IS_{ij}, the relative industrial density, ID_{ij}, and the relative firm size, PS_{ij}. All three components are modified location quotients. This is done to control for small and monostructural regions, which are identified as clusters otherwise (which is a problem in the original LQ). The cluster index CI_{ij} has a potential range from zero to infinity. This extended indicator is used e.g. by Hoffmann et al. (2017) for the German food processing industry.

The cluster indicators by Ellison, Glaeser (1997) compare the empirical distribution of firms to an arbitrary location pattern where agglomeration economies are absent (often referred to as a *dartboard approach*). Ellison, Glaeser (1997) differentiate between the clustering of firms from one industry (agglomeration) due to localization economies and the clustering of multiple industries (coagglomeration) due to urbanization economies. Their indices also take into account the industry-specific structure of the firms by including the Herfindahl-Hirschman index, HHI_i, for the employment concentration in industry i. This is the reason why individual firm-level data is required for the computation. The Herfindahl-Hirschman indicator is included to control the raw measures of spatial concentration, G_i and G, for firm employment concentration, which occurs especially when there are just a few firms with many employees. The Ellison-Glaeser (EG) index for agglomeration, γ_i, is designed for identifying the clustering of industry i, while the coagglomeration index, γ_c aims at the clustering of a set of U industries, where $U \leq I$. Values of γ equal to zero imply the absence of agglomeration economies, while values above zero indicate positive effects due to spatial clustering. When γ is negative, firm locations are less spatially concentrated than expected on condition of the dartboard approach, which indicates negative agglomeration economies. The EG index is used in several current regional economic studies (e.g. Dauth et al. 2015, 2018; Yamamura, Goto 2018).

In contrast, Howard et al. (2016) argue that agglomeration economies should not be analyzed regarding employment but the firms itself. Their colocation index, CL_{ab}, sums the colocation of K_i and K_q firms from two industries, i and q, controlling for all possible combinations. This colocation measure is compared to a counterfactual location structure constructed via bootstrapping; specifically the arithmetic mean of a number of (e.g. 50) random assignments of the regarded firms to the locations. The value of the resulting excess colocation index, XCL_{ab}, ranges between -1 and 1.

4.2 Application in REAT

4.2.1 REAT functions for regional specialization and industry concentration

Table 7 shows the REAT functions for agglomeration measures based on aggregate (employment) data. All functions require at least information about the employment in one or more regions j in one or more industries i, e_{ij}. The Herfindahl-Hirschman index (function `herf()`) for measuring regional diversity is not displayed as it is used exactly in the same way as described in Section 2, replacing x_i with e_{ij}.

Location quotients for one region and one or more industries are computed by the function `locq()`, including the option for an additive indicator instead of the multiplicative. When calculating the LQ for a set of J regions and I industries, one can use function `locq2()`, which is a kind of batch processing extension of `locq()`. As the dimension of the Litzenberger-Sternberg cluster index is the same as in the LQ (a single value for each combination of region j and industry i), the related functions `litzenberger()` and `litzenberger2()` work in the same way. When using `locq2()` or `litzenberger2()`, the user may choose the type of function output: either a `matrix` with I columns and J rows or a `data frame` with $I * J$ rows.

The Hoover-, Gini- and Krugman-type indicators require the same kind of input data. The `hoover()` function was already explained in Section 2, as it can be also

Table 7: REAT functions for regional specialization and industry concentration

Indicator	REAT function	Mandatory arguments	Optional arguments	Output
Hoover LQ/ Balassa RCA incl. extensions	locq()	vectors or single values of e_{ij} and e_i, single values of e_j and e	LQ method, plot	Single value or matrix with LQ_{ij}
	locq2()	vectors of e_{ij}, industry ID i and region ID j	normalization, output type, remove NAs	matrix or data frame with $I * J$ values of LQ_{ij}
Hoover specialization/ concentration	hoover() (see Section 2)	vectors of e_{ij} and reference vector e_i or e_j	remove NAs	value: H_j or H_i
Gini specialization	gini.spec()	vectors e_{ij} and e_i	plot LC	value: G_j, optional: LC plot
concentration	gini.conc()	vectors e_{ij} and e_j	plot LC	value: G_i, optional: LC plot
Krugman specialization	krugman.spec() (regions j and l)	vectors e_{ij} and e_{il}		value: K_{jl}
	krugman.conc2() (all J regions)	vector e_{ij} and matrix or data frame e_{il}		value: K_j
concentration	krugman.conc() (industries i and u)	vectors e_{ij} and e_{uj}		value: K_{iu}
	krugman.conc2() (all I industries)	vector e_{ij} and matrix or data frame e_{uj}		value: K_i
All at once: specialization	spec()	vectors of e_{ij}, industry ID i and region ID j	remove NAs	matrix with H_j, G_j and K_j (columns) for J regions (rows)
concentration	conc()	vectors of e_{ij}, industry ID i and region ID j	remove NAs	matrix with H_i, G_i and K_i (columns) for I industries (rows)
Duranton-Puga	durpug()	vectors e_{ij} and e_i		value: RDI_j
Litzenberger-Sternberg	litzenberger()	single values of e_ij, e_i, a_j, a, p_j, p, b_{ij} and b_i		value: CI_{ij}
	litzenberger2()	vectors of e_{ij}, industry ID i, region ID j, a_j, p_j and b_{ij}	output type, remove NAs	matrix or data frame with $I * J$ values of CI_{ij}

Source: own compilation.

used for measuring spatial concentration of industries or the specialization of regions with all-over employment vectors, e_i and e_j, respectively, as reference distributions. The spatial Gini coefficients are available through functions `gini.spec()` for regional specialization and `gini.conc()` for spatial concentration. The Krugman coefficients are divided into functions for the comparison of two regions/industries (`krugman.spec()` and `krugman.conc()`, respectively) and for applying all regions/industries as reference (`krugman.spec2()` and `krugman.conc2()`, respectively). The functions `spec()` and `conc()` are wrapper functions providing a convenient way to compute Hoover, Gini and Krugman coefficients of a given set of J regions and I industries at once, e.g. originating from official statistics on regional employment.

Table 8 shows the functions operating on the level of individual firm data. The Ellison-Glaeser (*EG*) indices are available through the functions `ellison.a()` (agglomeration index for industry i) and `ellison.a2()` (agglomeration indices for I industries) as well as `ellison.c()` (coagglomeration index for U industries) and `ellison.c2()` (coagglomeration indices for $I * I - I$ industry combinations). All functions require the firm size (e.g. no. of employees) for the k-th firm from industry i (`numeric vector`) and the

Table 8: REAT functions for agglomeration and coagglomeration using firm data

Indicator	REAT function	Mandatory arguments	Optional arguments	Output
Ellison-Glaeser agglomeration	ellison.a()	vectors of e_{ik}, e_j and region ID j		visible: value γ_i, invisible: matrix with γ_i, G_i, z_i, K_i and HHI_i
	ellison.a2()	vectors e_{ik}, industry ID i and region ID j		visible: values γ_i, invisible: matrix with γ_i, G_i, z_i, K_i and HHI_i, for I industries (rows)
coagglomeration	ellison.c()	vectors e_{ik}, industry ID i and region ID j	vectors e_j and U industries	value: γ^c
	ellison.c2()	vectors e_{ik}, industry ID i and region ID j	vector e_j	matrix with γ^c for $I * I - I$ industry combinations (rows)
Howard et al. colocation	howard.cl()	firm ID k, industry ID i, and region ID j, industries a and b		value: CL_{ab}
excess colocation	howard.xcl()	firm ID k, industry ID i and region ID j, industries a and b, no. of samples		value: XCL_{ab}
	howard.xcl2()	firm ID k, industry ID i and region ID j		matrix with XCL_{ab} for $I * I - I$ industry combinations (rows)

Source: own compilation.

region j the firm is located in. The functions incorporating more than one industry (all except for `ellison.a()`) require a `vector` containing the industry i. The data could e.g. be stored in a `data frame` with at least three columns (firm size, region, industry). Like some of the convergence functions (see Section 3), the EG agglomeration index functions in `REAT` also distinguish between a visible and an invisible output: `ellison.a()` and `ellison.a2()` show the value(s) auf γ_i but return an invisible `matrix` including the raw measure of concentration (G_i), the z-standardized results (z_i) and the related Herfindahl-Hirschman index for industry-specific firm concentration (HHI_i) as well as the number of firms in industry i (K_i).

The Howard-Newman-Tarp coagglomeration measure is distributed over the functions `howard.cl()` (calculation of the colocation index for one pair of industries a and b), `howard.xcl()` (calculation of the excess colocation index for industries a and b) and `howard.xcl2()` (calculation of the excess colocation index for $I * I - I$ combinations of I industries). As this cluster index works with firms instead of employment, we only need a `vector` containing the IDs of the firms k, the corresponding industry i and the region j where the firm is located. When calculating this measure for one pair of industries, the user must state the IDs of industries a and b. Note that calculation time for this index increases heavily with the number of firms and/or industries.

4.2.2 Application example 1: Regional specialization of Göttingen

We use the German classification of economic activities (WZ2008) on the level of 21 sections (A-U) for the classification of industries in the following examples (see Table 9).

Starting with a simple example, we analyze the regional specialization of Göttingen, a city with a population of about 134,000 in Niedersachsen, Germany. The example dataset `Goettingen`, which is included in `REAT`, contains the dependent employees in Göttingen and Germany for 2008 to 2017 in industries A to R (rows 2 to 16; row 1 contains the all-over employment). First, we load the data:

Table 9: Classification of economic activities in Germany, edition 2008 (WZ 2008)

WZ2008 Code	Title
A	Agriculture, forestry and fishing
B	Mining and quarrying
C	Manufacturing
D	Electricity, gas, steam and air conditioning supply
E	Water supply; sewerage, waste management and remediation activities
F	Construction
G	Wholesale and retail trade; repair of motor vehicles and motorcycles
H	Transportation and storage
I	Accommodation and food service activities
J	Information and communication
K	Financial and insurance activities
L	Real estate activities
M	Professional, scientific and technical activities
N	Administrative and support service activities
O	Public administration and defence; compulsory social security
P	Education
Q	Human health and social work activities
R	Arts, entertainment and recreation
S	Other service activities
T	Activities of households as employers; undifferentiated goods-and services-producing activities of households for own use
U	Activities of extraterritorial organisations and bodies

Source: own compilation based on Statistisches Bundesamt (2008).

```
data(Goettingen)
```

Using the `REAT` function `locq()`, we calculate a location quotient for Göttingen with respect to the manufacturing industry ("Verarbeitendes Gewerbe"), which is represented by letter C:

```
locq (Goettingen$Goettingen2017[4], Goettingen$Goettingen2017[1],
Goettingen$BRD2017[4], Goettingen$BRD2017[1])
# Industry: manufacturing (letter C) in row 4
# row 1 = all-over employment

[1] 0.5369
```

The output is simply the LQ value (LQ_{ij}, where i is manufacturing and j is Göttingen). We see that the LQ is very low, indicating that manufacturing is underrepresented in Göttingen as compared to Germany. Now, we calculate LQ values for all industries (A-R), including a simple plot (function argument `plot.results = TRUE`):

```
locq (Goettingen$Goettingen2017[2:16], Goettingen$Goettingen2017[1],
Goettingen$BRD2017[2:16], Goettingen$BRD2017[1],
industry.names = Goettingen$WZ2008_Code[2:16], plot.results = TRUE,
plot.title = "Location quotients for Göttingen 2017")
# all industries (rows 2-16 in the dataset)
```

The output is a `matrix` with one row for each industry:

```
Location quotients
I = 15 industries

            LQ
A   0.08407652
BDE 0.40085663
C   0.53687366
F   0.34366928
G   0.74603541
H   0.67117311
```

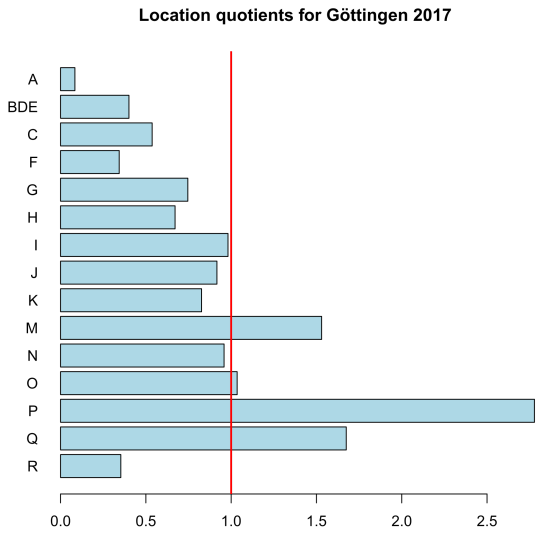


Figure 3: Location quotients for 15 industries in Göttingen

```
I   0.98141916
J   0.91654277
K   0.82650178
M   1.53027645
N   0.95843423
O   1.03509027
P   2.77790858
Q   1.67459967
R   0.35317012
```

The result is plotted in Figure 3. The function plots a vertical line at $LQ_{ij} = 1$ automatically. This is the (only) reference value for the LQ. It indicates a stock of the related industry equal to the whole economy. The highest LQ values can be found for the industries with letters P (education) and Q (health). This is because Göttingen is mainly characterized by a large university (about 30,000 students) with a university hospital with about 7,000 employees.

Now, we want to measure the specialization of Göttingen with a single indicator. First, we simply use the Herfindahl-Hirschman coefficient for both Göttingen and Germany using the function `herf()`:

```
herf(Goettingen$Goettingen2017[2:16])
[1] 0.127314

herf(Goettingen$BRD2017[2:16])
[1] 0.1104567
```

The *HHI* for Göttingen is slightly larger than for Germany, which indicates a higher specialization (or lower economic diversity) of the region. To combine this information in one indicator, we calculate the Hoover coefficient of specialization using the function `hoover()`, where the reference distribution is the German industry structure:

```
hoover(Goettingen$Goettingen2017[2:16], ref = Goettingen$BRD2017[2:16])
[1] 0.2254234
```

We finish our analysis of Göttingen's regional specialization by calculating both the Gini and the Krugman coefficient of regional specialization with the same data, using the `REAT` functions `gini.spec()` and `krugman.spec()`, respectively. Note that, here,

we use the Krugman coefficient to compare the industry structure of Göttingen to the structure of whole Germany (instead of another region within the country, for which this coefficient was originally formulated):

```
gini.spec(Goettingen$Goettingen2017[2:16], Goettingen$BRD2017[2:16])
[1] 0.359852

krugman.spec(Goettingen$Goettingen2017[2:16], Goettingen$BRD2017[2:16])
[1] 0.4508469
```

There seems to be some specialization in Göttingen, but, unfortunately, we do not have any real reference value to interpret the results.

4.2.3 Application example 2: Identifying clusters in Germany using aggregate data

In this example, we will compute indicators of regional specialization and industry concentration for a set of J regions and I industries at once. We load the included test dataset `G.regions.industries` containing employment and firms on the level of $I = 17$ industries (WZ2008 codes B-S) and $J = 16$ regions ("Bundesländer") in Germany:

```
data(G.regions.industries)
```

The number of employees in the column `emp_all` includes dependent employees and self-employed persons. The classification code of industries (see Table 9) can be found in column `ind_code`, while the region code (abbreviation of the region's official name) is in column `region_code`. First, we want to detect the spatial concentration of the 17 industries in Germany by calculating Hoover, Gini and Krugman coefficients for all industries at once, applying the `REAT` function `conc()` which is a wrapper function for the mentioned indicators. We save our output in the `matrix` object `conc_i`:

```
conc_i <- conc (e_ij = G.regions.industries$emp_all,
industry.id = G.regions.industries$ind_code,
region.id = G.regions.industries$region_code)
```

The output is:

```
Spatial concentration of industries
I = 17 industries, J = 16 regions

              H i        G i        K i
WZ08-B 0.22959050 0.42334831 0.45675385
WZ08-C 0.09933363 0.17047620 0.26813759
WZ08-D 0.07754576 0.12509360 0.16260016
WZ08-E 0.11972072 0.16742909 0.20369011
WZ08-F 0.07676634 0.15357575 0.16996098
WZ08-G 0.03034962 0.05471323 0.07977056
WZ08-H 0.06006957 0.11921850 0.10076748
WZ08-I 0.05177262 0.09939075 0.11450791
WZ08-J 0.10230712 0.22605802 0.24450967
WZ08-K 0.08982871 0.17610712 0.20565974
WZ08-L 0.09798632 0.16784764 0.17472656
WZ08-M 0.06490185 0.14760918 0.14931991
WZ08-N 0.06714816 0.08575299 0.09053327
WZ08-P 0.03019678 0.05053848 0.07043586
WZ08-Q 0.04679962 0.06170335 0.06406058
WZ08-R 0.09424708 0.16748405 0.17023603
WZ08-S 0.04507988 0.07246697 0.06441360
```

The function returns a `matrix` with 17 rows (one for each industry) and three columns: `H i` is the Hoover coefficient, `G i` is the Gini coefficient and `K i` is the Krugman coefficient for industry i. We cannot interpret or compare all of these results, but we may pick out some findings: The strongest spatial concentration is found with respect to mining

and quarrying (WZ08-B), no matter which indicator is regarded, which may be interpreted with "natural advantages" due to the spatial distribution of mineral resources in Germany. Services (such as retailing) as well as education and health are least concentrated, as these industries are bound to regional demand and/or their locations are regulated by policy and planning authorities.

At a first glance, the three indicators seem to produce similar results. Now, we want to test the similarity between Hoover, Gini and Krugman coefficients of concentration. As we saved our result `matrix`, we now calculate Pearson correlation coefficients (r) for each pair of indicators using the basic `R` function `cor()`, which is implemented in the `stats` package (included automatically in any `R` release). The function is applied to the three columns of `conc_i`, producing a $3 * 3$ correlation matrix:

```
cor(conc_i[,1:3])
          H i       G i       K i
H i 1.0000000 0.9676518 0.9527747
G i 0.9676518 1.0000000 0.9681770
K i 0.9527747 0.9681770 1.0000000
```

As we can see, each combination of the three indicators shows a strong positive correlation (H_i vs. G_i: $r \approx 0.97$, H_i vs. K_i: $r \approx 0.95$, G_i vs. K_i: $r \approx 0.97$). At least in this context, we may conclude that these indicators are interchangeable. However, we have to recognize that the analysis presented here is on a large-scale regional level (German "Bundesländer") and all of the mentioned indicators are affected by the *modifiable areal unit problem*, which means that the results depend on the aggregation unit in the analysis (see e.g. Dapena et al. 2016 for a discussion of this effect).

Now, we do exactly the same with respect to regional specialization of the 16 regions, using the same data. Analogously, we use the wrapper function `spec()` for calculating Hoover, Gini and Krugman coefficients of regional specialization, also saving the resulting `matrix`:

```
spec_j <- spec (e_ij = G.regions.industries$emp_all,
industry.id = G.regions.industries$ind_code,
region.id = G.regions.industries$region_code)
```

The output is:

```
Specialization of regions
I = 17 industries, J = 16 regions

          H j        G j        K j
BB 0.11530353 0.20632682 0.18555259
BE 0.17891265 0.29040841 0.34552331
BW 0.08024011 0.10300695 0.22675612
BY 0.05008135 0.07659148 0.16019603
HB 0.09502615 0.18563500 0.17467291
HE 0.05494422 0.12160142 0.11282696
HH 0.16413456 0.22616814 0.33190321
MV 0.13270849 0.18974606 0.22056868
NI 0.03772799 0.08237225 0.07972852
NW 0.02940091 0.05997505 0.07181569
RP 0.04793147 0.07432361 0.12036513
SH 0.08901907 0.11384295 0.15994524
SL 0.05726933 0.11921727 0.15071159
SN 0.05400855 0.10643512 0.10341280
ST 0.08821395 0.21120287 0.15280711
TH 0.08234046 0.13902924 0.17720208
```

The strongest specialization can be found in the city states Berlin (BE) and Hamburg (HH), while Niedersachsen (NI) and Nordrhein-Westfalen (NW) show the lowest values in all three indicators. As already mentioned in the concentration example, we have to remember the large-scale aggregation unit. If we used smaller scale units (e.g. counties like in Section 3.2.2), our results would surely be more differentiated. Again, we check the correlation between the indicators:

```
cor(spec_j[,1:3])
          H j       G j       K j
H j 1.0000000 0.9179127 0.9322604
G j 0.9179127 1.0000000 0.7907841
K j 0.9322604 0.7907841 1.0000000
```

Again, we find a strong positive correlation between the Hoover coefficient and both Gini and Krugman coefficient (H_j vs. G_j: $r \approx 0.92$, H_j vs. K_j: $r \approx 0.93$), while the third Pearson correlation coefficient is a little lower, but still showing the same direction (G_j vs. K_j: $r \approx 0.79$).

Now we check for clusters in a combination of a specific industry and a specific region. First, we calculate location quotients for the dataset `G.regions.industries` using the `REAT` function `locq2()`. Here, the optional function argument `LQ.norm` could be used for computing z-standardized location quotients according to O'Donoghue, Gleave (2004) (`LQ.norm = "OG"`) or z-standardized values of the natural-logged LQs according to Tian (2013) (`LQ.norm = "T"`). However, we produce the original LQs, since we need exactly the same columns as in the examples above:

```
locq2(e_ij = G.regions.industries$emp_all,
G.regions.industries$ind_code, G.regions.industries$region_code)
```

The output is a `matrix` with J rows and I columns:

```
Location quotients
I = 17 industries, J = 16 regions

              BB          BE        BW        BY        HB        HE
WZ08-B 2.5314363 0.04030901 0.6607950 0.8078054 0.0000000 0.3735773
WZ08-C 0.6857231 0.37224900 1.3968652 1.1902785 0.7863570 0.8580352
WZ08-D 1.1736475 0.46721079 1.0861988 0.8343784 0.9179718 0.9627955
WZ08-E 1.7945685 1.30128835 0.5896526 0.7137388 1.2393228 0.8532203
WZ08-F 1.5997778 0.77160121 0.9070096 1.0280409 0.6212923 0.8927681
WZ08-G 0.9550127 0.83133221 0.9492523 0.9879826 0.9013193 1.0006321
WZ08-H 1.3212794 0.87982228 0.8189666 0.8664163 1.7692815 1.2208728
WZ08-I 1.0379426 1.35561299 0.9132949 1.0390886 0.9904308 0.9571339
WZ08-J 0.5625876 1.78334039 1.0316114 1.1550764 1.0577107 1.1407078
WZ08-K 0.6529329 0.76630600 0.9329930 1.1058890 0.7825178 1.6710583
WZ08-L 1.1088846 2.13220960 0.7310014 0.8633894 1.3254723 1.1132939
WZ08-M 0.7366238 1.39880205 1.0337139 1.0265993 1.1202532 1.1457770
WZ08-N 1.2571301 1.24261162 0.7977054 0.8486971 1.2938161 1.0525912
WZ08-P 0.9052976 1.38842157 0.9649289 0.9252245 1.0563169 1.0085207
WZ08-Q 1.1540423 1.09329902 0.8695241 0.9079679 0.9544891 0.8980680
WZ08-R 1.0656945 2.55595102 0.8518192 0.8220540 1.3196613 0.8651451
WZ08-S 1.1409373 1.32596177 0.8626829 0.9092125 1.1396616 1.0528184
              HH        MV        NI        NW        RP        SH
WZ08-B 0.6029388 0.6235796 1.4987086 1.4595767 1.0371236 0.5078145
WZ08-C 0.4781934 0.6230156 0.9512438 0.9312325 1.0822678 0.7082513
WZ08-D 0.4332870 1.0118838 0.9932719 1.2139740 0.9349679 1.1248685
WZ08-E 1.1442005 1.5642257 1.0645497 1.0408356 0.9886860 1.0707585
WZ08-F 0.5432163 1.2716537 1.0969756 0.8735506 1.1134885 1.1043449
WZ08-G 1.0654315 0.9485377 1.0758977 1.0612190 1.0111274 1.2456100
WZ08-H 1.4958610 1.1243732 1.0409143 0.9961224 0.9633972 1.0112557
WZ08-I 1.0634066 1.7574637 1.0227196 0.8750205 1.1121264 1.2483966
WZ08-J 1.9266913 0.4751473 0.6716376 0.9830609 0.8122058 0.7496925
WZ08-K 1.5175078 0.5383900 0.9108456 1.0205798 0.8879292 0.8355178
WZ08-L 1.5871838 1.3034074 0.8040270 0.9928161 0.7774500 1.1980553
WZ08-M 1.6293913 0.6897571 0.8693026 1.0366589 0.7764558 0.7905498
WZ08-N 1.2530608 1.2484353 0.9675147 1.0659893 0.8026181 0.9727871
WZ08-P 0.9422739 0.9966228 1.0888054 0.9846351 1.0976178 0.9540262
WZ08-Q 0.8564604 1.2893168 1.0728412 1.0595648 1.0418460 1.1662290
WZ08-R 1.4914564 1.0500685 0.9204586 0.9611539 0.8498053 1.0418794
WZ08-S 0.8055128 1.1158184 0.9965451 1.0283571 1.1658852 1.1455178
```

```
              SL        SN        ST        TH
WZ08-B 0.2826284 1.2746172 2.4654331 0.7140637
WZ08-C 1.1752810 0.9867417 0.9297172 1.1849897
WZ08-D 1.1465539 1.0637093 1.2642787 0.8607578
WZ08-E 0.9555581 1.4457486 1.8251853 1.6042935
WZ08-F 0.9016858 1.3794286 1.4104724 1.3481005
WZ08-G 1.0370901 0.8787739 0.9172598 0.8661184
WZ08-H 0.8851047 1.0476688 1.2012430 0.8944907
WZ08-I 0.9111877 0.9496370 0.9020582 0.8644257
WZ08-J 0.7133587 0.7717704 0.4874344 0.6869177
WZ08-K 1.0082983 0.6620719 0.6133933 0.6316347
WZ08-L 0.7018816 1.1395422 1.0111694 0.8511896
WZ08-M 0.8060753 0.8459317 0.6627301 0.6814339
WZ08-N 1.0751749 1.1656467 1.2796548 1.1093251
WZ08-P 0.9147874 0.9658590 0.9798932 0.9710576
WZ08-Q 1.0760969 1.0475595 1.1401680 1.0628602
WZ08-R 0.7631263 1.1419135 0.8329295 0.8582919
WZ08-S 0.8840741 0.9774726 0.9257397 1.0923137
```

These $I * J = 17 * 16 = 272$ coefficients are too much information. Thus, we calculate them again using the optional argument `LQ.output = "df"`, which produces a `data frame` with $I * J$ rows and three columns (`j_region`: ID of region j, `i_industry`: ID of industry i and `LQ`: location quotient LQ_{ij}). We save the results in the object `lqs`:

```
lqs <- locq2(e_ij = G.regions.industries$emp_all,
G.regions.industries$ind_code, G.regions.industries$region_code,
LQ.output = "df")
```

As we forego an inspection of these singe values, the results are not displayed here. Instead, we only deal with the five highest LQs in our results (the "top five"). We sort the resulting `data frame` decreasing and take a look at the first five rows:

```
lqs_sort <- lqs[order(lqs$LQ, decreasing = TRUE),]
# Sort decreasing by size of LQ

lqs_sort[1:5,]

    j_region i_industry       LQ
33        BE     WZ08-R 2.555951
1         BB     WZ08-B 2.531436
239       ST     WZ08-B 2.465433
28        BE     WZ08-L 2.132210
111       HH     WZ08-J 1.926691
```

The highest LQ is found for the arts, entertainment, and recreation sector (WZ08-R) in the German capital Berlin. Note that this result is congruent with several studies about the "creative class", showing a large stock of "creative" employment in Berlin (e.g. Martin 2015). We also find a strong concentration of mining and quarrying in two Eastern regions, Brandenburg and Sachsen-Anhalt. Note that the LQ is a *relative* measure with respect to the total regional employment as well as the total industry-specific employment and the employment in the whole economy, not considering other aspects of industry or spatial structure.

These deficiencies should be overcome with the Litzenberger-Sternberg cluster index, also taking into account area, population and firm size. This additional data is also included in our current dataset (columns `area_sqkm`, `pop` and `firms`). The functions `litzenberger()` and `litzenberger2()` work equivalently to `locq()` and `locq2()`. To compute cluster indices for all $I * J$ combinations, we use the function `litzenberger2()`:

```
litzenberger2(G.regions.industries$emp_all,
G.regions.industries$ind_code, G.regions.industries$region_code,
G.regions.industries$area_sqkm, G.regions.industries$pop,
G.regions.industries$firms)
```

Like in `locq2()`, the default output is a `matrix` with I rows and J columns:

```
Litzenberger-Sternberg cluster indices
I = 17 industries, J = 16 regions

              BB           BE        BW        BY        HB        HE
WZ08-B 0.5736692   0.05611505 0.8041813 1.1073446       NaN 0.4745084
WZ08-C 0.1610679   3.24717820 2.6250805 1.2043415  5.119669 1.0603087
WZ08-D 0.2213627   1.37720778 1.7043505 1.4178208  4.172162 0.8541359
WZ08-E 0.8810260  10.14891585 0.8235517 0.6890213  6.705744 1.1427285
WZ08-F 0.7142888  11.36434353 1.2372108 0.9442221  3.498921 1.1225087
WZ08-G 0.2707787  12.10626625 1.3903532 0.9404677  7.136205 1.3361060
WZ08-H 0.4386878  13.26265081 1.0955747 0.7982074 22.656924 1.8358272
WZ08-I 0.2672336  26.60020727 1.4098657 0.9633029  8.481338 1.2880210
WZ08-J 0.1130326  59.24931837 1.4342037 1.2393579  8.683998 1.9024826
WZ08-K 0.1524825  10.28664194 1.4774980 1.1692461  6.650213 2.4739044
WZ08-L 0.2564814  56.65943460 0.9594093 0.8695636 11.839900 1.6099929
WZ08-M 0.1685895  39.30149403 1.5306799 1.0110472  9.410498 1.7302434
WZ08-N 0.4232166  26.91532975 1.0228326 0.7471872 10.796027 1.5265846
WZ08-P 0.2043023  25.30839556 1.3656409 0.9509028  7.322709 1.4627871
WZ08-Q 0.3445630  21.86956483 1.1770297 0.7873877  7.850886 1.2163624
WZ08-R 0.2450932 104.73565741 1.0839767 0.7821779 10.555369 1.0489672
WZ08-S 0.2891132  24.71310833 1.3435576 0.9594882  9.893688 1.5421903
              HH         MV        NI        NW        RP        SH
WZ08-B  2.319611 0.16000177 1.5004530 2.074735 1.1951266 0.3119091
WZ08-C  4.000104 0.11993714 0.5036838 2.010757 0.9646351 0.4008598
WZ08-D  1.679371 0.20954802 0.8848956 2.001708 0.6016715 1.2989105
WZ08-E 11.129156 0.44055556 0.6476797 1.915196 0.8616891 0.8101087
WZ08-F  5.526083 0.45291220 0.6276791 1.656384 0.8973162 0.8055662
WZ08-G 15.627090 0.22665565 0.6898359 2.377365 0.8115817 0.9008287
WZ08-H 44.371836 0.32192237 0.6420146 1.980491 0.7087341 0.7205961
WZ08-I 14.795885 0.59842705 0.6335126 1.731963 1.0572996 1.0361389
WZ08-J 59.720584 0.05895184 0.2905769 2.139796 0.5142953 0.4427180
WZ08-K 26.189623 0.12112900 0.5648050 1.953963 0.7104929 0.5758818
WZ08-L 33.175443 0.25167830 0.5228248 2.223854 0.5349641 0.8587588
WZ08-M 44.433527 0.11657637 0.4440959 2.297308 0.4961611 0.4456119
WZ08-N 23.255195 0.31231467 0.5271293 2.413351 0.5537510 0.7348068
WZ08-P 14.294075 0.24008982 0.7203953 2.022971 0.9704975 0.6937792
WZ08-Q 13.211918 0.38626491 0.6877487 2.260518 0.7770980 0.8426253
WZ08-R 44.256214 0.20066782 0.4508862 2.190752 0.5287290 0.6694273
WZ08-S 14.195156 0.27631480 0.5364848 1.941470 0.7996291 0.9238816
              SL        SN         ST        TH
WZ08-B 0.3673090 1.1179110 1.49064630 0.4562730
WZ08-C 1.8348713 1.0311722 0.32128528 0.7892360
WZ08-D 0.9643263 0.4534029 0.29391783 0.2086042
WZ08-E 1.9939461 1.5554125 1.09973945 1.1815726
WZ08-F 1.3245152 1.8167263 0.68075807 1.0313749
WZ08-G 1.7528078 0.7644695 0.31005922 0.4313306
WZ08-H 1.1099151 0.9297053 0.42973342 0.4881643
WZ08-I 1.7871163 0.7119567 0.29998391 0.4012910
WZ08-J 0.8984679 0.4927030 0.08046178 0.2087174
WZ08-K 1.5505928 0.5980588 0.21942805 0.3160432
WZ08-L 0.8170723 0.8226773 0.21044244 0.2886068
WZ08-M 1.0137151 0.6489868 0.15797219 0.2493458
WZ08-N 1.3940298 1.2492658 0.42285503 0.5935329
WZ08-P 1.2331390 0.7800531 0.31406013 0.4389675
WZ08-Q 1.8551266 1.0749000 0.48353914 0.5854787
WZ08-R 0.8477702 0.8666478 0.21562997 0.2919300
WZ08-S 1.6138955 0.8600519 0.34624522 0.5369766
```

Note that there is a value equal to `NaN`, which means "not a number", due to a division by zero; this is because there is no mining and quarrying (WZ08-B) in Bremen (HB). However, we take a look at the "top five" again:

```
lss <- litzenberger2(G.regions.industries$emp_all,
G.regions.industries$ind_code, G.regions.industries$region_code,
G.regions.industries$area_sqkm, G.regions.industries$pop,
G.regions.industries$firms, CI.output = "df")

lss_sort <- lss[order(lss$CI, decreasing = TRUE),]

lss_sort[1:5,]

    j_region i_industry        CI
33        BE     WZ08-R 104.73566
111       HH     WZ08-J  59.72058
26        BE     WZ08-J  59.24932
28        BE     WZ08-L  56.65943
114       HH     WZ08-M  44.43353
```

Again, we find the largest cluster value for the arts and entertainment sector in Berlin. Also the other four highest indicators are discovered in the largest city states Berlin and Hamburg, especially with respect to the information and communication industry (WZ08-J) and other knowledge-intensive services. Obviously, the results of the Litzenberger-Sternberg index differ in a noticeable way from those of the LQ, which can be attributed to the consideration of other spatial aspects, especially controlling for the size of the regions.

4.2.4 Application example 3: Identifying clusters using micro-data

In our last example about agglomerations, we use the Ellison-Glaeser indices and the Howard-Newman-Tarp colocation index, which both require individual firm data. As this kind of micro-data is sensitive and, of course, not available in official statistics, we have to use fictional data from the textbook by Farhauer, Kröll (2014).

At first, we compute the Ellison-Glaeser agglomeration index for one industry i, γ_i. We use the `REAT` function `ellison.a()`, which is designed for this purpose and requires three `vectors`: the size (employment) of firm k, e_{ik}, the IDs of the regions j each firm is located in, and the total regional employment, e_j. The numerical example in Farhauer, Kröll (2014), Table 14.11, contains ten firms in three regions (Wien, Linz, and Graz). We simply compile the data from the original table into separate `vectors`:

```
region <- c("Wien", "Wien", "Wien", "Wien", "Wien", "Linz",
"Linz", "Linz", "Linz", "Graz")
# regions (Austrian cities)
emp_firm <- c(200,650,12000,100,50,16000,13000,1500,1500,25000)
# employment of the ten firms
emp_region <- c(500000,400000,100000)
# employment of the three regions
```

Now, we apply `ellison.a()` to this data:

```
ellison.a (emp_firm, emp_region, region)
[1] 0.05990628
```

The EG agglomeration index of $\gamma_i \approx 0.06$, which is, by the way, the same result as in the textbook, indicates a stronger clustering than expected from a dartboard approach. Since this data is fictional, we refrain from interpreting this result.

The `REAT` package contains the dataset `FK2014_EGC`, which is compiled from the numerical example in Farhauer, Kröll (2014), Tables 14.14 to 14.17. There are $k = 42$ firms from $I = 4$ industries (clothing trade, forestry, textiles dyeing and textiles trade) in $J = 3$ regions (1, 2 and 3). We load this example data:

```
data(FK2014_EGC)
```

We compute γ_i for all industries in the dataset. This can be done with the function `ellison.a2()`, which requires `vectors` containing the size of firm k, the corresponding industry i, and region j. We save the results in the object `ega`:

```
ega <- ellison.a2 (FK2014_EGC$emp_firm, FK2014_EGC$industry,
FK2014_EGC$region)
```

Here, we see the output of the function:

```
Ellison-Glaeser Agglomeration Index
K = 42 firms, I = 4 industries, J = 3 regions

                    Gamma i
Clothing trade  -0.09379384
Forestry         0.16838003
Textiles dyeing -0.08012539
Textiles trade  -0.13040134
```

We see a strong clustering of the forestry industry, which is attributed to localization economies, but spatial avoidance in the three other industries. The visible output of `ellison.a2()` contains the γ_i values only, but the invisible `matrix` output also includes the other information referring to the *EG* agglomeration index:

```
ega
                    Gamma i         G i       z i K i      HHI i
Clothing trade  -0.09379384 0.017909653 -0.5025978  11 0.13124350
Forestry         0.16838003 0.088262934  1.3660878  13 0.09240553
Textiles dyeing -0.08012539 0.027764811 -0.3801644   9 0.14559983
Textiles trade  -0.13040134 0.002734966 -0.7663541   9 0.12208059
```

When looking at the forestry industry, we also see a high standardized value ($z_i \approx 1.37$) and a relatively low firm concentration ($HHI_i \approx 0.09$).

In the next step, we compute the *EG* coagglomeration index, γ^c, for the same data using the function `ellison.c()`. This function requires the same information as `ellison.a2()` plus the total employment in the regarded regions (column `emp_region`):

```
ellison.c (FK2014_EGC$emp_firm, FK2014_EGC$industry,
FK2014_EGC$region, FK2014_EGC$emp_region)

[1] 12.0729
```

Congruent with the calculation in Farhauer, Kröll (2014), the function returns $\gamma^c \approx 12.07$. This value is very large, which indicates urbanization economies in this fictional example.

If we want to analyze the coagglomeration of industry pairs instead, we may use the function `ellison.c2()`, which requires the same data:

```
ellison.c2 (FK2014_EGC$emp_firm, FK2014_EGC$industry,
FK2014_EGC$region, FK2014_EGC$emp_region)
```

The output is a `matrix` with $I * I - I$ rows (one for each industry pair, omitting the combination of the same industry i):

```
Ellison-Glaeser Co-Agglomeration Index
K = 42 firms, I = 4 industries, J = 3 regions

                                Gamma c
Forestry-Clothing trade        1.382257
Textiles dyeing-Clothing trade 2.465609
Textiles trade-Clothing trade  2.067766
Clothing trade-Forestry        1.382257
Textiles dyeing-Forestry       1.570292
Textiles trade-Forestry        1.336020
Clothing trade-Textiles dyeing 2.465609
Forestry-Textiles dyeing       1.570292
Textiles trade-Textiles dyeing 2.294259
Clothing trade-Textiles trade  2.067766
Forestry-Textiles trade        1.336020
Textiles dyeing-Textiles trade 2.294259
```

If we want to focus on firm numbers instead of employment size, we may compute the Howard-Newman-Tarp excess colocation index, which is included in `REAT` through the functions `howard.cl()` for one colocation index for one pair of industries, `howard.xcl()` for the corresponding excess colocation index and `howard.xcl2()` for all combinations of $I * I$ industries. Subsequent to the numerical example above, we calculate XCL_{ab} for all industry pairs in the dataset `FK2014_EGC`, where the firm ID of k is stored in the column `firm`:

```
howard.xcl2 (FK2014_EGC$firm, FK2014_EGC$industry,
FK2014_EGC$region)
# this takes some seconds
```

The output has the same structure as the output from `ellison.c2()`:

```
Howard-Newman-Tarp Excess Colocation Index
K = 42 firms, I = 4 industries, J = 3 regions

                                    XCL
Forestry-Clothing trade        0.01902098
Textiles dyeing-Clothing trade 0.02909091
Textiles trade-Clothing trade  0.02020202
Clothing trade-Forestry        0.02377622
Textiles dyeing-Forestry       0.03282051
Textiles trade-Forestry        0.03589744
Clothing trade-Textiles dyeing 0.02707071
Forestry-Textiles dyeing       0.02666667
Textiles trade-Textiles dyeing 0.02814815
Clothing trade-Textiles trade  0.02101010
Forestry-Textiles trade        0.01743590
Textiles dyeing-Textiles trade 0.03012346
```

We see that the index by Howard et al. (2016) is structured differently than the indicators presented above: Although they are based on exactly the same data, the value for forestry and clothing trade ($XCL \approx 0.019$) is *not* equal to the value for clothing trade and forestry ($XCL \approx 0.024$). Why? The XCL_{ab} is the difference between the colocation index, CL_{ab}, and the mean of a set of bootstrap samples, CL_{ab}^{RND} (see Table 6). These random samples are drawn again each time a XCL value is computed, consequently, also the XCL value changes.

5 Proximity and accessibility

5.1 Distance-based measures of accessibility and proximity using individual point-level data

In this chapter, we mix two different concepts of indicators, accessibility and spatial proximity (see Table 10), both frequently used especially in the context of GIS (geographic information systems). Both concepts are discussed together because they have two aspects in common: 1) they are based on the geographical distance between point locations, in particular, the distance between an origin point i or several origin points ($i = 1, ..., n$) and one or more destination points j ($j = 1, ..., m$), and 2) for the calculation, they require geocoded (with geographical coordinates) individual point data.

One popular indicator of accessibility is the Hansen accessibility, developed by Hansen (1959) in the context of land use theory. The basic idea is that "accessibility" equals the sum of opportunities outgoing from a specific origin i. These opportunities are spread over a set of m locations ($j = 1, ..., m$). The summation is weighted with the distance between i and the j-th location. This distance, no matter how measured (e.g. street distance, Euclidean distance, driving time) is assumed to be perceived in a nonlinear way, which is operationalized by a nonlinear distance decay function (a.k.a. distance impedance function or response function), e.g. power, exponential or logistic. A similar concept was introduced by Harris (1954) attempting to model the market potential of

Table 10: Accessibility and proximity indicators using point-level data

Indicator	Non-normalized	Normalized	
Accessibility/Market potential			
Harris	$M_j = \sum_{i=1}^{n} O_i d_{ij}^{-1}$ $0 \leq M_j \leq \infty$		
Hansen	$A_i = \sum_{j=1}^{m} O_j f(d_{ij})$ $i \neq j$ $0 \leq A_i \leq \infty$	$A_i^* = \frac{\sum_{j=1}^{m} O_j f(d_{ij})}{\sum_{j=1}^{m} O_j}$ $i \neq j$ $0 \leq A_i^* \leq 1$	
	where: $f(d_{ij}) = d_{ij}^{-\lambda}$ or $f(d_{ij}) = e^{-\lambda * d_{ij}}$ or $f(d_{ij}) = \frac{1}{1+e^{-\lambda_1+\lambda_2 d_{ij}}}$		
Proximity			
Count within buffer	$N_i = \sum_{i=1}^{n} I(d_{ij} \leq t)$ $i \neq j$		
Weighted count within buffer	$N_i^w = \sum_{i=1}^{n} I(d_{ij} \leq t) O_j$ $i \neq j$		
Ripley	$K_t = \frac{1}{\lambda} \sum_{i=1}^{n} \frac{I(d_{ij} \leq t)}{n}$ $i \neq j$	$L_t = \sqrt{\frac{K_t}{\pi}}$ $i \neq j$	$H_t = L_t - t$ $i \neq j$
	$E(K_t) = \pi t^2$	$E(L_t) = t$	$E(H_t) = 0$
	where: $\lambda = \frac{n}{A}$		

Notes: d_{ij} is the distance from origin location i ($i = 1, ..., n$) to destination location j ($j = 1, ..., m$), O_j is a variable quantifying the size of destination j, t is a maximum search radius and $I(d_{ij} \leq t)$ is the indicator function taking the value of $I = 1$ if $d_{ij} \leq t$, and $I = 0$ otherwise.
Compiled from: Kiskowski et al. (2009); Krider, Putler (2013); Peña Carrera (2002); Pooler (1987); Reggiani et al. (2011); Smith (2016)

locations. If we replace the inverse distance weighting in the Harris indicator with another type of distance weighting, we see that both concepts are mathematically equivalent. The only difference is that the Harris indicator is conceptualized from the supplier's perspective j (e.g. market potential of a retail store) and the Hansen accessibility takes the demand location i as a starting point (Pooler, 1987; Reggiani et al., 2011). As these indicators are dimensionless and range from zero to infinity, a normalization with a range from zero to one can be computed by weighting the results with the opportunities without distance correction.

This accessibility/potential concept can be used in the regional economic context e.g. to quantify the over-regional job potential (e.g. Wieland, Fuchs 2018) or the clustering of point locations of a specific type, such as retail stores (e.g. Larsson, Öner 2014). The most common application of these indicators may be the context of transport economics and transport geography (e.g. Albacete et al. 2017).

In the GIS context, spatial proximity can be measured using concentric zones within a radius of t (buffers) around point i, where the number of the j points within this radius is counted (Longley et al., 2005). A systematic analysis of spatial proximity or cluster patterns is possible using Ripley's K function (Ripley, 1976). It compares empirical point counts with expected values from a random spatial point process based on a Poisson distribution. Ripley's K computes empirical values for each distance band with a maximum distance of t, which can be compared to the expected value. A more comprehensible (and linear) interpretation is provided when normalizing the K function in the form of the L or H function. Also, confidence intervals for the expected values can be calculated by bootstrapping (Kiskowski et al., 2009; Smith, 2016). All of these measures are based on a simple indicator function, $I(d_{ij} \leq t)$, which takes the value of $I = 1$ if point j is within a distance of t from point i or not ($I = 0$). Originating from natural sciences, especially Ripley's K is frequently used when analyzing location patterns in spatial economic contexts, such as the clustering of retail stores (e.g. Krider,

Table 11: REAT functions for accessibility and proximity on the point level

Indicator	REAT function	Mandatory arguments	Optional arguments	Output
Distance matrix	dist.mat()	data frame(s) with start points i (ID, lat, lon) and end points j (ID, lat, lon), distance unit	$i \neq j$	data frame with from, to, from-to and distance d_{ij} (distance matrix)
Buffer	dist.buf()	data frame(s) with start points (ID, lat, lon) and end points (ID, lat, lon), max. distance t, distance unit	$i \neq j$, sum O_j at endpoints	list with distance matrix (data frame) and count table (data frame)
Hansen/ Harris	hansen()	distance matrix (data frame with start points i and end points j as well as distance d_{ij} and O_j), weighting functions, parameters λ and γ	distance constant, max. distance t, $i \neq j$	data frame with origins i and accessibility A_i
Ripley	ripley()	data frame with points (ID, lat, lon), total area A, max. distance t, number of distance intervals	local K values, confidence intervals no. of samples, significance level, plot (K, L or H)	visible: matrix with t, K_t, $E(K_t)$, $K_t - E(K_t)$, L_t and H_t for each distance interval, invisible: matrix (as described above) and optional: matrices with local K values and confidence intervals

Source: own compilation.

Putler 2013) or other types of firms assumed to be connected in a network (e.g. Espa et al. 2010).

5.2 Application in REAT

5.2.1 REAT functions for accessibility and proximity on the point level

Table 11 shows the `REAT` functions for the accessibility and proximity methods described above. A simple Euclidean distance matrix for georeferenced points (`data frame` with latitude and longitude) can be calculated using the function `dist.mat()`. The function `dist.buf()` computes a "count points within buffer", where also a weighting, O_j, can be summarized (e.g., if the destination points are cities of a given population, one could count the number of cities within 50 kilometers and their corresponding population). The latter function uses `dist.mat()`, thus, it is not necessary to create a distance matrix before.

The same is the case for the function `ripley()`, which calculates Ripley's K function for georeferenced data (`data frame` with lat/lon) and a given number of distance intervals up to a maximum distance of t. The differences between the empirical values, K_t, and the expected values, $E(K_t)$, as well as the normalizations (L_t and H_t) are calculated and returned automatically. Optionally, local K values for each distance interval and corresponding confidence intervals are computed. These confidence intervals are based on bootstrapping with a given number of samples (default: 100) on a given significance level (the default value is $\alpha = 0.05$, which leads to confidence intervals of a range from $\alpha/2 = 2.5\%$ to $1-\alpha/2 = 97.5\%$). Note that the plot of the K function (or, when desired, L or H function) provides a graphical and more intuitive interpretation of the analyzed point pattern, especially when including confidence intervals.

When calculating the Hansen accessibility (or the Harris market potential) with `hansen()`, a distance matrix including the opportunities, O_j, is required. This can be, of course, done with `dist.mat()` (if straight-line distances are sufficient), but also with any other software creating distance matrices (and any type of transport costs indicator). In `hansen()`, the user may choose between a power, exponential or logistic distance decay function. Optionally, the normalized Hansen accessibility is returned additionally.

5.2.2 Application example 1: Location analysis of medical practices

In the example in Section 2.2.2, we dealt with small-scale regional inequality in health care in South Lower Saxony, Germany. We have seen that e.g. psychotherapists are more spatially clustered than general practitioners (GPs). Returning to this topic, we want to use proximity and accessibility measures for determining the market potential (in the sense of the Harris model) of these health care locations. Obviously, there are different location patterns of general practitioners and psychotherapists. In the related study, there was evidence that psychotherapists are not just clustered but clustered within some districts of larger cities (Wieland, Dittrich, 2016). In the German health care planning system, the market potential of medical practices is the main determinant of the official authorization to be included into the allocation system of health insurance, while psychotherapists are assumed to need quite larger market areas than GPs (Kassenärztliche Bundesvereinigung, 2013). Consequently, our research hypothesis is that the population potential of psychotherapists is larger than that of general practitioners.

We use the same test data as in the mentioned example, containing the health locations (`GoettingenHealth1`) and the corresponding settlements (`GoettingenHealth2`). We load both `R` datasets:

```
data(GoettingenHealth1)
data(GoettingenHealth2)
```

Table `GoettingenHealth1` contains 617 locations, whose ID is stored in the column `location`. Columns `lat` and `lon` contain the latitude and longitude, respectively, while the corresponding location type can be found in column `type` (`phys_gen`: general practitioners, `psych`: psychotherapists, `pharm`: pharmacies). As the following applications may be time-consuming, we extract the general practitioners from `GoettingenHealth1` and draw a random sample of ten doctor's practices:

```
physgen <- GoettingenHealth1[GoettingenHealth1$type == "phys_gen",]
# general practitioners: column "type" is equal to "phys_gen"
physgen_sample <- physgen[sample(nrow(physgen),10),]
# random sampling of ten general practitioners
```

Now, we want to summarize the population potential of these health locations in a 1,000 meters buffer. We apply the function `dist.buf()` to the sample data `physgen_sample` and sum up the local population of the districts within this distance (column `pop` in `GoettingenHealth2`):

```
physgen_pot <- dist.buf (physgen_sample, "location", "lat", "lon",
GoettingenHealth2, "district", "lat", "lon", bufdist = 1000,
ep_sum = "pop")
# counting all districts within a radius of 1000 meters
# and summing the corresponding population
```

We calculate the arithmetic mean of all ten potentials:

```
mean2(physgen_pot$count_table$sum_pop)
[1] 8027.7
```

On average, the ten GP practices have a population potential of about 8,028 inhabitants. One problem related to the buffer technique is the lack of distance weighting: All origin points up to a given distance are included completely, while all points above 1,000 meters are ignored. Thus, we repeat estimating the population potential using the Hansen accessibility. At first, we need an origin-destination matrix (distance matrix) from the origin points to the sampled GP locations. We use the function `dist.mat()` and merge the returned distance matrix with the population values from `GoettingenHealth2`:

```
physgen_od <- dist.mat(GoettingenHealth2, "district", "lat", "lon",
physgen_sample, "location", "lat", "lon")
# creating OD matrix from all districts to the
# sampled general practitioners

physgen_od <- merge (physgen_od, GoettingenHealth2,
by.x = "from", by.y = "district")
# merging with GoettingenHealth2 to include the
# population values of the districts
```

Then, we use the function `hansen()` to calculate the Hansen accessibility (used in the sense of the Harris market potential model) for each GP location in `physgen_od`.

The required columns in this dataset are the IDs of the GP locations (`to`), the IDs of the districts (`from`) and the population of the districts (`pop`) as well as the distances calculated above (`distance`). Finally, we have to set a distance weighting (which has an important influence in all types of spatial interaction models like this). For this purpose, we fall back on the results of a study by Fülöp et al. (2011): Based on empirical patient's choice of doctor, they estimated distance decay functions in spatial interaction models (Huff model) for several types of physicians. For GPs, an exponential distance decay function with $\lambda = -0.28$ was found to fit the empirical data best. To set a distance decay function type and the related weighting(s), the function arguments `dtype` and `lambda` must be used. We save the results under the name `physgen_hansen`:

```
physgen_hansen <- hansen (physgen_od, "to", "from", "pop",
"distance", dtype = "exp", lambda = -0.28)
# calculating Hansen accessibility for the ten
# sampled general practitioners
```

The output of the `hansen()` function is:

```
Hansen Accessibility

J = 420 locations with mean attractivity = 1138.486
I = 10 origins with mean transport costs = 28.07581
Attractivity weighting (pow) with Gamma = 1
Distance weighting (exp) with Lambda = -0.28

     to accessibility
1  1103     24267.054
2  1171     17629.564
3  1206      9581.732
4  1220      9213.407
5   197     10023.854
6   301      6489.571
7   600     69676.232
8   755     66921.123
9   966     13154.921
10  974      3666.171
```

Again, we calculate the arithmetic mean of the distance-weighted market potentials:

```
mean2(physgen_hansen$accessibility)
[1] 23062.36
```

The average population potential of the ten GPs is equal to 23,063 inhabitants.

As we want to compare the market potential of GPs and psychotherapists, we repeat the same analysis for them, now in the "fast mode", leaving out most comments, as the functions and commands are exactly the same as above, only applied to psychotherapists.

```
psychgen <- GoettingenHealth1[GoettingenHealth1$type == "psych",]

psych_sample <- psychgen[sample(nrow(psychgen),10),]

psych_pot <- dist.buf (psych_sample, "location", "lat", "lon",
GoettingenHealth2, "district", "lat", "lon", bufdist = 1000,
ep_sum = "pop")

mean2(psych_pot$count_table$sum_pop)
[1] 12245.88
```

The calculation of Hansen accessibility is different from the one for GPs with respect to the assumed distance reaction of the (potential) clients: For psychotherapists, Fülöp et al. (2011) found a distance impedance which is considerably smaller than for GPs (and any other type of doctor), resulting in a weighting parameter of $\lambda = -0.11$ in the exponential decay function:

```
psych_od <- dist.mat(GoettingenHealth2, "district", "lat", "lon",
psych_sample, "location", "lat", "lon")

psych_od <- merge (psych_od, GoettingenHealth2,
by.x = "from", by.y = "district")

psych_hansen <- hansen (psych_od, "to", "from", "pop",
"distance", dtype = "exp", lambda = -0.11)


Hansen Accessibility

J = 420 locations with mean attractivity = 1138.486
I = 10 origins with mean transport costs = 25.56756
Attractivity weighting (pow) with Gamma = 1
Distance weighting (exp) with Lambda = -0.11

     to accessibility
1  1031      43415.63
2  1213      39226.26
3   179      33491.41
4   313      51228.41
5   506     147887.43
6   786     147969.39
7   791     147971.80
8   811     148021.51
9   872     147475.57
10  922      42424.51

mean2(psych_hansen$accessibility)
[1] 94911.19
```

We see that the average population potential of the sampled psychotherapists on the 1,000 meters buffer level is equal to 12,246 inhabitants, which is about one third more than for GPs. The Hansen/Harris market potential of psychotherapists of about 94,911 persons is a fourfold increase compared to the GPs. We have to remember that the last result is not only a matter of location but also due to a lower assumed distance decay. However, the population potential of the sampled psychotherapists is obviously higher than the potential of the GPs, which can be attributed to a different location pattern, where psychotherapists are more clustered within larger city districts.

5.2.3 Application example 2: Clustering of health service providers

We stick to the example of health care locations. As we have found different degrees of regional inequality with respect to suppliers (Section 2.2.2) and of market potentials

(Section 5.2.2), we now analyze the clustering patterns of health service providers. In South Lower Saxony there is nearly the same number of psychotherapists (118) and pharmacies (120), but we should not expect their location patterns to be similar or even equal. Following the results above, we hypothesize that psychotherapists are more spatially clustered than pharmacies (as we already know about clustering with respect to districts in the former case and we can expect an avoidance tendency in the latter case due to a high degree of substitutability).

For this analysis, we compute Ripley's K with the `REAT` function `ripley()`. Before going on, we have to prepare two things: First, we load the required dataset. Then, we must calculate the total area of the study area manually (here: in square meters).

```
data (GoettingenHealth1)

area_goe <- 1753000000
# area of Landkreis Goettingen (sqm)
area_nom <- 1267000000
# area of Landkreis Northeim (sqm)
area_gn <- area_goe+area_nom
```

Now, we compute Ripley's K for the pharmacies only, which means processing only those locations in `GoettingenHealth1` which are pharmacies (`type == "pharm"`). We set our maximum search radius equal to $t = 30000$ (function argument `t.max`), divided into 300 distance intervals (`t.sep`), resulting in distance steps of 100 meters. As we want to check for a significant deviation from a random spatial pattern, we instruct the function to construct confidence intervals (`ci.boot = TRUE`) using the default settings ($\alpha = 0.05$, 100 bootstrapping samples). We also plot the results (default function argument: `K.plot = TRUE`) to inspect our results graphically. Here, we plot K_t, which is also the default setting (if the user wants to plot L_t or H_t instead, the function argument `Kplot.func` has to be changed to "`L`" or "`H`", respectively):

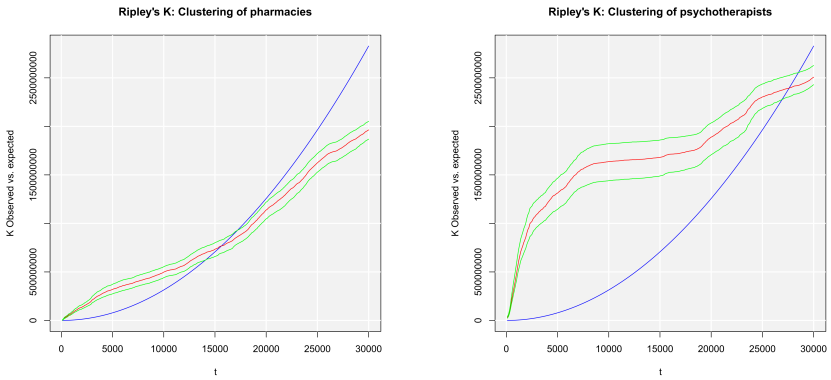
```
ripley(GoettingenHealth1[GoettingenHealth1$type == "pharm",],
"location", "lat", "lon", area = area_gn, t.max = 30000, t.sep = 300,
K.local = TRUE, ci.boot = TRUE, ci.alpha = 0.05, ciboot.samples = 100,
plot.title = "Ripley's K: Clustering of pharmacies")
```

The output is a `matrix` with six columns and one row for each distance interval. Thus, we skip the full output here:

```
Ripley's K
n = 120 points

     t <=       K t exp        K t  Kt-Kt exp       L t         H t
1     100      31415.93    3355556    3324140  1033.492   933.49238
2     200     125663.71   12583333   12457670  2001.349  1801.34940
3     300     282743.34   25586111   25303368  2853.824  2553.82412
4     400     502654.82   32297222   31794567  3206.326  2806.32580
5     500     785398.16   39008333   38222935  3523.739  3023.73923
...
```

We repeat the computation of Ripley's K for the psychotherapists:

```
ripley(GoettingenHealth1[GoettingenHealth1$type == "psych",],
"location", "lat", "lon", area = area_gn, t.max = 30000, t.sep = 300,
K.local = TRUE, ci.boot = TRUE, ci.alpha = 0.05, ciboot.samples = 100,
plot.title = "Ripley's K: Clustering of psychotherapists")
```



(a) Pharmacies (n=120) (b) Psychotherapists (n=118)

Figure 4: Plots of the Ripley-K function with confidence intervals

The output is (also truncated):

```
Ripley's K
n = 118 points

     t <=       K t exp         K t    Kt-Kt exp       L t          H t
1     100      31415.93    30798621   30767205.2  3131.055  3031.055025
2     200     125663.71    48583740   48458076.6  3932.516  3732.516350
3     300     282743.34    80249928   79967184.8  5054.141  4754.141421
4     400     502654.82   132737719  132235064.2  6500.133  6100.132940
5     500     785398.16   202143062  201357664.2  8021.480  7521.479612
...
```

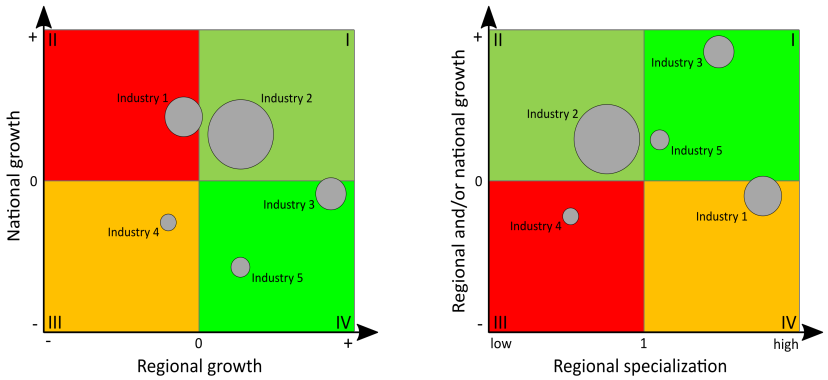
The graphical output is shown in Figures 4a (pharmacies) and 4b (psychotherapists), respectively. The expected value of K_t is plotted as blue line, while the empirical K_t values are red and the corresponding confidence intervals are colored in green (These colors are the default values in `ripley()` and can be changed by the function arguments `lcol.exp` and `lcol.emp`, respectively). As we have nearly the same number of points in both cases within the same field area, a direct comparison seems reasonable. Obviously, both types of locations show a significant spatial clustering: Also the pharmacies are more clustered than expected on condition of complete spatial randomness up to a distance of about 15,000 meters. We have to remember that also the population is already clustered (see Section 2.2.2) and the spatial distribution of pharmacies may follow this pattern. However, the clustering of psychotherapists exceeds this level enormously, especially within smaller distances up to about 8,000 meters. In conclusion, the psychotherapists are more spatially clustered than pharmacies.

6 Analysis and prognosis of regional growth

6.1 Tools and models concerning regional growth

6.1.1 Analyzing regional growth: shift-share analysis and portfolio matrix

Aspects of regional growth have already been discussed in the context of regional convergence in Section 3. The identification of clusters was the topic of Section 4. Combining some aspects of both, this section presents a collection of tools and models concerning regional growth with respect to industries. Like the indicators in Section 4, these techniques are of high significance especially in the context of local economic policy and municipal business promotion activities, aiming at e.g. strengthening a city's or region's competitiveness, defining its profile or increasing the number of jobs (Dinc, 2015; Nischwitz et al., 2017). Inspired by Farhauer, Kröll (2014) and congruent with the mathematical formulations in Section 4, we calculate on the basis of local/regional employment, e_{ij},



(a) Regional growth portfolio

(b) Regional growth & specialization portfolio

Figure 5: Regional economic portfolio matrix

which is the number of employees of industry i in region j. Its growth from time t to time $t + y$ can be operationalized as an absolute value ($\Delta e_{ij} = e_{ij_{t+y}} - e_{ij_t}$) or as a relative growth ($\Delta e_{ij}^{rg} = e_{ij_{t+y}}/e_{ij_t}$) or as a (percentage) growth rate ($\Delta e_{ij}^{gr} = e_{ij_{t+y}}/e_{ij_t} - 1$).

The first technique described is the regional economic portfolio matrix, originating from the portfolio matrix in marketing, developed by the Boston Consulting Group (BCG) for the identification of growing and declining business fields of firms (Henderson, 1973). However, this technique can also be applied to several regional economic contexts (Baker et al., 2002; Howard, 2007). Here, we present a portfolio matrix which compares the growth in *one* region with the growth in a superordinate reference region (e.g. whole economy). When using the matrix in this way, it is a plot of the growth rate with respect to industry i in the region (Δe_{ij}^{gr}) on the x axis and the corresponding growth in the reference region (Δe_{i}^{gr}) on the y axis (see Figure 5a). The size of the points for each industry may be the total size of employment in the region (e_{ij}) to reflect the absolute relevance of the i-th industry. The plot is segmented into four quadrants, differentiated with respect to positive or negative growth rates. As implied by the colors of the quadrants, they can be interpreted as follows: Quadrant I (top right) contains the industries growing in both the region and the whole economy (or any other reference region). Quadrant II (top left) shows all industries growing in the whole economy but shrinking in the regarded region, which may indicate significant locational handicaps. Quadrant III (bottom left) includes all industries shrinking in the region as well as in the whole economy. Quadrant IV (bottom right) shows the special case of "star" industries, indicating that these industries grow in the regarded region while shrinking in the whole economy. Note that this segmentation (and the corresponding interpretation) differs from the original BCG matrix.

Another variant of the portfolio matrix, which was developed in the context of designing the `REAT` package, is shown in Figure 5b. Combining the aspects of regional specialization (see Section 4) and regional growth, we can plot the location quotient as an indicator of local specialization on the x axis, while plotting an industry-specific growth indicator on the y axis. For identifying "growing" industries, there are at least three options of operationalization: We can plot the industry-specific regional growth rate (Δe_{ij}^{gr}) on the y axis (which is on the x axis in the portfolio matrix in Figure 5a) or the industry-specific national rate (Δe_{i}^{gr}) or, if we want to show regional growth in relation to national growth, the quotient of industry-specific regional and national growth rates ($\Delta e_{ij}^{gr}/\Delta e_{i}^{gr}$). In quadrant I, we see now all industries overrepresented in the region (in terms of the location quotient) as well as growing on the regional/national level. Quadrant II shows all industries underrepresented in the region but growing as well. In quadrants III and IV, we can identify all industries with negative growth rates, which are underrepresented or overrepresented, respectively.

Table 12: Shift-share analysis: Dunn and Gerfin type

Component	Dunn-type (absolute)	Gerfin-type (index)
	$\Delta e_j = e_{j_{t+y}} - e_{j_t} = n_{j_{t,t+y}} + m_{j_{t,t+y}} + c_{j_{t,t+y}}$	
Net total shift	$t_{t+y} = e_{j_{t+y}} - e_{j_t} - n_{j_{t,t+y}} = m_{j_{t,t+y}} + c_{j_{t,t+y}}$	$t_{t+y} = m_{j_{t,t+y}} c_{j_{t,t+y}} = \frac{\frac{e_{j_{t+y}}}{e_{j_t}}}{\frac{e_{t+y}}{e_t}}$
static (two time periods t and $t+y$)		
National share	$n_{j_{t,t+y}} = e_{j_t} \frac{e_{t+y}}{e_t} - e_{j_t}$	$n_{j_{t,t+y}} = 1$ (omitted)
Industrial mix	$m_{j_{t,t+y}} = \sum_{i=1}^{I} e_{ij_t} \frac{e_{i_{t+y}}}{e_i} - e_{j_t} \frac{e_{t+y}}{e_t}$	$m_{j_{t,t+y}} = \frac{\sum_{i=1}^{I} e_{ij_t} \frac{e_{i_{t+y}}}{e_i}}{e_{j_t} \frac{e_{t+y}}{e_t}}$
Regional share	$c_{j_{t,t+y}} = \sum_{i=1}^{I} e_{ij_t} (\frac{e_{ij_{t+y}}}{e_{ij_t}} - \frac{e_{i_{t+y}}}{e_{i_t}})$	$c_{j_{t,t+y}} = \frac{\frac{e_{ij_{t+y}}}{e_{ij_t}}}{\sum_{i=1}^{I} e_{ij_t} \frac{e_{i_{t+y}}}{e_i}}$
dynamic (T time periods, while $T > 2$)		
National share	$n_{j_{t,T}} = \sum_{t=1}^{T} e_{j_t} \frac{e_{t+1}}{e_t} - e_{j_t}$	
Industrial mix	$m_{j_{t,T}} = \sum_{t=1}^{T} \sum_{i=1}^{I} e_{ij_t} \frac{e_{i_{t+1}}}{e_i} - e_{j_t} \frac{e_{t+1}}{e_t}$	
Regional share	$c_{j_{t,T}} = \sum_{t=1}^{T} \sum_{i=1}^{I} e_{ij_t} (\frac{e_{ij_{t+1}}}{e_{ij_t}} - \frac{e_{i_{t+1}}}{e_{i_t}})$	
industry-specific		
National share	$n^i_{j_{t,t+y}} = e_{ij_t} \frac{e_{i_{t+y}}}{e_{i_t}} - e_{ij_t}$	
Regional share	$c^i_{j_{t,T}} = e_{ij_t} (\frac{e_{ij_{t+y}}}{e_{ij_t}} - \frac{e_{i_{t+y}}}{e_{i_t}})$	
prognosis for time period z		
Employment		$\Delta e_{ij_{t+z}} = e_{ij_{t+y}} (\frac{e^P_{i_{t+z}}}{e_{i_{t+y}}}) c_{j,t+y}$

Notes: e_{j_t} is the employment in region j at time t, e_{ij_t} is the employment of industry i in region j at time t, e_t is the total employment in the whole economy at time t, e_{i_t} is the total employment in industry i, y and z are numbers of time periods added to t ($z > y$), T is the number of regarded time periods and I is the number of industries.
Compiled from: Farhauer, Kröll (2014); Haynes, Parajuli (2014); Schätzl (2000); Schönebeck (1996); Spiekermann, Wegener (2008); Barff, Knight (1988)

A well-established model of regional growth is the shift-share analysis, which is, although developed independently from the portfolio matrix, closely linked to the concept presented above. The original shift-share analysis was introduced by Dunn Jr. (1960) and given a theoretical foundation by Casler (1989). Parallelly and independently, Gerfin (1964) developed a variant of shift-share analysis, which is more popular in the German-speaking regional economic science. Both concepts have been extended in several ways. Table 12 shows the basics of shift-share analysis with respect to "Dunn" and "Gerfin" type. As there are several ways of formulating the shift-share formulae and calling the particular elements of the shift-share analysis, the description here is based on the mathematical formulations in Farhauer, Kröll (2014) and the terms used in Haynes, Parajuli (2014).

The basic idea of shift-share analysis is the decomposition of regional growth into components, recognizing that single economic regions are embedded into and influenced by a larger regional system, normally the whole economy, just called "the nation" hereinafter: The (employment or e.g. gross value added) growth of industry i in region j from time t to time $t+y$ can be attributed to 1) a national trend, which means the economic climate in the whole system of regions, 2) the all-over growth or decline of the regarded industries and 3) the industry-specific performance of the region, which is linked to locational advantages or disadvantages. The first component is called *national share* and reflects the growth in region j that *would* have occurred if region j *would* have developed exactly as the nation. The second component is the *industrial mix*, represent-

ing the aggregated industry-specific growth in region j if the regarded industries *would* have developed like in the whole economy, adjusted by the national effect. The third component is the *regional share*, which is the "residuum" of the first two components; this share of growth is attributed to locational advantages (or disadvantages), showing the regional growth adjusted by national and industry effects (Farhauer, Kröll, 2014; Haynes, Parajuli, 2014).

The Dunn-type models deal with absolute growth (Δe_{ij} or Δe_j), which is the sum of all shift-share components, and a *net total shift*, which is the sum of the industrial mix and the regional share (as these components are region-specific). Thus, this technique is also called the "difference method". The Gerfin-type approaches express growth in terms of indices, while the net total shift for region j is the result of a multiplication of the industrial mix index and the regional share index, resulting in the alternative denomination "index method" (Schätzl, 2000).

Several extensions have been developed for the Dunn-type shift-share analysis (Haynes, Parajuli, 2014). One regular application calculates a shift-share analysis for each industry i in region j (instead of computing components for the whole region), while skipping the industrial mix effect. A main contribution was the dynamic shift-share analysis by Barff, Knight (1988). It extended the Dunn model by dealing with growth within a longitudinal cut of T years. Other extensions of the Dunn-type technique provide a deeper differentiation of the three components, which are regarded as correlated (e.g. Arcelus 1984; Esteban-Marquillas 1972).

6.1.2 Commercial area prognosis

Also developed independently in the context of German urban planning, a commercial area prognosis deals with an absolute (assumed) employment growth (Δe_{ij}) over T years, which is used to forecast the required commercial area within a city or region j up to time T. Note that "commercial area" represents the type of urban area which is used by specific economic activities, especially industrial plants, and/or designated for this purpose in municipal land-use plans. This technique is a demand-side approach, as it derives the required commercial area from the (expected) demand for it (Bonny, Kahnert, 2005). See Table 13 for the calculation of two types of commercial area prognosis based on employment growth.

The basic model called *GIFPRO* (German abbreviation for "Gewerbe- und Industrieflächenbedarfsprognose", roughly translated: prognosis of future demand of commercial area) was developed by Stark et al. (1981). The usual procedure is to estimate – starting from the current employment – the future industry-specific employment in region j. This number of employees is weighted by the industry-specific shares of workers usually located in commercial areas and multiplied by a resettlement rate (sq_{ij} percent of employees from industry i are resettled in one time period) and a relocation rate (rq_{ij} percent of employees from industry i are relocated in one time period) as well as a reutilization rate (ru_{ij} percent of employees from industry i will be located at reused commercial area). This "commercial area-relevant" employment is weighted with an areal index, a_{ij} (commercial area per employee), to compute the commercial area for industry i in region j for one time period t. The expected commercial area is summed over all I industries (A_{j_t}) and, finally, over all T years and I industries (A_{j_T}) (Bonny, Kahnert, 2005; Planungsgruppe MWM, 2009).

A significant extension was developed in the context of establishing a land-use plan for Dresden: The *TBS-GIFPRO* (German abbreviation for "Trendbasierte und standortspezifische Gewerbe- und Industrieflächenbedarfsprognose", roughly translated: trend-based and location-specific prognosis of future demand of commercial area) technique (Deutsches Institut für Urbanistik GmbH, Spath + Nagel (GbR), 2010). It includes a stochastic approach for forecasting employment as well as other region-specific data. The employment prognosis is done using a trend regression model (employment against time) based on past empirical employment data for region j (mostly from official employment statistics) which are used for forecasting future employment. For each i industry, a single regression model is estimated, where the function type is not pre-defined but chosen e.g. based on the explained variance (R^2) and/or plausibility considerations.

Table 13: Commercial area prognosis

Prognosis	GIFPRO	TBS-GIFPRO
Employment	$e^A_{ij_t} = \left[\left(e_{ij_{t0}} \frac{a_i}{100} \frac{sq_{ij}}{100}\right) + \left(e_{ij_{t0}} \frac{a_i}{100} \frac{rq_{ij}}{100}\right)\right] - \left(e_{ij_{t0}} \frac{ru_{ij}}{100}\right)$	$e^A_{ij_t} = \left[\left(e_{ij_t} \frac{a_i}{100} \frac{sq_{ij}}{100}\right) + \left(e_{ij_t} \frac{a_i}{100} \frac{rq_{ij}}{100}\right)\right] - \left(e_{ij_t} \frac{ru_{ij}}{100}\right)$ where: $e_{ij_t} = f(t) = a + bt$ or $f(t) = at^b$ or $f(t) = ae^{bt}$ or $f(t) = \frac{e^{MAX}_{ij}}{1+e^{-a+bt}}$
Areal index	pre-defined: ai_{ij}	empirical estimation: $ai_{ij} = \frac{A_{ij}}{e_{ij}}$
Commercial area	$A_{ij_t} = e^A_{ij_t} ai_{ij}$ $A_{j_t} = \sum_{i=1}^{I} A_{ij_t}$ $A_{j_T} = \sum_{i=1}^{I} \sum_{t=1}^{T} A_{ij_t}$	

Notes: $e^A_{ij_t}$ is the (expected) number of employees of industry i in region j which is located in commercial areas at time t, $e_{ij_{t0}}$ is the employment of industry i in region j at start time $t0$ (empirical value), e_{ij_t} is the (expected) employment of industry i in region j at time t, a_i is the share of employees in industry i which is located in commercial areas, sq_{ij} is the resettlement rate with respect to industry i in region j in one time period, rq_{ij} is the relocation rate with respect to industry i in region j in one time period, ru_{ij} is the reutilization rate with respect to industry i in region j in one time period, ai_{ij} is the areal index with respect to industry i in region j (commercial area per employee), A_{ij_t} is the (expected) commercial area for industry i in region j at time t, A_{j_t} is the (expected) commercial area in region j at time t and A_{j_T} is the sum of the (expected) commercial area in region j over all T time periods.
Compiled from: Bonny, Kahnert (2005); CIMA Projekt + Entwicklung GmbH et al. (2011); Deutsches Institut für Urbanistik GmbH, Spath + Nagel (GbR) (2010); Planungsgruppe MWM (2009); Mulligan (2006); Vallée et al. (2012)

The function may be linear (which seems unrealistic) or not: Deutsches Institut für Urbanistik GmbH, Spath + Nagel (GbR) (2010) use linear and exponential functions. However, from the growth perspective, also a logistic function may be applied (see Mulligan 2006 for a discussion of logistic growth with respect to population). If possible, the areal index and, maybe, other parameters are also estimated empirically for the specific region j (e.g. via firm-level surveys and/or official statistical data).

6.2 Application in REAT

6.2.1 REAT functions for analyzing and forecasting regional growth

Table 14 shows the functions for the analysis of regional growth as implemented in `REAT`. Table 15 presents the functions related to commercial area prognosis. All of these functions require at least current employment data for each industry in the regarded region j, e_{ij}, which may be a single `numeric vector` or the column of a `data frame` or `matrix`. Another similarity of all mentioned functions is the optional argument of the industry names (or codes). If no industry names are stated by the user (default function argument: `industry.names = NULL`), the industries are numbered consecutively. With respect to the function output, all regional growth functions distinguish between a visible and an invisible output (see e.g. Section 3), where the main results are returned automatically and the details are included in the invisible output (mostly a `list` with several entries of type `matrix`).

The portfolio matrix (growth portfolio and growth-specialization portfolio, respectively) can be plotted using the functions `portfolio()` and `locq.growth()`, respectively. The different techniques of shift-share analysis are distributed over five functions (`shift()`, `shiftd()`, `shifti()`, `shiftid()` and `shiftp()`). The usage of portfolio and shift-share functions is similar: In any case, the user needs industry-specific employment data for the regarded region and the reference region (e.g. whole economy) for at least two time periods (e.g. years).

Table 14: REAT functions for analyzing regional growth

Model	REAT function	Mandatory arguments	Optional arguments	Output
Growth portfolio matrix	portfolio()	vectors of e_{ij_t} and $e_{ij_{t+y}}$ and vectors of e_{i_t} and $e_{i_{t+y}}$ or matrix/data frame with e_{ij_t} and e_{i_t} for T years, point size (e.g. $e_{ij_{t+y}}$)	point size factor, industry names	visible: plot, invisible: growth rates (matrix)
Growth and specialization portfolio matrix	locq.growth()	vectors of e_{ij_t} and $e_{ij_{t+y}}$ and vectors of e_{i_t} and $e_{i_{t+y}}$ or matrix/data frame with e_{ij_t} and e_{i_t} for T years, point size (e.g. $e_{ij_{t+y}}$)	point size factor, industry names	visible: plot, invisible: list with portfolio data (matrix), LQ_{ij} (matrix) and growth rates (matrix)
Shift-share analysis	shift()	vectors of e_{ij_t} and $e_{ij_{t+y}}$, vectors of e_{i_t} and $e_{i_{t+y}}$	shift-share method (default: Dunn), industry names, plot components, plot portfolio	visible: matrix with components, invisible: list with components (matrix), growth (matrix) and shift method (char), optional: plot(s)
dynamic	shiftd()	vectors of $e_{ij_{t0}}$ and $e_{i_{t0}}$, matrix/data frame with e_{ij_t} and e_{i_t} for T years	shift-share method (default: Dunn), industry names, plot components, plot portfolio	visible: matrix with annual components, invisible: list with components (matrix), annual components (matrix), growth (matrix) and shift method (char), optional: plot(s)
industry-specific	shifti()	vectors of e_{ij_t} and $e_{ij_{t+y}}$, vectors of e_{i_t} and $e_{i_{t+y}}$	shift-share method (default: Dunn), industry names, plot components, plot portfolio	visible: matrix with industry components, invisible: list with components (matrix), industry components (matrix), growth (matrix) and shift method (char), optional: plot(s)
industry-specific and dynamic	shiftid()	vectors of $e_{ij_{t0}}$ and $e_{i_{t0}}$, matrix/data frame with e_{ij_t} and e_{i_t} for T years	shift-share method (default: Dunn), industry names, plot components, plot portfolio	visible: matrix with industry components, invisible: list with components (matrix), industry components (matrix), growth (matrix) and shift method (char), optional: plot(s)
prognosis	shiftp()	vectors of e_{ij_t} and $e_{ij_{t+y}}$, vectors of e_{i_t} and $e_{i_{t+y}}$, vector of $e^P_{i_{t+z}}$	industry names, plot	visible: matrix with industry components, invisible: list with industry employment prognosis (matrix), components (matrix), industry components (matrix), growth (matrix) and shift method (char), optional: plots

Source: own compilation.

Table 15: REAT functions for commercial area prognosis

Model	REAT function	Mandatory arguments	Optional arguments	Output
GIFPRO	gifpro()	vectors of e_{ij}, a_i, sq_{ij}, rq_{ij} and ai_{ij}, time interval, time base	vector of ru_{ij}, industry names, type of output	visible: total commercial area and (optional) annual values, invisible: list with components (matrices), annual and all-over results (list with two matrices)
TBS-GIFPRO	gifpro.tbs()	vectors of e_{ij_t} for T years, a_i, sq_{ij}, rq_{ij} and ai_{ij}, time interval, time base, trend function types	vector of ru_{ij}, industry names, type of output, employment forecast only	visible: total commercial area and (optional) annual values, invisible: list with components (matrices), annual and all-over results (list with two matrices), industry-specific forecast model results (list with I matrices)

Source: own compilation.

All functions for shift-share analysis (except for shift-share prognosis with `shiftp()`) provide three variants of calculation of the components: The classical Dunn method (default function argument `shift.method="Dunn"`), the Dunn extension by Esteban-Marquillas (1972) (`shift.method="Esteban"`) producing four components instead of three, and the Gerfin method (`shift.method="Gerfin"`). When calculating a dynamic shift-share analysis, the user must choose the function `shiftd()`. Industry-specific components are returned by the function `shifti()`. With `shiftid()` one can combine both approaches. Here, it is important to recognize that the function structure allows a combination of e.g. industry-specific and dynamic components while calculating the components from the Esteban-Marquillas extension of shift-share analysis. Additionally, the shift-share functions may plot a portfolio matrix (function argument `plot.portfolio = TRUE`), allowing portfolio and shift-share analysis at once.

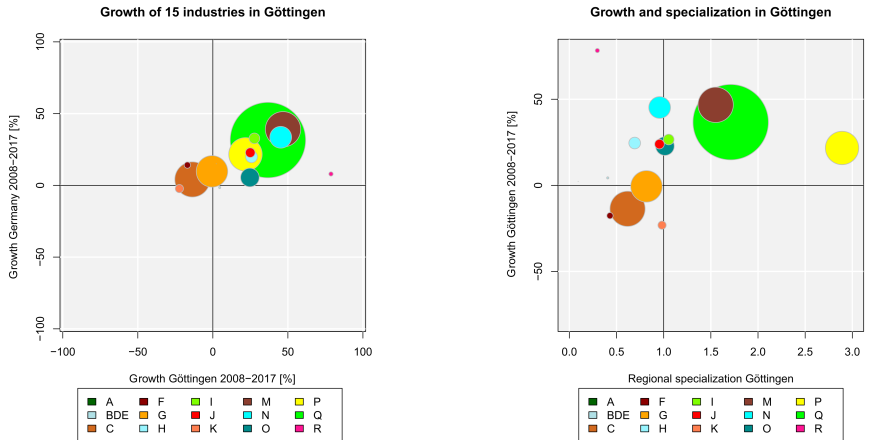
Both functions for commercial area prognosis (`gifpro()` and `gifpro.tbs()`) require vectors of employment data as well as the coefficients for resettlement etc. When forecasting commercial area using the trend-specific technique with `gifpro.tbs()`, the user needs time series data of previous industry-specific employment and has to specify a trend function type (linear, power, exponential or logistic) for each industry. The "best" function type may be examined visually by regarding the employment forecasting output (optional function argument `prog.plot = TRUE`) and the related R^2 values which is part of the invisible function output. Note that this function uses the `REAT` function `curvefit()`, which is a simple tool for bivariate regression, similar to the curve fitting functions in other spreadsheet or statistics software.

6.2.2 Application example 1: Analysis of regional growth in Göttingen

Referring to the example in Section 4.2.2, we perform a regional growth analysis for the German city Göttingen. We use the same dataset `Goettingen` as before, that contains industry-specific employment data for Göttingen and Germany from 2008 to 2017. We load our example data:

```
data(Goettingen)
```

In the first step, we want to examine the industry-specific growth in Göttingen visually. Using the function `portfolio()`, we plot a regional growth matrix with respect to the 15 industries (rows 2 to 16). We also set a plot title (argument `pmtitle`) and axis labels (arguments `pmx` and `pmy`, respectively) as well as industry-specific colors (argument `pcol`):



(a) Growth portfolio matrix for Göttingen (b) Growth-specialization matrix for Göttingen

Figure 6: Portfolio matrix analysis for 15 industries in Göttingen

```
portfolio (Goettingen$Goettingen2008[2:16],
Goettingen$Goettingen2017[2:16],
Goettingen$BRD2008[2:16], Goettingen$BRD2017[2:16],
psize = Goettingen$Goettingen2017[2:16], psize.factor = 15,
pmtitle = "Growth of 15 industries in Göttingen",
industry.names = Goettingen$WZ2008_Code[2:16],
pmx = "Growth Göttingen 2008-2017 [%]",
pmy = "Growth Germany 2008-2017 [%]",
pcol.border = "grey",
pcol = c("darkgreen", "powderblue", "chocolate", "darkred",
"orange", "cadetblue1", "chartreuse1", "red",  "coral",
"coral4", "cyan", "darkcyan", "yellow", "green", "deeppink"),
leg = TRUE, leg.x = -90)
```

Similarly, we plot a growth-specialization portfolio matrix using `locq.growth()` with the same options (colors etc.). On the y axis, we put the industry-specific regional growth which is stated by the function argument `y.axis = "r"` (if we would like to see the national growth instead, we had to set `y.axis = "n"`; for the quotient of regional and national growth, use `y.axis = "rn"`):

```
locq.growth (Goettingen$Goettingen2008[2:16],
Goettingen$Goettingen2017[2:16],
Goettingen$BRD2008[2:16], Goettingen$BRD2017[2:16],
psize = Goettingen$Goettingen2017[2:16], psize.factor = 15,
y.axis = "r", industry.names = Goettingen$WZ2008_Code[2:16],
pmtitle = "Growth and specialization in Göttingen",
pmx = "Regional specialization Göttingen",
pmy = "Growth Göttingen 2008-2017 [%]", pcol.border = "grey",
pcol = c("darkgreen", "powderblue", "chocolate", "darkred",
"orange", "cadetblue1", "chartreuse1", "red", "coral",
"coral4", "cyan", "darkcyan", "yellow", "green", "deeppink"),
leg = TRUE, leg.x = 0.1)
```

The resulting growth portfolio matrix is shown in Figure 6a, the growth-specialization portfolio in Figure 6b. The size of the points (or bubbles) is equal to the current industry-specific employment (e_{ij}) for 2017 (rows 2 to 16 of column `Goettingen2017` in the example data), normalized with respect to a maximum point size of 15 (argument `psize.factor = 15`). As we can see, the health sector (industry code Q, green bubble) has the highest absolute relevance, which can be attributed to the local university hospital (see Section 4.2.2). The axes in the growth portfolio are segmented at $x = 0$ and $y = 0$, respectively, which means a differentiation between positive and negative growth.

As we can see, most industries have grown from 2008 to 2017 in both the region and the whole economy (see quadrant I) with similar growth rates. There is one outlier: Industry R (arts, entertainment, and recreation) shows a regional growth of more than 75 percent, while the national growth is about 10 percent. Note that we see percentage growth rates from 2008 to 2017 here (if *average* growth rates are desired, use the function argument `time.periods`).

Looking at the growth-specialization portfolio, we can identify absolute relevance and growth rate as well as regional specialization of the industries (The colors and bubble sizes are equal to those in Figure 6a). In quadrant I, we find the industries which are overrepresented in Göttingen (specialization) and growing at this regional level. As expected in this university city and related to our results in Section 4.2.2, the "stars" in Göttingen are education (code P), health (code Q) and professional, scientific and technical services (code M).

While the portfolio matrix analysis tells us about the industry-specific growth, the shift-share analysis decomposes this growth into the national, industrial and regional components. In the first step, we perform a static shift-share analysis in the sense of Dunn Jr. (1960) for the same data as in the portfolio analysis by applying the function `shift()`:

```
shift(Goettingen$Goettingen2008[2:16], Goettingen$Goettingen2017[2:16],
Goettingen$BRD2008[2:16], Goettingen$BRD2017[2:16])
# rows 2-6: 15 industries
# columns Goettingen2008 and Goettingen2017:
# employment Goettingen 2008 and 2017, respectively
# columns BRD2008 and BRD2017:
# employment Germany 2008 and 2017, respectively
```

This is our (visible) output:

```
Shift-Share Analysis
Method: Dunn

Shift-share components
                Components
Growth (t1-t)   10411.0000
National share   9178.1916
Industrial mix   2204.8202
Regional share   -972.0118
Net total shift  1232.8084

Calculation for 15 industries
Regional employment at time t: 56872, at time t+1:
67283 (10411 / 18.30602 %)
National employment at time t: 27695398, at time t+1:
32164973 (4469575 / 16.13833 %)
```

In this cross-sectional analysis, we see that the overall employment in Göttingen increased by 10,411 persons from 2008 to 2017. However, a large share of this growth is due to the growth in the national economy ($n_{j_{t,t+y}} \approx 9,178$ employees), which is only a bit lower than Göttingen. The industrial mix component ($m_{j_{t,t+y}}$) shows that approximately 2,205 additional employees must be attributed to an overrepresentation of growing industries in Göttingen. The regional share is negative ($c_{j_{t,t+y}} \approx -972$), which indicates locational disadvantages. When interpreting the industrial mix also as a regional aspect (which seems plausible), we can look at the sum of the industrial mix and the regional share: The net total shift (t_{t+y}) is equal to 1,233 employees, representing the growth difference between the region and the whole economy.

We confirm our results using the Gerfin technique. We request it by setting the argument `shift.method` of the `shift()` function equal to `"Gerfin"`:

```
shift(Goettingen$Goettingen2008[2:16], Goettingen$Goettingen2017[2:16],
Goettingen$BRD2008[2:16], Goettingen$BRD2017[2:16],
shift.method = "Gerfin")
```

The output is:

```
Shift-Share Analysis
Method: Gerfin

Shift-share components
               Components
Industrial mix   1.0333810
Regional share   0.9857591
Net total shift  1.0186647

Calculation for 15 industries
Regional employment at time t: 56872, at time t+1:
67283 (10411 / 18.30602 %)
National employment at time t: 27695398, at time t+1:
32164973 (4469575 / 16.13833 %)
```

In the index method, there is no national share component (implicitly, it is equal to one), thus, we only take a look at the industrial mix and the regional share as well as the net total shift. The industrial mix component is above one ($n_{j_{t,t+y}} \approx 1.03$), showing a more advantageous sector structure in Göttingen compared to Germany. While the regional share in the Dunn-type shift-share analysis was negative, this component in the Gerfin analysis is slightly below one ($c_{j_{t,t+y}} \approx 0.99$), indicating locational disadvantages.

These traditional techniques only regard the overall growth with respect to cross-sectional data. To gain a deeper insight and take into account also seasonal effects, we perform a dynamic shift-share analysis in the sense of Barff, Knight (1988) which distinguishes between the 15 industries simultaneously. This can be done via the `REAT` function `shiftid()`, requiring data for the initial time period and at least for two following periods. In the `Goettingen` dataset, the rows 2 to 16 represent the industries and the columns represent the years (2008 to 2017). Data for the regarded region and the whole economy is arranged successively. We also use the industry codes in column `WZ2008_Code`. In this function, we have to define the start and end periods explicitly:

```
shiftid(Goettingen$Goettingen2008[2:16], Goettingen[2:16,3:12],
Goettingen$BRD2008[2:16], Goettingen[2:16,13:22],
time1 = 2008, time2 = 2017,
industry.names = Goettingen$WZ2008_Code[2:16])
# columns 3-12: employment in Göttingen 2009-2017
# columns 13-22: employment in Germany 2009-2017
```

The result is:

```
Dynamic Shift-Share Analysis
Method: Dunn

Shift-share components
                         A      BDE          C         F         G
Growth (t1-t)   -3.000000 29.00000 -1117.0000 -255.0000  -51.0000
National share   6.103502 -9.46377   254.5217  160.0638  561.7436
Regional share  -9.103502 38.46377 -1371.5217 -415.0638 -612.7436
Net total shift -9.103502 38.46377 -1371.5217 -415.0638 -612.7436
                        H         I         J           K        M
Growth (t1-t)   524.0000 470.00000 274.00000 -465.000000 2229.000
National share  368.2053 515.03493 286.32383    6.356612 1821.392
Regional share  155.7947 -45.03493 -12.32383 -471.356612  407.608
Net total shift 155.7947 -45.03493 -12.32383 -471.356612  407.608
                        N        O         P        Q         R
Growth (t1-t)   1178.0000 268.0000 1272.0000 4211.000 363.00000
National share   977.9869 167.9118 1138.5383 3556.692  47.50353
Regional share   200.0131 100.0882  133.4617  654.308 315.49647
Net total shift  200.0131 100.0882  133.4617  654.308 315.49647
```

```
Calculation for 15 industries
Regional employment at time t: 56872, at time t+1:
67283 (10411 / 18.30602 %)
National employment at time t: 27695398, at time t+1:
32164973 (4469575 / 16.13833 %)
```

The visible output is a `matrix` containing one row for each component (the number of components depends on the selected shift-share method, here: Dunn) and I columns (one for each industry). As we calculate industry-specific components, there is no industrial mix effect, which means that the calculations are on the level of single industries. Again, we detect large absolute growth for industries P (education) and Q (health) (see Table 9). Interestingly, this growth can be mainly attributed to effects in the whole economy. The corresponding regional shares are small but positive, showing locational advantages with respect to these industries in Göttingen.

The logic of shift-share analysis can also be regarded in two other examples: If industry C (manufacturing) *had* developed as in the national trend, the absolute growth in Göttingen *would* be equal to 255 employees. In fact, there was a decline of 1,117 employees, resulting in a negative regional share of -1,372 employees, indicating locational disadvantages with respect to the manufacturing sector. The opposite is true for the industries with code BDE (including electricity, gas, water supply, etc.): The absolute growth of 29 employees *would* not have occurred if this sector *had* developed as in the whole economy (negative national share equal to -9 employees). The residuum (regional share) is equal to 38 employees, indicating a trend contrary to the national.

6.2.3 Application example 2: Commercial area prognosis for Göttingen

Using the same data, we now perform a commercial area prognosis for Göttingen. We load our data:

```
data(Goettingen)
```

When using the GIFPRO-based commercial area prognosis techniques, several parameters have to be defined (employment shares in commercial areas a_i, resettlement rate sq_{ij}, relocation rate rq_{ij} and areal index ai_{ij}; a reutilization rate ru_{ij} is optional, thus, we ignore the reutilization of commercial area in this example). These parameters have to be defined for each industry. In our example, we use the employment shares as well as the resettlement and relocation rates from Deutsches Institut für Urbanistik GmbH, Spath + Nagel (GbR) (2010). Note that some sectors are, per definition, not located within commercial areas (e.g. agriculture), resulting in an employment share of $a_i = 0$. As we want to reuse the sets of parameters, we save them as single `numeric vectors`:

```
ca_share <- c(0, 0, 100, 90, 70, 100, 10, 20, 20, 20, 20, 0, 0, 0, 0)
# industry-specific shares of employees in commercial areas
sq_quote <- c(0.77, 0.77, 0.15, 0.15, 0.77, 0.15, 0.77, 0.77,
0.77, 0.77, 0.77, 0.77, 0.77, 0.77, 0.77)
# industry-specific resettlement quote
rq_quote <- rep(0.7, 15)
# industry-specific relocation quote (0.7 for each of the 15 industries)
area_index <- c(0, 0, 200, 75, 250, 250, 50, 100, 100, 100, 100,
50, 50, 50, 50)
# industry-specific area index (sqm commercial area per employee)
```

Now, we compute the traditional commercial area prognosis using the `gifpro()` function and the `Goettingen` data as well as the parameters defined above. We forecast the commercial area for five years (`tinterval = 5`). Our base is 2017 (`time.base = 2017`), as this is the last year empirical data is available for. We save the (invisible) output in the `list` object `gifpro_goettingen`:

```
gifpro_goettingen <- gifpro (e_ij = Goettingen$Goettingen2017[2:16],
a_i = ca_share, sq_ij = sq_quote, rq_ij = rq_quote, tinterval = 5,
ai_ij = area_index, time.base = 2017,
industry.names = Goettingen$WZ2008_Code[2:16], output = "full")
```

As we have set `output = "full"`, the visible function output contains overall as well as annual values:

```
GIFPRO
Method: GIFPRO

Employment and commercial area changes (allover)
        Employment Commercial Area
Sum      1113.8785       212981.94
Average   222.7757        42596.39

Employment and commercial area changes (per time unit)
     Employment CommercialArea
2018   222.7757       42596.39
2019   222.7757       42596.39
2020   222.7757       42596.39
2021   222.7757       42596.39
2022   222.7757       42596.39

Calculation for 15 industries
```

In all 15 industries, 1,114 new employees are predicted for the year 2022, resulting in 212,928 square meters required for new commercial area. As the employment prognosis is not based on (nonlinear) trend regression but on constant growth, the absolute employment growth and the required commercial area are equal in each year (223 employees and 42,596 sqm, respectively).

The object `gifpro_goettingen` contains a `list` called `components` containing the single components of prognosis as well as the results already shown in the visible output (`results`). To understand the GIFPRO technique and the related `REAT` function, we take a look at the single components:

```
gifpro_goettingen$components

$resettlement
        2018     2019     2020     2021     2022
A    0.00000  0.00000  0.00000  0.00000  0.00000
BDE  0.00000  0.00000  0.00000  0.00000  0.00000
C   11.81100 11.81100 11.81100 11.81100 11.81100
F    1.80090  1.80090  1.80090  1.80090  1.80090
G   38.00489 38.00489 38.00489 38.00489 38.00489
H    3.72150  3.72150  3.72150  3.72150  3.72150
I    1.73327  1.73327  1.73327  1.73327  1.73327
J    3.12928  3.12928  3.12928  3.12928  3.12928
K    2.67806  2.67806  2.67806  2.67806  2.67806
M   12.18910 12.18910 12.18910 12.18910 12.18910
N    7.50750  7.50750  7.50750  7.50750  7.50750
O    0.00000  0.00000  0.00000  0.00000  0.00000
P    0.00000  0.00000  0.00000  0.00000  0.00000
Q    0.00000  0.00000  0.00000  0.00000  0.00000
R    0.00000  0.00000  0.00000  0.00000  0.00000

$relocation
       2018    2019    2020    2021    2022
A    0.0000  0.0000  0.0000  0.0000  0.0000
BDE  0.0000  0.0000  0.0000  0.0000  0.0000
C   55.1180 55.1180 55.1180 55.1180 55.1180
F    8.4042  8.4042  8.4042  8.4042  8.4042
G   34.5499 34.5499 34.5499 34.5499 34.5499
H   17.3670 17.3670 17.3670 17.3670 17.3670
I    1.5757  1.5757  1.5757  1.5757  1.5757
J    2.8448  2.8448  2.8448  2.8448  2.8448
K    2.4346  2.4346  2.4346  2.4346  2.4346
```

```
M   11.0810 11.0810 11.0810 11.0810 11.0810
N    6.8250  6.8250  6.8250  6.8250  6.8250
O    0.0000  0.0000  0.0000  0.0000  0.0000
P    0.0000  0.0000  0.0000  0.0000  0.0000
Q    0.0000  0.0000  0.0000  0.0000  0.0000
R    0.0000  0.0000  0.0000  0.0000  0.0000

$reuse
    2018 2019 2020 2021 2022
A      0    0    0    0    0
BDE    0    0    0    0    0
C      0    0    0    0    0
F      0    0    0    0    0
G      0    0    0    0    0
H      0    0    0    0    0
I      0    0    0    0    0
J      0    0    0    0    0
K      0    0    0    0    0
M      0    0    0    0    0
N      0    0    0    0    0
O      0    0    0    0    0
P      0    0    0    0    0
Q      0    0    0    0    0
R      0    0    0    0    0

$employment
        2018     2019     2020     2021     2022
A    0.00000  0.00000  0.00000  0.00000  0.00000
BDE  0.00000  0.00000  0.00000  0.00000  0.00000
C   66.92900 66.92900 66.92900 66.92900 66.92900
F   10.20510 10.20510 10.20510 10.20510 10.20510
G   72.55479 72.55479 72.55479 72.55479 72.55479
H   21.08850 21.08850 21.08850 21.08850 21.08850
I    3.30897  3.30897  3.30897  3.30897  3.30897
J    5.97408  5.97408  5.97408  5.97408  5.97408
K    5.11266  5.11266  5.11266  5.11266  5.11266
M   23.27010 23.27010 23.27010 23.27010 23.27010
N   14.33250 14.33250 14.33250 14.33250 14.33250
O    0.00000  0.00000  0.00000  0.00000  0.00000
P    0.00000  0.00000  0.00000  0.00000  0.00000
Q    0.00000  0.00000  0.00000  0.00000  0.00000
R    0.00000  0.00000  0.00000  0.00000  0.00000
```

As we defined some industries as not relevant for commercial areas ($a_i = 0$), they do not contribute any employees neither resettled nor relocated (such as A - agriculture, B - mining and quarrying or R - arts, entertainment, and recreation). We see that e.g. in the manufacturing sector (code C), there is an annual increase of about 12 employees attributed to resettlement and 55 employees related to relocation each year (see row 3 in `resettlement` and `relocation`, respectively). As we ignored the reutilization of commercial area, the `matrix` containing the commercial area-relevant employment related to reutilization (`reuse`) contains only zeros. The sum of all three components is stored in the fourth `matrix`, `employment`. There is an annual increase of nearly 67 employees in the manufacturing sector. The contents of the `results list` is the same as shown in the visible output.

In the next step, we apply the trend-based commercial area prognosis (TBS-GIFPRO) to the `Goettingen` data. In the `gifpro.tbs()` function, we use the employment data from 2008 to 2017 (columns 3 to 12), and assume an exponential function for employment prognosis (function argument `prog.func`, repeating the argument `"exp"` for each industry). The employment prognosis is plotted (`prog.plot = TRUE`), showing all 15 plots in one (`plot.single = FALSE`):

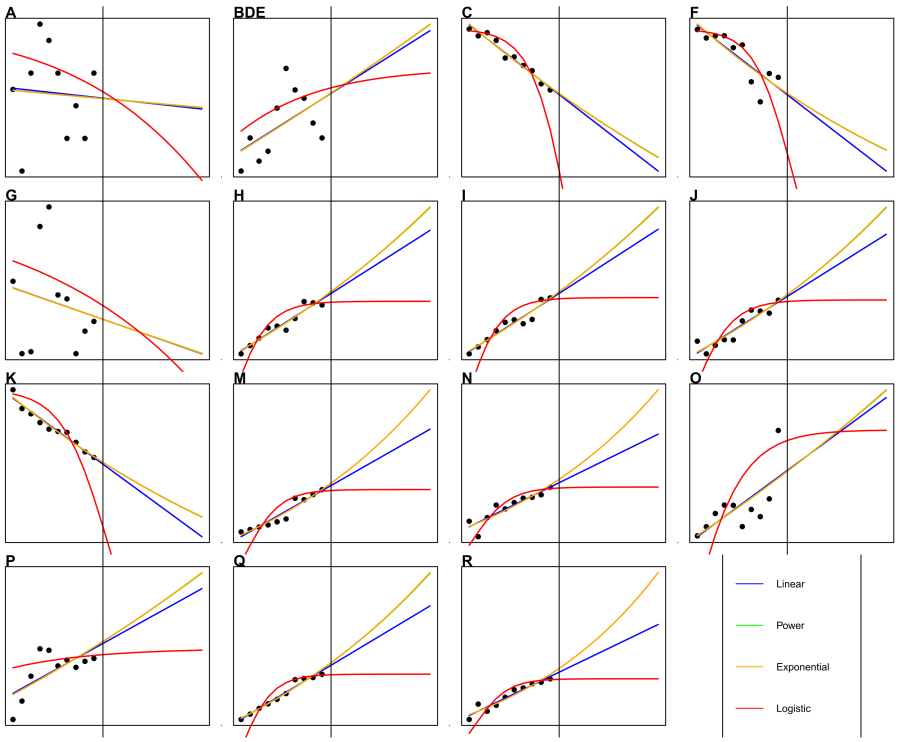


Figure 7: Employment prognosis for 15 industries in Goettingen (TBS-GIFPRO)

```
gifpro.tbs (e_ij = Goettingen[2:16,3:12],
a_i = ca_share, sq_ij = sq_quote, rq_ij = rq_quote, tinterval = 5,
prog.func = rep("exp", nrow(Goettingen[2:16,3:12])),
ai_ij = area_index, time.base = 2008,
industry.names = Goettingen$WZ2008_Code[2:16],
prog.plot = TRUE, plot.single = FALSE, output = "full")
```

The visible function output is similar to the output above:

```
GIFPRO
Method: TBS-GIFPRO

Employment and commercial area changes (allover)
        Employment CommercialArea
Sum      1139.6592      216012.46
Average   227.9318       43202.49

Employment and commercial area changes (per time unit)
     Employment CommercialArea
2018   224.9565       42945.53
2019   226.2904       43054.76
2020   227.7755       43182.97
2021   229.4169       43330.70
2022   231.2199       43498.50

Calculation for 10 industries
```

The resulting plot containing the employment forecasting functions is shown in Figure 7. The black vertical lines divide the plots into the esimation segment (2008 to 2017) and the prognosis segment (2018 to 2022). Four function types are supplied: linear (blue), power (green), exponential (yellow) and logistic (red). Note that a linear trend seems unrealistic as it implies continuous growth and may result in negative employment if the slope is negative. At this point, we should normally discuss and find the "best"

forecasting model for each industry and rerun our analysis a few times. In our example, we skip this step and just take a look at the prognosis functions: In most cases, an exponential growth (or decline) seems to be an appropriate approximation. The power functions (green lines) are nearly invisible as their data fit is nearly the same as that of the exponential functions. Thus, we could choose them instead. In our case, the exponential function seems sufficient.

As expected, a nonlinear industry growth results in a nonlinear overall employment growth and, consequently, the commercial area-relevant employment also grows in a nonlinear way. As we can see from the `gifpro.tbs()` output, employment increases by about 228 employees per year on average and by about 1,140 employees over the five years regarded (2018 to 2022). The annual commercial area required ranges from 42,946 sqm (2018) to 43,499 sqm (2022), all in all 216,012 sqm up to 2022. In our case, the estimated commercial area exceeds the prognosis derived from the simple GIFPRO analysis, which can be attributed to the positive differences between the exponential prognosis and a linear prognosis (see Figure 7). We skip the inspections of the components, which could be addressed by saving the results in an object (`list`), as we did in the first GIFPRO example.

7 Final remarks

This paper has shown how `R` and specifically the package `REAT` can be used for regional economic analysis. It should be noted that this package aims at width with respect to the treated analysis subjects rather than depth. The subsections provide the basic analysis methods regarded as most important from the package developer's point of view (with respect to usage in current papers and discussion in current textbooks as well as application in own research projects), while there are several other approaches as well as extensions of the basic methods. A more detailed survey of the common methods can be found in the cited literature, especially in review articles (e.g. Nakamura, Morrison Paul 2009; Portnov, Felsenstein 2010) and textbooks (e.g. Farhauer, Kröll 2014).

Finally, we have to keep in mind that this package (like nearly any other free software) was developed in a non-commercial context (and published under the GNU General Public License). All functions have been tested several times using various real data and single functions have already been used in a few research projects. However, there is no warranty that all functions always work perfectly. Like nearly any other `R` package, `REAT` is continuously refined, which means extending functions as well as correcting errors. This requires attentive usage and, of course, constructive feedback from the package users. It can be easily transmitted using the contact information on the CRAN package website.

References

Albacete X, Olaru D, Paül V, Biermann S (2017) Measuring the accessibility of public transport: A critical comparison between methods in Helsinki. *Applied Spatial Analysis and Policy* 10[2]: 161–188. CrossRef.

Allington NF, McCombie J (2007) Economic Growth and Beta-Convergence in the East European Transition Economies. In: Arestis P, Baddeley M, McCombie J (eds), *Economic Growth*. Edward Elgar publishing, Cheltenham, 200–222

Arcelus FJ (1984) An extension of shift-share analysis. *Growth and Change* 15[1]: 3–8. CrossRef.

Bai CE, Tao Z, Tong YS (2008) Bureaucratic integration and regional specialization in China. *China Economic Review* 19[2]: 308–319. CrossRef.

Baker P, von Kirchbach F, Mimouni M, Pasteels JM (2002) Analytical Tools for Enhancing the Participation of Developing Countries in the Multilateral Trading System in the Context of the Doha Development Agenda. *Aussenwirtschaft* 57[3]: 343–372. https://EconPapers.repec.org/RePEc:usg:auswrt:2002:57:03:343-372

Balassa B (1965) Trade Liberalisation and "Revealed" Comparative Advantage. *The Manchester School* 33[2]: 99–123. CrossRef.

Barff RA, Knight PL (1988) Dynamic shift-share analysis. *Growth and Change* 19[2]: 1–10. CrossRef.

Barro RJ, Sala-i Martin X (2004) *Economic Growth* (2nd ed.). MIT Press

Bonny HW, Kahnert R (2005) Zur Ermittlung des Gewerbeflächenbedarfs. *Raumforschung und Raumordnung* 63[3]: 232–240

Capello R, Nijkamp P (2009) Introduction: Regional growth and development theories in the twenty-first century - recent theoretical advances and future challenges. In: Capello R, Nijkamp P (eds), *Handbook of Regional Growth and Development Theories*. 1–18

Casler SD (1989) A Theoretical Context for Shift and Share Analysis. *Regional Studies* 23[1]: 43–48. CrossRef.

Ceapraz IL (2008) The Concepts of Specialisation and Spatial Concentration and the Process of Economic Integration: Theoretical Relevance and Statistical Measures. The Case of Romania's Regions. *Romanian Journal of Regional Science* 2[1]: 68–93

Charles-Coll JA (2011) Unterstanding Income Equality: Concept, Causes and Management. *International Journal of Economics and Management Science* 1[3]: 17–28

CIMA Projekt + Entwicklung GmbH, NIW Niedersächsisches Institut für Wirtschaftsforschung, NORD/LB Regionalwirtschaft, Planquadrat Dortmund GbR (2011) Gewerbeflächenkonzeption für die Metropolregion Hamburg (GEFEK). Research report

Cracau D, Durán Lima JE (2016) On the Normalized Herfindahl-Hirschman Index: A Technical Note. *International Journal on Food System Dynamics* 7[4]: 382–386

Damgaard C, Weiner J (2000) Describing inequality in plant size or fecundity. *Ecology* 81[4]: 1139–1142. CrossRef.

Dapena AD, Fernández Vázquez E, Rubiera Morollón F (2016) The role of spatial scale in regional convergence: the effect of MAUP in the estimation of β-convergence equations. *The Annals of Regional Science* 56[2]: 473–489. CrossRef.

Dauth W, Fuchs M, Otto A (2015) Standortmuster in Westdeutschland: Nur wenige Branchen sind räumlich stark konzentriert. IAB Kurzbericht 16/2015, Institut für Arbeitsmarkt- und Berufsforschung. http://doku.iab.de/kurzber/2015/kb1615.pdf

Dauth W, Fuchs M, Otto A (2018) Long-run processes of geographical concentration and dispersion: Evidence from Germany. *Papers in Regional Science* 97[3]: 569–593. CrossRef.

Deutsches Institut für Urbanistik GmbH, Spath + Nagel (GbR) (2010) Stadtentwicklungskonzept Gewerbe für die Landeshauptstadt Potsdam. Research report, Landeshauptstadt Potsdam. https://www.potsdam.de/sites/default/files/documents/STEK_Gewerbe_Langfassung_2010.pdf

Dinc M (2015) *Introduction to Regional Economic Development. Major Theories and Basic Analytical Tools*. Elgar

Dixon R, Freebairn J (2009) Trends in Regional Specialisation in Australia. *Australasian Journal of Regional Studies* 15[3]: 281–296

Doran J, Jordan D (2013) Decomposing European NUTS2 regional inequality from 1980 to 2009: National and European policy implications. *Journal of Economic Studies* 40[1]: 22–38. CrossRef.

Dunn Jr. ES (1960) A Statistical and Analytical Technique for Regional Analysis. *Papers in Regional Science* 6[1]: 97–112. CrossRef.

Duranton G, Puga D (2000) Diversity and Specialisation in Cities: Why, Where and When Does it Matter? *Urban Studies* 37[3]: 533–555. CrossRef.

Ellison G, Glaeser E (1997) Geographic Concentration in U.S. Manufacturing Industries: A Dartboard Approach. *Journal of Political Economy* 105[5]: 889–927

Espa G, Arbia G, Giuliani D (2010) Measuring industrial agglomeration with inhomogeneous K-function: the case of ICT firms in Milan (Italy). Department of Economics Working Papers 1014, Department of Economics, University of Trento, Italia

Esteban-Marquillas JM (1972) I. A reinterpretation of shift-share analysis. *Regional and Urban Economics* 2[3]: 249 – 255. CrossRef.

Farhauer O, Kröll A (2014) *Standorttheorien. Regional- und Standortökonomik in Theorie und Praxis* (2nd ed.). Springer, Heidelberg

Fujita M, Krugman P, Venables A (2001) *The Spatial Economy: Cities, Regions, and International Trade* (1st ed.), Volume 1. The MIT Press

Fülöp G, Kopetsch T, Schöpe P (2011) Catchment areas of medical practices and the role played by geographical distance in the patient's choice of doctor. *The Annals of Regional Science* 46[3]: 691–706. CrossRef.

Furceri D (2005) Beta and sigma convergence: A mathematical relation of causality. *Economics Letters* 89[2]: 212–215. CrossRef.

Gerfin H (1964) Gesamtwirtschaftliches Wachstum und regionale Entwicklung. *Kyklos* 17[4]: 565–593. CrossRef.

Gini C (1912) *Variabilità e Mutuabilità. Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche*. Cuppini

Gluschenko K (2018) Measuring regional inequality: to weight or not to weight? *Spatial Economic Analysis* 13[1]: 36–59. CrossRef.

Goecke H, Hüther M (2016) Regional Convergence in Europe. *Intereconomics* 51[3]: 165–171. CrossRef.

Goschin Z, Constantin D, Roman M, Ileanu B (2009) Regional specialization and geographic concentration of industries in Romania. *South-Eastern Europe Journal of Economics* 1[1]: 99–113. https://ojs.lib.uom.gr/index.php/seeje/article/view/5536

Haas A, Südekum J (2005) Spezialisierung und Branchenkonzentration in Deutschland: Regionalanalyse. IAB-Kurzbericht 1/2005. http://hdl.handle.net/10419/158181

Habánik J, Hošták P, Kútik J (2013) Economic and social disparity development within regional development of the Slovak Republic. *Economics and Management* 18[3]: 457–464. CrossRef.

Hansen WG (1959) How Accessibility Shapes Land Use. *Journal of the American Institute of Planners* 25[2]: 73–76. CrossRef.

Harris CD (1954) The Market as a Factor in the Localization of Industry in the United States. *Annals of the Association of American Geographers* 44[4]: 315–348

Haynes KE, Parajuli J (2014) Shift-share analysis: decomposition of spatially integrated systems. In: *Handbook of Research Methods and Applications in Spatially Integrated Social Science*. Elgar, 315–344. CrossRef.

Heinemann M (2008) Messung und Darstellung von Ungleichheit. Working Paper Series in Economics 108, University of Lüneburg, Institute of Economics. https://EconPapers.repec.org/RePEc:lue:wpaper:108

Henderson BD (1973) The Experience Curve - Reviewed. IV. The Growth Share Matrix or The Product Portfolio. Reprint 135. https://www.bcg.com/documents/file13904.pdf

Herfindahl OC (1950) *Concentration in the U.S. Steel Industry*. Colombia University Press

Hirschman AO (1945) *National Power and the Structure of Foreign Trade*. Publications of the Bureau of Business and Economic Research. University of California Press

Hoen AR, Oosterhaven J (2006) On the measure of comparative advantage. *The Annals of Regional Science* 40[3]: 677–691. CrossRef.

Hoffmann J, Hirsch S, Simons J (2017) Identification of spatial agglomerations in the German food processing industry. *Papers in Regional Science* 96[1]: 139–162. CrossRef.

Hoover EM (1936) The Measurement of Industrial Localization. *The Review of Economics and Statistics* 18[4]: 162–171

Howard D (2007) A regional economic performance matrix – an aid to regional economic policy development. *Journal of Economic and Social Policy* 11[2]: article 4. https://EconPapers.repec.org/RePEc:usg:auswrt:2002:57:03:343-372

Howard E, Newman C, Tarp F (2016) Measuring industry coagglomeration and identifying the driving forces. *Journal of Economic Geography* 16[5]: 1055–1078

Huang Y, Leung Y (2009) Measuring Regional Inequality: A Comparison of Coefficient of Variation and Hoover Concentration Index. *The Open Geography Journal* 2[1]: 25–34. CrossRef.

Jiang L, Guan M, Tian J (2007) On Chinese Regional Specialization and Industry Concentration. In: *2007 International Conference on Machine Learning and Cybernetics*, Volume 6, 3396–3400

Kabacoff RI (2017) Quick-R: Data Types. Manual. https://www.statmethods.net/input/datatypes.html

Kassenärztliche Bundesvereinigung (2013) Die neue Bedarfsplanung. Grundlagen, Instrumente und regionale Möglichkeiten. Brochure. https://www.kbv.de/media/sp/Instrumente_Bedarfsplanung_Broschuere.pdf

Kim S (1995) Expansion of Markets and the Geographic Distribution of Economic Activities: The Trends in U. S. Regional Manufacturing Structure, 1860–1987. *The Quarterly Journal of Economics* 110[4]: 881–908. CrossRef.

Kiskowski MA, Hancock JF, Kenworthy A (2009) On the Use of Ripley's K-function and its Derivatives to Analyze Domain Skriderize. *Biophysical Journal* 97[4]: 1095–1103. CrossRef.

Kohn W, Öztürk R (2013) *Statistik für Ökonomen. Datenanalyse mit R und SPSS* (2nd ed.). Springer Gabler

Krider R, Putler DS (2013) Which Birds of a Feather Flock Together? Clustering and Avoidance Patterns of Similar Retail Outlets. *Geographical Analysis* 45[2]: 123–149. CrossRef.

Krugman P (1979) Increasing returns, monopolistic competition, and international trade. *Journal of International Economics* 9[4]: 469–479. CrossRef.

Krugman P (1991) *Geography and trade.* MIT Press

Larsson JP, Öner Ö (2014) Location and co-location in retail: a probabilistic approach using geo-coded data for metropolitan retail markets. *The Annals of Regional Science* 52[2]: 385–408. CrossRef.

Lehocký F, Rusnák J (2016) Regional specialization and geographic concentration: experiences from Slovak industry. *Miscellanea Geographica – Regional Studies on Development* 20[3]: 5–13. https://www.degruyter.com/downloadpdf/j/mgrsd.2016.20.issue-3/mgrsd-2016-0011/mgrsd-2016-0011.pdf

Lessmann C (2005) Regionale Disparitäten in Deutschland und ausgesuchten OECD-Staaten im Vergleich. *ifo Dresden berichtet* 3/2005: 25–33

Lessmann C (2014) Spatial inequality and development - Is there an inverted-U relationship? *Journal of Development Economics* 106: 35–51. CrossRef.

Lessmann C (2016) Regional inequality and internal conflict. *German Economic Review* 17[2]: 157–191. CrossRef.

Lessmann C, Seidel A (2017) Regional inequality, convergence, and its determinants – a view from outer space. *European Economic Review* 92: 110–132. CrossRef.

Litzenberger T, Sternberg R (2006) Der Clusterindex – eine Methodik zur Identifizierung regionaler Cluster am Beispiel deutscher Industriebranchen. *Geographische Zeitschrift* 94[2]: 209–224

Longley PA, Goodchild MF, Maguire DJ, Rhind DW (2005) *Geographical Information Systems and Science* (2nd ed.). Wiley

Lorenz MO (1905) Methods of measuring the concentration of wealth. *Publications of the American Statistical Association* 9[70]: 209–219. CrossRef.

Martin C (2015) Kreative Klasse 2015. Kreativität als entscheidender Faktor für wirtschaftlichen Erfolg: Entwicklungen und Ausprägungen in Deutschland. Research report. https://www.kreativ-sta.de/wp-content/uploads/2017/10/agiplan_Kreative_Klasse_2015_Studie.pdf

Midelfart-Knarvik K, Overman H, Redding S, Venables A (2000) The Location of European Industry. *European Economy - Economic Papers* 142

Moga LM, Constantin DL (2011) Specialization and Geographic Concentration of the Economic Activities in the Romanian Regions. *Journal of Applied Quantitative Methods* 6[2]: 12–21. https://pdfs.semanticscholar.org/aa9d/365d6a8ef4c3585595c8ba03fe373ab02010.pdf

Mulligan GF (2006) Logistic Population Growth in the World's Largest Cities. *Geographical Analysis* 38[4]: 344–370. CrossRef.

Mussini M (2017) Decomposing Changes in Inequality and Welfare Between EU Regions: The Roles of Population Change, Re-Ranking and Income Growth. *Social Indicators Research* 130[2]: 455–478. CrossRef.

Myrdal G (1957) *Economic theory and under-developed regions*. G. Duckworth

Nakamura R, Morrison Paul C (2009) Measuring agglomeration. In: Capello R, Nijkamp P (eds), *Handbook of Regional Growth and Development Theories*. Elgar, 305–328

Nischwitz G, Böhme R, Fortmann F (2017) Kommunale Wirtschaftsförderung in Bremen: Handlungsrahmen, Programme und Wirkungen. Schriftenreihe Institut Arbeit und Wirtschaft 23/2017. http://hdl.handle.net/10419/172756

O'Donoghue D, Gleave B (2004) A Note on Methods for Measuring Industrial Agglomeration. *Regional Studies* 38[4]: 419–427. CrossRef.

OECD (2019) OECD Territorial Reviews. Website. https://www.oecd-ilibrary.org/fr/urban-rural-and-regional-development/oecd-territorial-reviews_19900759

Palan N (2017) Konzentrations- und Ungleichheitsindizes: ein methodischer Überblick sowie ein empirischer Vergleich anhand der Textilindustrie. *Zeitschrift für Wirtschaftsgeographie* 61[3-4]: 135–155. CrossRef.

Peña Carrera L (2002) Tracing accessibility over time: two swiss case studies. Technical report. http://hdl.handle.net/2099.1/6327

Petrakos G, Psycharis Y (2016) The spatial aspects of economic crisis in Greece. *Cambridge Journal of Regions, Economy and Society* 9[1]: 137–152. CrossRef.

Planungsgruppe MWM (2009) Flächennutzungsplanung Gemeinde Wachtberg - Fachbeitrag Arbeiten. Report. http://www.wachtberg.de/imperia/md/content/cms127/gemeindeentwicklung/fnp-fb-arbeiten-24-02-2009.pdf

Pooler J (1987) Measuring geographical accessibility: a review of current approaches and problems in the use of population potentials. *Geoforum* 18[3]: 269 – 289. CrossRef.

Porter ME (1990) *The Competitive Advantage of Nations*. Free Press

Portnov BA, Felsenstein D (2005) Measures of Regional Inequality for Small Countries. In: Felsenstein D, Portnov B (eds), *Regional Disparities in Small Countries*. 47–62. CrossRef.

Portnov BA, Felsenstein D (2010) On the suitability of income inequality measures for regional analysis: Some evidence from simulation analysis and bootstrapping tests. *Socio-Economic Planning Sciences* 44[4]: 212–219. CrossRef.

Puente S (2017) Regional convergence in Spain: 1980-2015. Research report. Economic Bulletin 3/2017, Banco de Espana

R Core Team (2018a) R: A Language and Environment for Statistical Computing. Software, Vienna, Austria. https://www.R-project.org/

R Core Team (2018b) The R Manuals. Manual. https://cran.r-project.org/manuals.html

Reggiani A, Bucci P, Russo G (2011) Accessibility and Impedance Forms: Empirical Applications to the German Commuting Network. *International Regional Science Review* 34[2]: 230–252. CrossRef.

Ricardo D (1821) *On the Principles of Political Economy and Taxation* (3rd ed.). McMaster University Archive for the History of Economic Thought

Ripley BD (1976) The second-order analysis of stationary point processes. *Journal of Applied Probability* 13[2]: 255–266. CrossRef.

RStudio Team (2016) RStudio: Integrated Development Environment for R. Software, RStudio, Inc., Boston, MA. http://www.rstudio.com/

Schmidt H (1997) *Konvergenz wachsender Volkswirtschaften. Theoretische und empirische Konzepte sowie eine Analyse der Produktivitätsniveaus westdeutscher Regionen*, Volume 152 of *Wirtschaftswissenschaftliche Beiträge*. Springer

Schönebeck C (1996) *Wirtschaftsstruktur und Regionalentwicklung : theoretische und empirische Befunde für die Bundesrepublik Deutschland*, Volume 75 of *Dortmunder Beiträge zur Raumplanung Blaue Reihe*. IRPUD

Schätzl L (2000) *Wirtschaftsgeographie 2: Empirie* (3rd ed.). Schöningh

Smith TE (2016) Notebook on Spatial Data Analysis. Technical report. http://www.seas.upenn.edu/~ese502/#notebook

Spiekermann K, Wegener M (2008) Modelle in der Raumplanung I: 4. Input-Output-Modelle. Presentation, Lecture "Modelle in der Raumplanung" WS 2008/2009. http://www.spiekermann-wegener.de/mir/pdf/MIR1_4_111108.pdf

Stark KD, Velsinger P, Bauer M, Bonny HW, Kricke J, Schwetlick D, Striedel HD (1981) *Flächenbedarfsberechnung für Gewerbe- und Industrieansiedlungsbereiche: GIFPRO*. Number 4.029 in Schriftenreihe Landes- und Stadtentwicklungsforschung des Landes Nordrhein-Westfalen. ILS, Dortmund

Statistisches Bundesamt (2008) German Classification of Economic Activities, Edition 2008. Dataset (XLS). https://www.destatis.de/DE/Methoden/Klassifikationen/GueterWirtschaftklassifikationen/klassifikationWZ08englisch.xls

Störmann W (2009) *Regionalökonomik. Theorie und Praxis*. Oldenbourg, Munich

Taylor JK, Cihon C (2004) *Statistical Techniques for Data Analysis* (2nd ed.). Taylor and Francis

Theil H (1967) *Economics and information theory*. North-Holland

Tian Z (2013) Measuring agglomeration using the standardized location quotient with a bootstrap method. *Journal of Regional Analysis and Policy* 43[2]: 186–197

Vallée D, Witte A, Brandt T, Bischof T (2012) Bedarfsberechnung für die Darstellung von Allgemeinen Siedlungsbereichen (ASB) und Gewerbe- und Industrieansiedlungsbereichen (GIB) in Regionalplänen. Research report, Staatskanzlei des Landes Nordrhein-Westfalen. https://www.wirtschaft.nrw/sites/default/files/asset/document/lep_nrw_flaechenbedarf_endbericht_endfassung_04122012.pdf

Vogiatzoglou K (2006) Increasing agglomeration or dispersion? Industrial specialization and geographic concentration in NAFTA. *Journal of Economic Integration* 21[2]: 379–396

von Neumann J, Kent RH, Bellinson HR, Hart BI (1941) The Mean Square Successive Difference. *The Annals of Mathematical Statistics* 12[2]: 153–162. CrossRef.

Weddige-Haaf K, Kool C (2017) Determinants of regional growth and convergence in Germany. Discussion paper. Discussion Paper Series 17-12, Utrecht University School of Economics

Wieland T (2019) REAT: Regional Economic Analysis Toolbox. R package version 3.0.1. Software. https://CRAN.R-project.org/package=REAT

Wieland T, Dittrich C (2016) Bestands- und Erreichbarkeitsanalyse regionaler Gesundheitseinrichtungen in der Gesundheitsregion Göttingen. Research report, Georg-August-Universität Göttingen, Geographisches Institut, Abteilung Humangeographie. http://webdoc.sub.gwdg.de/pub/mon/2016/3-wieland.pdf

Wieland T, Fuchs H (2018) Regionalökonomische Disparitäten im Spiegel von Raumtypisierungen. Ein Konzept zur Identifikation strukturell benachteiligter Gebiete in Südtirol (Italien). *Standort - Zeitschrift für Angewandte Geographie* 42[3]: 152–163. CrossRef.

Williamson JG (1965) Regional Inequality and the Process of National Development: A Description of the Patterns. *Economic Development and Cultural Change* 13[4]: 1–84

Yamamura S, Goto H (2018) Location patterns and determinants of knowledge-intensive industries in the Tokyo Metropolitan Area. *Japan Architectural Review* 1[4]: 443–456. CrossRef.

Young AT, Higgins MJ, Levy D (2008) Sigma Convergence versus Beta Convergence: Evidence from U.S. County-Level Data. *Journal of Money, Credit and Banking* 40[5]: 1083–1093. CrossRef.

REGION
The Journal of ERSA
Powered by WU

Volume 6, Number 3, 2019, 17–37
DOI: 10.18335/region.v6i3.276

journal homepage: region.ersa.org

Demonstrating the utility of machine learning innovations in address matching to spatial socio-economic applications*

Sam Comber[1]

[1] University of Liverpool, Liverpool, United Kingdom

Received: 16 August 2019/Accepted: 30 December 2019

Abstract. The last decade has heralded an unprecedented rise in the number, frequency and availability of data sources. Yet they are often incomplete, meaning data fusion is required to enhance their quality and scope. In the context of spatial analysis, address matching is critical to enhancing household socio-economic and demographic characteristics. Matching administrative, commercial, or lifestyle data sources to items such as household surveys has the potential benefits of improving data quality, enabling spatial data visualisation, and the lowering of respondent burden in household surveys. Typically when a practitioner has high quality data, unique identifiers are used to facilitate a direct linkage between household addresses. However, real-world databases are often absent of unique identifiers to enable a one-to-one match. Moreover, irregularities between the text representations of potential matches mean extensive cleaning of the data is often required as a pre-processing step. For this reason, practitioners have traditionally relied on two linkage techniques for facilitating matches between the text representations of addresses that are broadly divided into deterministic or mathematical approaches. Deterministic matching consists of constructing hand-crafted rules that classify address matches and non-matches based on specialist domain knowledge, while mathematical approaches have increasingly adopted machine learning techniques for resolving pairs of addresses to a match. In this notebook we demonstrate methods of the latter by demonstrating the utility of machine learning approaches to the address matching work flow. To achieve this, we construct a predictive model that resolves matches between two small datasets of restaurant addresses in the US. While the problem case may seem trivial, the intention of the notebook is to demonstrate an approach that is reproducible and extensible to larger data challenges. Thus, in the present notebook, we document an end-to-end pipeline that is replicable and instructive towards assisting future address matching problem cases faced by the regional scientist.

1 Introduction

Our overarching objective is to demonstrate how machine learning can be integrated into the address matching work flow. By definition, address matching pertains to the process of resolving pairs of records with a spatial footprint. While geospatial matching links the geometric representations of spatial objects, address matching typically involves linking the text-based representations of address pairs. The utility of address matching, and

*This paper is available as computational notebook on the REGION webpage.

record linkage in general, lies in the ability to unlock attributes from sources of data that cannot be linked by traditional means. This is often because the datasets lack a common key to resolve a join between the address of a premise. Two example applications of address matching uses include: the linkage of historical censuses across time for exploring economic and geographic mobility across multiple generations (Ruggles et al. 2018), and exploring how early-life hazardous environmental exposure, socio-economic conditions, or natural disasters impact the health and economic outcomes of individuals living in particular residential locations (Cayo, Talbot 2003, Reynolds et al. 2003, Baldovin et al. 2015).

For demonstrative purposes, we rely on small a set of addresses from the Fodors and Zagat restaurant guides that contain 112 matched addresses for training a predictive model that resolves address pairs to matches and non-matches. In a real-world application, training a machine learning model on a small sample of matched addresses could be used to resolve matches between the remaining addresses of a larger dataset. While we use the example of restaurant addresses, these could easily be replaced by addresses from a far less trivial source and the work flow required to implement the address matching exercise would remain the same. Therefore, it is the intention of this guide to provide insight on how the work flow of a supervised address matching work flow proceeds, and to inspire interested users to scale the supplied code to larger and more interesting problems.

2 Packages and dependencies

[1]:
```
%matplotlib inline
import os
import uuid
import warnings
from IPython.display import HTML

# load external libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import jellyfish
import recordlinkage as rl
import seaborn as sns
from postal.parser import parse_address # CRF parser
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_validate, train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix

# configure some settings
np.random.seed(123)
sns.set_style('whitegrid')
pd.set_option('display.max_colwidth', -1)
warnings.simplefilter(action='ignore', category=FutureWarning)

def hover(hover_color="#add8e6"):
    return dict(selector="tbody tr:hover",
            props=[("background-color", "%s" % hover_color)])

# table CSS
styles = [
    #table properties
    dict(selector=" ",
         props=[("margin","0"),
                ("font-family",'"Helvetica", "Arial", sans-serif'),
                ("border-collapse", "collapse"),
                ("border","none"), ("border-style", "hidden")]),
    dict(selector="td", props = [("border-style", "hidden"),
                                 ("border-collapse", "collapse")]),

    #header color
    dict(selector="thead",
         props=[("background-color","#a4dbc8")]),
```

```
    #background shading
    dict(selector="tbody tr:nth-child(even)",
         props=[("background-color", "#fff")]),
    dict(selector="tbody tr:nth-child(odd)",
         props=[("background-color", "#eee")]),

    #header cell properties
    dict(selector="th",
         props=[("text-align", "center"),
                ("border-style", "hidden"),
                ("border-collapse", "collapse")]),

    hover()
]
```

3 Data loading, cleaning and segmentation

To begin our exercise we specify the file location that contains the entirety of the 112 Zagat and Fodor matched address pairs. This file can be downloaded from the dedicated Github repository that accompanies the paper (https://github.com/SamComber/address_matching_workflow) using the `wget` command.

[2]:
```
! wget https://raw.githubusercontent.com/SamComber/address_matching_workflow/master/
...zagat_fodor_matched.txt
```

[2]:
```
--2019-12-21 09:11:31--  https://raw.githubusercontent.com/SamComber/address_matching_
workflow/master/zagat_fodor_matched.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.56.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.56.133|:443
... connected.
HTTP request sent, awaiting response... 200 OK
Length: 19939 (19K) [text/plain]
Saving to: ‘zagat_fodor_matched.txt’

zagat_fodor_matched 100%[===================>]  19.47K  --.-KB/s    in 0.03s

2019-12-21 09:11:32 (670 KB/s) - ‘zagat_fodor_matched.txt’ saved [19939/19939]
```

[3]:
```
f = 'zagat_fodor_matched.txt'
```

Address matching is principally a data quality challenge. Similar to other areas of data analysis, when the quality of input data to the match classification is low, the output generated will typically be of low accuracy (Christen 2012). Problematically, most address databases we encounter in the real world are inconsistent, are missing of several values, and lack standardisation. Thus, a first step in the address matching work flow is to increase the quality of input data. In this way we increase the accuracy, completeness and consistency of our address records, which increases the ease in which they can be linked by the techniques we apply later on. Typically this stage begins by parsing the text representations of addresses into rows of a dataframe.

[4]:
```
# load matched addresses, remove comment lines and reshape into two columns
data = pd.read_csv(f, comment='#',
                   header=None,
                   names=['address']).values.reshape(-1, 2)

matched_address = pd.DataFrame(data, columns=['addr_zagat', 'addr_fodor'])
```

[5]:
```
print('{} matched addresses loaded.'.format(matched_address.shape[0]))
matched_address.head(10).style.set_table_styles(styles)
```

Table 1: Output produced by code 5

	addr_zagat	addr_fodor
0	Arnie Morton's of Chicago 435 S. La Cienega Blvd. Los Angeles 90048 310-246-1501 Steakhouses	Arnie Morton's of Chicago 435 S. La Cienega Blvd. Los Angeles 90048 310/246-1501 American
1	Art's Deli 12224 Ventura Blvd. Studio City 91604 818-762-1221 Delis	Art's Delicatessen 12224 Ventura Blvd. Studio City 91604 818/762-1221 American
2	Bel-Air Hotel 701 Stone Canyon Rd. Bel Air 90077 310-472-1211 Californian	Hotel Bel-Air 701 Stone Canyon Rd. Bel Air 90077 310/472-1211 Californian
3	Cafe Bizou 14016 Ventura Blvd. Sherman Oaks 91423 818-788-3536 French Bistro	Cafe Bizou 14016 Ventura Blvd. Sherman Oaks 91423 818/788-3536 French
4	Campanile 624 S. La Brea Ave. Los Angeles 90036 213-938-1447 Californian	Campanile 624 S. La Brea Ave. Los Angeles 90036 213/938-1447 American
5	Chinois on Main 2709 Main St. Santa Monica 90405 310-392-9025 Pacific New Wave	Chinois on Main 2709 Main St. Santa Monica 90405 310/392-9025 French
6	Citrus 6703 Melrose Ave. Los Angeles 90038 213-857-0034 Californian	Citrus 6703 Melrose Ave. Los Angeles 90038 213/857-0034 Californian
7	Fenix at the Argyle 8358 Sunset Blvd. W. Hollywood 90069 213-848-6677 French (New)	Fenix 8358 Sunset Blvd. West Hollywood 90069 213/848-6677 American
8	Granita 23725 W. Malibu Rd. Malibu 90265 310-456-0488 Californian	Granita 23725 W. Malibu Rd. Malibu 90265 310/456-0488 Californian
9	Grill The 9560 Dayton Way Beverly Hills 90210 310-276-0615 American (Traditional)	Grill on the Alley 9560 Dayton Way Los Angeles 90210 310/276-0615 American

[5]:
```
112 matched addresses loaded.

Output in Table 1
```

A series of data cleaning exercises will then modify the data in ways that support the application of the linkage techniques. This might involve writing data cleaning scripts that convert all letters to lowercase characters, delete leading and trailing whitespaces, remove unwanted characters and tokens such as punctuation, or using hard-coded look-up tables to find and replace particular tokens. All together coding these steps contributes towards a standard form between the two address databases the user is attempting to match. This is important because standards between the two sources of address data under consideration will typically differ due to different naming conventions.

In the following cell blocks, we execute these steps by standardising our addresses. More specifically, we remove non-address components, convert all text to lower case and remove punctuation and non-alphanumeric characters.

[6]:
```
# our rows contain non-address components such as phone number and
# restaurant type so lets parse these using regular expressions into new columns
zagat_pattern = r"(?P<address>.*?)(?P<phone_number>\b\d{3}\-\d{3}\-\d{4}\b)(?
...P<category>.*$)"
fodor_pattern = r"(?P<address>.*?)(?P<phone_number>\b\d{3}\/\d{3}\-\d{4}\b)(?
...P<category>.*$)"

matched_address[["addr_zagat", "phone_number_zagat", "category_zagat"]] =
...matched_address["addr_zagat"].str.extract(zagat_pattern)
matched_address[["addr_fodor", "phone_number_fodor", "category_fodor"]] =
...matched_address["addr_fodor"].str.extract(fodor_pattern)
```

[7]:
```
# standardise dataframe by converting all strings to lower case
matched_address = matched_address.applymap(lambda row : row.lower() if type(row) ==
... str else row)

# remove punctuation and non-alphanumeric characters
matched_address['addr_zagat'] = matched_address['addr_zagat'].str.replace('[^\w\s]','')
matched_address['addr_fodor'] = matched_address['addr_fodor'].str.replace('[^\w\s]','')
```

3.1 Segmentation of address string into field columns

After removing unwanted characters and tokens, our next step is to segment the entire address string into tagged attribute values. Addresses rarely come neatly formatted into sensible fields that identify each component, and so segmentation is a vital and often overlooked stage of the work flow. For example, an address might come in an unsegmented format such as "19 Water St. New York 11201". Our objective is then to segment (or label) this address into the appropriate columns for street number, street name, city and postcode. When we segment both sets of addresses from the datasets we intend to link, we build well-defined output fields that are suitable for matching.

In our case we use a statistical segmentation tool called **Libpostal** which is a Conditional Random Fields (CRFs) model trained on OpenStreetMap addresses. Before using the Python bindings, users are required to install the Libpostal C library first (see https://github.com/openvenues/pypostal#installation for installation instructions). CRFs are popular methods in natural language processing (NLP) for predicting sequence of labels across sequences of text inputs. Unlike discrete classifiers, CRFs model the probability of a transition between labels on "neighbouring" elements, meaning they take into account past and future address field states into the labelling of addresses into address fields. This mitigates a limitation of segmentation models such as hidden markov models (HMMs) called the *label bias problem*: "transitions leaving a given state to compete only against each other, rather than against all transitions in the model" (Lafferty et al. 2001). Take, for example, the business address for "1st for Toys, 244 Ponce de Leon Ave. Atlanta 30308". A naive segmentation model would incorrectly parse "1st" as a property number, whereas it actually completes the business name "1st for Toys", leading to an erroneous sequence of label predictions. When a CRFs has parsed "1st" and reaches the second token, "for", the model scores an $l \times l$ matrix where l is the maximal number of labels (or address fields) that can be assigned by the CRFs. In L, l_{ij} reflects the probability of the current word being labelled as i and the previous word labelled j (Diesner, Carley 2008). In a CRFs model, when the parser reaches the *actual* property number, "244", high scoring in the matrix indicates the current label should be a property number, and the previous label revised to a business name. For a more detailed account, see Comber, Arribas-Bel (2019).

To segment each address, we apply the `parse_address` function row-wise for both the Zagat and Fodors addresses. This generates a list of tuples (see below code block for an example of the first two addresses from the Zagat dataset) that we convert into dictionaries before finally reading these into a `pandas` dataframe.

[8]:
```
[[('arnie mortons of chicago', 'house'),
  ('435', 'house_number'),
  ('s la cienega blvd', 'road'),
  ('los angeles', 'city'),
  ('90048', 'postcode')],
 [('arts deli', 'house'),
  ('12224', 'house_number'),
  ('ventura blvd', 'road'),
  ('studio city', 'city'),
  ('91604', 'postcode')]]
```

[8]:
```
[[('arnie mortons of chicago', 'house'),
  ('435', 'house_number'),
  ('s la cienega blvd', 'road'),
  ('los angeles', 'city'),
  ('90048', 'postcode')],
 [('arts deli', 'house'),
  ('12224', 'house_number'),
  ('ventura blvd', 'road'),
  ('studio city', 'city'),
  ('91604', 'postcode')]]
```

[9]:
```
# parse address string using libpostal CRF segmentation tool
addr_zagat_parse = [parse_address(addr, country='us') for addr in
... matched_address.addr_zagat]
addr_fodor_parse = [parse_address(addr, country='us') for addr in
... matched_address.addr_fodor]

# convert to pandas dataframe
addr_zagat_parse = pd.DataFrame.from_records([{k: v for v, k in row} for row in
... addr_zagat_parse]).add_suffix('_zagat')
addr_fodor_parse = pd.DataFrame.from_records([{k: v for v, k in row} for row in
... addr_fodor_parse]).add_suffix('_fodor')

# vertical join of CRF-parsed addresses between both dataframes
matched_address = matched_address.join(addr_zagat_parse).join(addr_fodor_parse)
```

Given we know the match status of our training data, we can safely join the records back together once we have successfully segmented them. Moreover, as we know the match status in advance, we can assign unique IDs that we will use later to create a binary variable for indicating whether an address pair is matched or non-matched.

[10]:
```
# create unique ID for matched addresses, these will be used later to create a match
 status
uids = [str(uuid.uuid4()) for i in matched_address.iterrows()]

# the following two lines will assign the same uid to both columns, thus facilitating
 a match
addr_zagat_parse['uid'], addr_fodor_parse['uid'] = uids, uids
match_ids = pd.DataFrame({'zagat_id' : addr_fodor_parse['uid'], 'fodor_id' :
... addr_fodor_parse['uid']})
```

[11]:
```
# join match ids to main dataframe
matched_address = matched_address.join(match_ids)

# preview of our parsed dataframe with uids assigned
matched_address.head().style.set_table_styles(styles)
```

[11]:
```
Output in Table 2
```

4 Creation of candidate address pairs using a 'full index'

Once our addresses have met a particular standard of quality and are segmented into the desired address fields, the next step requires us to create candidate pairs of addresses that potentially resolve to the same address. In record linkage, this step is typically called indexing or blocking, and is required to reduce the number of address pairs that are compared. In doing so we remove pairs that are unlikely to resolve to true matches. To demonstrate the utility of blocking and why it is so important to address matching, we first create a **full index** which creates all possible combinations of address pairs. More concretely, a full index generates the Cartesian product between both sets of addresses. Conditional on the size of both dataframes, full blocking is highly computationally inefficient, and in our case we create $112 \times 112 = 12544$ candidate links; this has a complexity of $O(n^2)$. We demonstrate the full index method to motivate the desire for practitioners to implement more sophisticated blocking techniques.

4.1 Full index

Below, we instantiate an `Index` class before specifying the desired full index method for generating pairs of records. We then create the Cartesian join between the Zagat and Fodor addresses which creates a `MultiIndex` that links every Zagat address with every Fodor address.

Table 2: Output produced by code 11

Nr	addr_zagat	addr_fodor	phone_number_zagat	category_zagat	phone_number_fodor	category_fodor	city_zagat	city_district_zagat
0	arnie mortons of chicago 435 s la cienega blvd los angeles 90048	arnie mortons of chicago 435 s la cienega blvd los angeles 90048	310-246-1501	steakhouses	310/246-1501	american	los angeles	nan
1	arts deli 12224 ventura blvd studio city 91604	arts delicatessen 12224 ventura blvd studio city 91604	818-762-1221	delis	818/762-1221	american	studio city	nan
2	belair hotel 701 stone canyon rd bel air 90077	hotel belair 701 stone canyon rd bel air 90077	310-472-1211	californian	310/472-1211	californian	nan	nan
3	cafe bizou 14016 ventura blvd sherman oaks 91423	cafe bizou 14016 ventura blvd sherman oaks 91423	818-788-3536	french bistro	818/788-3536	french	sherman oaks	nan
4	campanile 624 s la brea ave los angeles 90036	campanile 624 s la brea ave los angeles 90036	213-938-1447	californian	213/938-1447	american	los angeles	nan

Continued (additional columns) on next page

Table 2 – *Continued from previous page*

Nr	house_zagat	house_number_zagat	postcode_zagat	road_zagat	suburb_zagat	city_fodor	city_district_fodor	house_fodor
0	arnie mortons of chicago	435	90048	s la cienega blvd	nan	los angeles	nan	arnie mortons of chicago
1	arts deli	12224	91604	ventura blvd	nan	studio city	nan	arts delicatessen
2	belair hotel	701	90077	stone canyon rd bel air	nan	nan	nan	hotel belair
3	cafe bizou	14016	91423	ventura blvd	nan	sherman oaks	nan	cafe bizou
4	campanile	624	90036	s la brea ave	nan	los angeles	nan	campanile

Nr	house_number_fodor	postcode_fodor	road_fodor	state_fodor	suburb_fodor
0	435	90048	s la cienega blvd	nan	nan
1	12224	91604	ventura blvd	nan	nan
2	701	90077	stone canyon rd bel air	nan	nan
3	14016	91423	ventura blvd	nan	nan
4	624	90036	s la brea ave	nan	nan

Nr	zagat_id	fodor_id
0	99bbcd03-ce45-40b5-907f-47f5ae16ae29	99bbcd03-ce45-40b5-907f-47f5ae16ae29
1	1b1e1ee1-c880-4722-abaa-4ec44c7d94a6	1b1e1ee1-c880-4722-abaa-4ec44c7d94a6
2	f2548f68-2326-4706-bdc1-dbfc265ecbf3	f2548f68-2326-4706-bdc1-dbfc265ecbf3
3	936687e2-1161-4ecd-98c3-b5ac620d8776	936687e2-1161-4ecd-98c3-b5ac620d8776
4	20a05006-d08e-4245-b014-ab2d0f552662	20a05006-d08e-4245-b014-ab2d0f552662

[12]:
```
indexer = rl.Index()
indexer.full()

# create cartesian join between zagat and fodor restaurant addresses
candidate_links = indexer.index(matched_address.city_zagat, matched_address.city_fodor)
```

[12]:
```
WARNING:recordlinkage:indexing - performance warning - A full index can result in large
number of record pairs.
```

[13]:
```
# this creates a two-level multiindex, so we name addresses from the zagat and fodor
 databases, respectively.
candidate_links.names = ['zagat', 'fodor']

print('{} candidate links created using full indexing.'.format(len(candidate_links)))
```

[13]:
```
12544 candidate links created using full indexing.
```

In practice, a full index creates a dataframe with 12,544 rows and thus creates candidate address pairs between every possible combination of address from both the Zagat and Fodor datasets. Once we generate this dataframe of potential matches, we create a match status column and assign a 1 to actual matched addresses and 0 to non-matches based on the unique IDs created earlier.

[14]:
```
# lets create a function we can reuse later on
def return_candidate_links_with_match_status(candidate_links):

    # we return a vector of label values for both the zagat and fodor restaurant
    IDs from the multiindex
    zagat_ids = candidate_links.get_level_values('zagat')
    fodor_ids = candidate_links.get_level_values('fodor')

    # now we create a new dataframe as long as the number of candidate links
    zagat = matched_address.loc[zagat_ids][['city_zagat', 'house_zagat',\
                                            'house_number_zagat', 'road_zagat',\
                                            'suburb_zagat', 'zagat_id']]
    fodor = matched_address.loc[fodor_ids][['city_fodor','house_fodor',\
                                            'house_number_fodor', 'road_fodor',\
                                            'suburb_fodor', 'fodor_id']]

    # vertically concateate addresses from both databases
    candidate_link_df = pd.concat([zagat.reset_index(drop=True),
    ... fodor.reset_index(drop=True)], axis=1)

    # next we create a match status column that we will use to train a machine
     learning model
    candidate_link_df['match_status'] = np.nan

    # assign 1 for matched, 0 non-matched
    candidate_link_df.loc[candidate_link_df['zagat_id'] ==
    ... candidate_link_df['fodor_id'], 'match_status'] = 1.
    candidate_link_df.loc[ ~(candidate_link_df['zagat_id'] ==
    ... candidate_link_df['fodor_id']), 'match_status'] = 0.

    return candidate_link_df

candidate_link_df = return_candidate_links_with_match_status(candidate_links)
```

4.2 Creation of comparison vectors from indexed addresses

To resolve addresses into matches and non-matches we generate comparison vectors between each candidate address pair. Each element of this comparison vector is a similarity metric used to assess the closeness of two address fields. In our case, we use **Jaro-Winkler similarity** because it has been observed to perform best on attributes containing named values (e.g., property names, street names, or city names) (Christen

2012, Yancey 2005). The Jaro similarity of two given address components a_1 and a_2 is given by

$$jaro\_sim = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|a_1|} + \frac{m}{|a_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

where $|a_i|$ is the length of the address component string a_i, m is the number of matching characters, and t is the number of transpositions required to match the two address components. We will create a function that makes use of the `jellyfish` implementation of Jaro-winkler similarity. Several other string similarity metrics are available and are optimised for particular use cases and data types. See Chapter 5 of Christen (2012) for an excellent overview.

[15]:
```
def jarowinkler_similarity(s1, s2):

    conc = pd.concat([s1, s2], axis=1, ignore_index=True)

    def jaro_winkler_apply(x):

        try:
            return jellyfish.jaro_winkler(x[0], x[1])
        # raise error if fields are empty
        except Exception as err:
            if pd.isnull(x[0]) or pd.isnull(x[1]):
                return np.nan
            else:
                raise err

    # apply row-wise to concatenated columns
    return conc.apply(jaro_winkler_apply, axis=1)
```

Before applying Jaro-Winkler similarity we need to choose columns that were segmented in **both** the Zagat and Fodor datasets.

[16]:
```
# lets take a look at the columns we have available
candidate_link_df.columns
```

[16]:
```
Index(['city_zagat', 'house_zagat', 'house_number_zagat', 'road_zagat',
       'suburb_zagat', 'zagat_id', 'city_fodor', 'house_fodor',
       'house_number_fodor', 'road_fodor', 'suburb_fodor', 'fodor_id',
       'match_status'],
      dtype='object')
```

As we can only match columns that were parsed in both address datasets, this means we lose two columns, `city_district_zagat` and `state_fodor`, that were parsed by the CRF segmentation model. Once we observe which address fields are common to both datasets, we create so-called comparison vectors from candidate address pairs of the Zagat and Fodor datasets. Each element of the comparison vector represents the string similarity between address fields contained in both databases. For example, `city_jaro` describes the string similarity between the columns `city_zagat` and `city_fodor`. Looking at the first two rows of our comparison vectors dataframe, a `city_jaro` value of 1.00 implies an exact match whereas a value of 0.4040 implies a number of modifications are required to match the two city names, and so these are less likely to correspond to a match.

[17]:
```
# create a function for building comparison vectors we can reuse later
def return_comparison_vectors(candidate_link_df):

   candidate_link_df['city_jaro'] = jarowinkler_similarity(candidate_link_df.
   ...city_zagat, candidate_link_df.city_fodor)
   candidate_link_df['house_jaro'] = jarowinkler_similarity(candidate_link_df.
   ...house_zagat, candidate_link_df.house_fodor)
   candidate_link_df['house_number_jaro'] = jarowinkler_similarity(candidate_link_df.
   ...house_number_zagat, candidate_link_df.house_number_fodor)
```

Table 3: Output of code 17

	city_jaro	house_jaro	house_number_jaro	road_jaro	suburb_jaro	match_status
0	1	1	1	1	0	1
1	0.40404	0.568301	0	0.629085	0	0
2	0	0.482143	0	0.674077	0	0
3	0.626263	0.502778	0.511111	0.629085	0	0
4	1	0.45463	0	0.831493	0	0

```
    candidate_link_df['road_jaro'] = jarowinkler_similarity(candidate_link_df.
    ...road_zagat, candidate_link_df.road_fodor)
    candidate_link_df['suburb_jaro'] = jarowinkler_similarity(candidate_link_df.
    ...suburb_zagat, candidate_link_df.suburb_fodor)

    # now we build a dataframe that contains the jaro-winkler similarity between the
    address components and the matching status
    comparison_vectors = candidate_link_df[['city_jaro', 'house_jaro',\
                                            'house_number_jaro','road_jaro',\
                                            'suburb_jaro', 'match_status']]

    # set NaN values to 0 so the comparison vectors can work with the applied classifiers
    comparison_vectors = comparison_vectors.fillna(0.)

    return comparison_vectors

comparison_vectors = return_comparison_vectors(candidate_link_df)

# lets preview this dataframe to build some intution as to how it looks
comparison_vectors.head().style.set_table_styles(styles)
```

[17]:
```
Output in Table 3
```

4.3 *Classification and evaluation of match performance*

Once we obtain comparison vectors for each candidate address pair, we frame our approach as a binary classification problem by resolving the vectors into matches and non-matches. As the Zagat and Fodors dataframe has labels that describe our address pairs as matched, we use supervised classification to train a statistical model, a **random forest**, to classify address pairs with an unknown match status into matches and non-matches. As a reminder, a random forest is generated using a multitude of decision trees during training which then outputs the mode of the match status decision for the individual trees.

In practice, we initialize a random forest object and split our `comparison_vectors` dataframe into features containing our Jaro-Winkler string similarity features, X, and a vector used to predict match status of the addresses, y.

[18]:
```
# create a random forest classifier that uses 100 trees and number of cores equal to
those available on machine
rf = RandomForestClassifier(n_estimators = 100,
                            # Due to small number of features (5) we do not limit
                            # depth of trees
                            max_depth = None,
                            # max number of features to evaluate split is
                            # sqrt(n_features)
                            max_features = 'auto',
                            n_jobs = os.cpu_count())

# define metrics we use to assess the model
scoring = ['precision', 'recall', 'f1']
folds = 10

# extract the jaro-winkler string similarity and match label
X = comparison_vectors.iloc[:, 0:5]
y = comparison_vectors['match_status']
```

To evaluate the performance of our built classification model, we use 10-fold cross-validation meaning the performance measures are averaged across the test sets used within the 10 folds. We use three metrics that are commonly used to evaluate machine learning models. Recall measures the proportion of address pairs that should have been classified, or recalled, as matched (Christen 2012). The precision (or, equivalently, the positive predictive value) calculates the proportion of the matched address pairs that are classified correctly as true matches (Christen 2012). Finally, the F1 score reflects the harmonic mean between precision and recall. Our cross-validation exercise is executed in the following cell.

[19]:
```
# 10-fold cross-validation procedure
scores = cross_validate(estimator = rf,
                X = X,
                y = y,
                cv = folds,
                scoring = scoring,
                return_train_score = False)
```

[20]:
```
print('Mean precision score is {} over {} folds.'.format( np.round(np.
...mean(scores['test_precision']), 4), folds))
print('Mean recall score is {} over {} folds.'.format( np.round(np.
...mean(scores['test_recall']), 4), folds))
print('Mean F1 score is {} over {} folds.'.format( np.round(np.
...mean(scores['test_f1']), 4), folds))
```

[20]:
```
Mean precision score is 0.9546 over 10 folds.
Mean recall score is 0.928 over 10 folds.
Mean F1 score is 0.9383 over 10 folds.
```

Overall, the high precision value implies that 95% of true positives are successfully disambiguated from false positives. Moreover, our recall value implies that 93% of all potential matches were successfully returned, with the remaining 7% of correct matches incorrectly labelled as false negatives. Given the high values in both of these metrics, the accompanying F1 score is equally high.

5 Creation of candidate address pairs by blocking on zipcode

While a Cartesian product could be useful in a linkage exercise where we have a very small number of matched addresses, in production environments more sophisticated techniques are generally required to create candidate address links. This is particularly the case when we have a large number of addresses. Thus, blocking is typically introduced to partition the set of all possible address comparisons to within mutually exclusive blocks. If we let b equal the number of blocks, we reduce the complexity of the comparison exercise to $O(\frac{n^2}{b})$, which is far more computationally tractable than the full index method used above.

When deciding which column to use as a blocking key we generally need pay attention to two main considerations. Firstly, we pay attention to attribute data quality. Typically when identifying a blocking key, we choose a key that has a **low number of missing values**. This is because choosing a key with many missing values forces a large number of addresses into a block where the key is an empty value, which may lead to many misclassified address matches. And secondly we pay attention to the **frequency distribution** of attribute values. We optimise towards a uniform distribution of values, as typically skewed distributions that result in some values occurring very frequently mean that these values will dominate the candidate pairs of address generated.

These considerations are addressed in the following two code blocks.

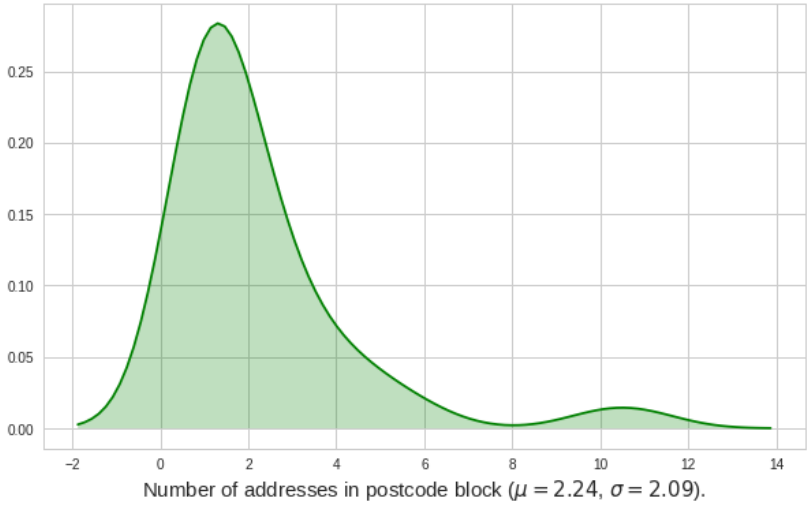


Figure 1: Figure generated by code 22

[21]:
```
print("Missing postcodes for Zagat addresses: {}. \n
Missing postcodes for Fodor addresses: {}.".format(matched_address.postcode_zagat.
...isnull().sum(), matched_address.postcode_fodor.isnull().sum()))
```

[21]:
```
Missing postcodes for Zagat addresses: 1.
Missing postcodes for Fodor addresses: 0.
```

[22]:
```
# check distribution of postcode blocks
pc_dist = matched_address.groupby('postcode_fodor').size().to_frame().
...rename(columns={0:'n_addresses'})

f, ax = plt.subplots(1, figsize=(10,6))
sns.kdeplot(pc_dist.n_addresses.values, color='g', shade=True, legend=False)
ax.set_xlabel('Number of addresses in postcode block ($\mu = {}$, $\sigma = {}$).'
              .format(np.mean(pc_dist.n_addresses), np.round(np.std(pc_dist.n_addresses),
              2)), size=15)
plt.show()
```

[22]:
```
Output in Figure 1
```

The postcode attribute looks like a sensible choice of blocking key because it contains just one missing value and there are very low numbers of candidate address comparisons within each block. As you can see in the output below, when we use a more sophisticated indexing technique we generate a far lower number of candidate address comparisons. In fact, we create only 1014 candidate address links despite adding synthetic non-matches (discussed below). Overall, our introduction of blocking substantially lowers the computational requirement of the linkage task.

5.1 Creation of synthetic non-matched addresses

To make this exercise more realistic, let's also create 112 synthetic non-matches so we have 224 addresses in total. This will also be important for training our machine learning technique to learn the representations of non-matched addresses in addition to matches. In this case we use the FEBRL data set generator script `generate.py` to create an artificially generated dataset (see http://users.cecs.anu.edu.au/ Peter.Christen/Febrl/febrl-0.3/febrldoc-0.3/node70.html). The script uses Python 2.7, so we read the output as JSON so the user does not have to rely on an external input. We do this in keeping with

a self-contained notebook but describe the steps required to reproduce the non-matches below.

The synthetic non-matches are essentially random permutations of the matched addresses. These are constructed on the basis of frequency tables for each address field that count the occurrence of particular values. For example, the first row of a frequency table for a house number would look like:

<house_number_attribute_value>,<frequency_of_occurence>.

[23]:
```
# first we need columns from the zagat and fodor databases to create random addresses
zagat_cols = ['city_zagat', 'house_number_zagat',\
              'house_zagat', 'suburb_zagat',\
              'road_zagat', 'postcode_zagat']
fodor_cols = ['city_fodor', 'house_number_fodor',\
              'house_fodor', 'suburb_fodor',\
              'road_fodor','postcode_fodor']

# create a directory for address component frequencies
if not os.path.exists('freqs'):
    os.makedirs('freqs')

# create distributions of address components for both datasets that will be used to
# create fake addresses
for cols in [zagat_cols, fodor_cols]:
    for col in cols:
        freq = matched_address[col].value_counts().reset_index()
        freq.to_csv('freqs/{}_freq.csv'.format(col), index=False, header=False)
```

The script `generate.py` takes six parameters that are used to create non-matched addresses. The first argument demarcates the number of original records to be generated; the second specifies the number of duplicate records from the original to be generated; and the third, fourth and fifth arguments define the maximal number of duplicate records that will be created based on one original record, the maximum number of modifications introduced to the address field, and the maximum number of modifications introduced to the address, respectively. The final parameter is used to enter which probability distribution will be to create duplicate records – i.e. uniform, poisson, or zipf. In our case we are only interested in building synthetic non-matches (and not duplicates), so we set the number of original records to be built as 112, the number of duplicates generated as 0, and leave the number of modifications introduced by the recommended default settings.

In addition, for each address field, users are asked to define a dictionary inside `generate.py` that outlines the probability for particular modifications. This includes setting the probability of modifications such as misspellings, insertions, deletions, substitutions and transpositions of word and characters. An example dictionary for the house number address field is given below where we set the file path to the word frequency CSV generated above:

[24]:
```
house_number_dict = {'name':'house_number',
                     'type':'freq',
                'char_range':'digit',
               # 'freq_file':'freqs/house_number_fodor_freq.csv',
                'freq_file':'freqs/house_number_zagat_freq.csv',
               'select_prob':0.20,
                  'ins_prob':0.10,
                  'del_prob':0.16,
                  'sub_prob':0.54,
                'trans_prob':0.00,
             'val_swap_prob':0.00,
             'wrd_swap_prob':0.00,
              'spc_ins_prob':0.00,
              'spc_del_prob':0.00,
                 'miss_prob':0.00,
              'new_val_prob':0.20}
```

Damerau (1964) finds the proportions of typographical errors are typically spread as substitutions (59%), deletions (16%), transpositions (2%), insertions (10%) and multiple errors (13%). For this reason we broadly align our dictionary probabilities with these findings. After defining sensible probabilities for modifications, we execute the following scripts on a terminal which will create a file, `zagat_synthetic_addresses.csv` and `fodor_synthetic_addresses.csv` consisting of synthetic addresses from the Zagat and Fodor datasets, respectively.

For simplicity we generate our non-matches using all the data at once. However, in a real-world application, we might wish to create non-matches within each zipcode block one at a time. This would create more realistic synthetic non-matches. This is because non-matched addresses would be constructed from the frequency tables of each zipcode block, meaning each non-match would share more commonality to actual matched addresses. In practice, this would improve the predictive power of our classification model to disambiguate between candidate address pairs that have very subtle differences yet are still matched or non-matched.

[25]:
```
# ! python2 generate.py zagat_synthetic_addresses.csv 112 0 4 2 2 poisson
```

[25]:
```
Create 112 original and 0 duplicate records
  Distribution of number of duplicates (maximal 4 duplicates):
  [(1, 0.0), (2, 0.375), (3, 0.75), (4, 0.9375)]

Step 1: Load and process frequency tables and misspellings dictionaries

Step 2: Create original records


Step 2: Create duplicate records


Step 3: Write output file
End.
```

[26]:
```
# ! python2 generate.py fodor_synthetic_addresses.csv 112 0 4 2 2 poisson
```

[26]:
```
Create 112 original and 0 duplicate records
  Distribution of number of duplicates (maximal 4 duplicates):
  [(1, 0.0), (2, 0.375), (3, 0.75), (4, 0.9375)]

Step 1: Load and process frequency tables and misspellings dictionaries

Step 2: Create original records


Step 2: Create duplicate records


Step 3: Write output file
End.
```

We then read these synthetic non-matches into a dataframe.

[27]:
```
# read parsed synthetic addresses
synthetic_zagat_address = pd.read_csv('zagat_synthetic_addresses.csv').
...add_suffix('_zagat').drop(columns=['rec_id_zagat'])
synthetic_fodor_address = pd.read_csv('fodor_synthetic_addresses.csv').
...add_suffix('_fodor').drop(columns=['rec_id_fodor'])
```

```
# set uids for synthetic addresses
synthetic_zagat_address['zagat_id'] = [str(uuid.uuid4()) for i in
...synthetic_zagat_address.iterrows()]
synthetic_fodor_address['fodor_id'] = [str(uuid.uuid4()) for i in
...synthetic_fodor_address.iterrows()]

# join synthetic zagat and fodor addresses vertically
synthetic_non_matches = synthetic_zagat_address.join(synthetic_fodor_address)

# remove whitespace from column names and attributes
synthetic_non_matches = synthetic_non_matches.rename(columns = lambda x : x.strip())
synthetic_non_matches = synthetic_non_matches.applymap(lambda x : x.strip() if
...type(x) == str else x)
```

Now we have generated synthetic non-matches, we need to join these back to our dataframe of matched addresses. As the above steps require external scripts we provide the JSON required to reconstruct the synthetic dataframe in the dedicated Github repository. This can be read by executing the cell below which uses the `pd.read_json` function.

[28]:
```
! wget https://raw.githubusercontent.com/SamComber/address_matching_workflow/master/
...synthetic_addresses.json
```

[28]:
```
--2019-12-21 09:11:11--  https://raw.githubusercontent.com/SamComber/address_matching_
workflow/master/synthetic_addresses.json
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.56.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.56.133|:443
... connected.
HTTP request sent, awaiting response... 200 OK
Length: 29098 (28K) [text/plain]
Saving to: ‘synthetic_addresses.json’

synthetic_addresses 100%[===================>]  28.42K  --.-KB/s    in 0.02s

2019-12-21 09:11:11 (1.16 MB/s) - ‘synthetic_addresses.json’ saved [29098/29098]
```

[29]:
```
f = 'synthetic_addresses.json'

synthetic_non_matches = pd.read_json(f)
```

In the cell below we join our matched addresses with our synthetic non-matches, creating a dataframe of 224 address pairs.

[30]:
```
# align columns of matched_address dataframe for horizontal join
matched_address = matched_address[['house_zagat', 'house_number_zagat', 'road_zagat',
... 'suburb_zagat', 'city_zagat', 'postcode_zagat','zagat_id', 'house_fodor',
... 'house_number_fodor', 'road_fodor', 'suburb_fodor', 'city_fodor', 'postcode_fodor',
... 'fodor_id']]

# horizontal join between matched addresses and synthetic non-matches
matches_with_non_matches = pd.concat([matched_address, synthetic_non_matches],
... ignore_index=True)

print('{} address pairs created consisting of {} matches and {} synthetic non-matches.'.
...format(matches_with_non_matches.shape[0], matched_address.shape[0],
...       synthetic_non_matches.shape[0]))
```

[30]:
```
224 address pairs created consisting of 112 matches and 112 synthetic non-matches.
```

5.2 *Blocking on postcode attribute*

With our matches and synthetic non-matches assembled into a dataframe with 224 address pairs, we can proceed to block on postcode values to create mutually exclusive address partitions. Thus, for every unique postcode value, a dataframe (or block) will be created in which candidate address pairs will be matched and non-matched based on attributes of their comparison vectors.

The following code block creates a `MultiIndex` that links together the IDs of addresses that are within the same zipcode block.

[31]:
```
indexer = rl.Index()

# block on postcode attribute
indexer.block(left_on='postcode_zagat', right_on='postcode_fodor')
candidate_links = indexer.index(matches_with_non_matches, matches_with_non_matches)

# this creates a two-level multiindex, so we name addresses from the zagat and fodor
databases, respectively.
candidate_links.names = ['zagat', 'fodor']

print('{} candidate links created using the postcode attribute as a blocking key.'.
...format(len(candidate_links)))
```

[31]:
```
1014 candidate links created using the postcode attribute as a blocking key.
```

We follow the same work flow as before and create comparison vectors for every 1014 candidate address links.

[32]:
```
candidate_link_df = return_candidate_links_with_match_status(candidate_links)

comparison_vectors = return_comparison_vectors(candidate_link_df)
```

Following this, we train our random forest on the comparison vectors and match status labels. We use a 75/25 split for our train and test data.

[33]:
```
X = comparison_vectors.iloc[:, 0:5]
y = comparison_vectors.match_status

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# create a random forest classifier that uses 100 trees and number of cores equal to
those available on machine
rf = RandomForestClassifier(n_estimators = 100,
                            # Due to small number of features (5) we do not limit
                            # depth of trees
                            max_depth = None,
                            # max number of features to evaluate split is
                            # sqrt(n_features)
                            max_features = 'auto',
                            n_jobs = os.cpu_count())

# predict match status of unseen address pairs
y_pred = rf.fit(X_train, y_train).predict(X_test)
```

5.3 *Classification and evaluation of match performance*

Having fit our random forest on the training data we can now assess the model under the number of metrics we introduced earlier. We can also produce a confusion matrix which shows true negatives in the top-left quadrant, false positives in the top-right, false negatives in the bottom-left and true positives in the bottom-right. At first glance, the

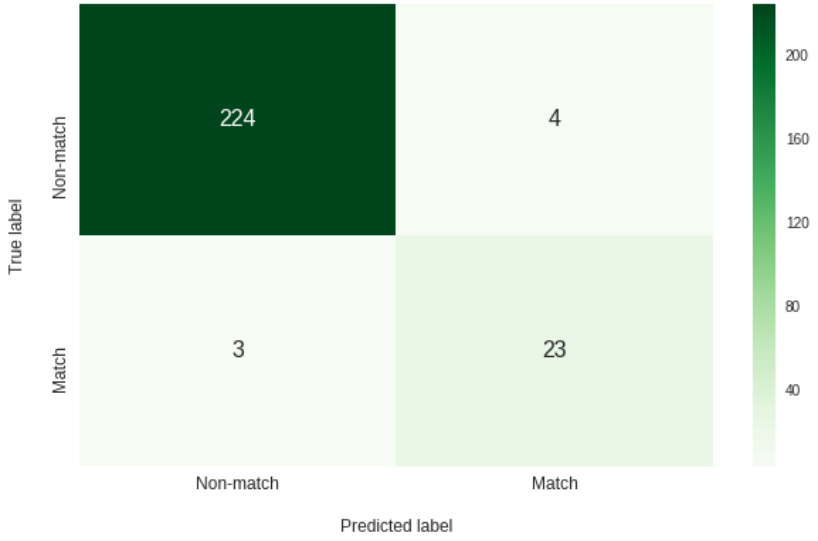


Figure 2: Figure generated by code 34

findings from the evaluation metrics below may seem counter-intuitive, especially as the results of the classification exercise using the full index performed better. However, it is pertinent to remind ourselves that we trained our classification model on **matched address only**, which reflected an idealised but unrealistic scenario. In the results below we introduced synthetic non-matches which reflected a scenario that a user is more likely to encounter in a real-world address matching exercise.

In the following code block we generate evaluation metrics and a confusion matrix for evaluating match performance.

[34]:
```
print('Precision score: {}.'.format(np.round(precision_score(y_test, y_pred), 4)))
print('Recall score: {}.'.format(np.round(recall_score(y_test, y_pred), 4)))
print('F1 score: {}.'.format(np.round(f1_score(y_test, y_pred), 4)))

f,ax = plt.subplots(1, figsize=(10,6))
f.set_tight_layout(False)

fontsize=12
sns.heatmap(confusion_matrix(y_test, y_pred),
            ax=ax,
            annot=True,
            annot_kws={'fontsize': 16},
            cmap='Greens',
            fmt='g')
ax.set_yticklabels(['Match', 'Non-match'], fontsize=fontsize)
ax.set_xticklabels(['Non-match', 'Match'], fontsize=fontsize);
ax.set_ylabel('True label', fontsize=fontsize)
ax.set_xlabel('Predicted label', fontsize=fontsize)
ax.xaxis.labelpad = 18
ax.yaxis.labelpad = 18
plt.show();
```

[34]:
```
Precision score: 0.8519.
Recall score: 0.8846.
F1 score: 0.8679.

Output in Figure 2
```

Overall, our precision value implied 85% of true positives were correctly separated from false positives, and our recall value indicated that 88% of all true address matches were successfully retrieved, with the remaining 12% incorrectly classified as non-matches.

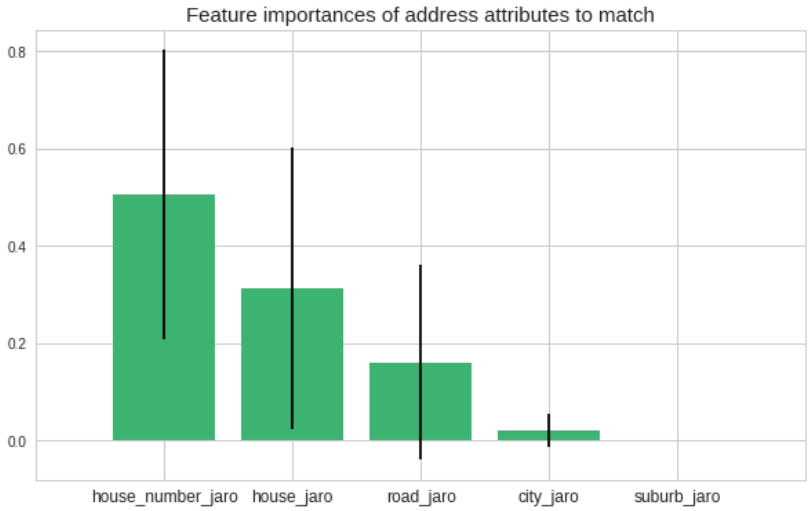


Figure 3: Output generated by code 35

With our model now fitted and tested, we could extend its use to predict the match status of unseen address pairs. As an example application, if we had a small sample of matched addresses that belonged to a larger set of unmatched addresses, we could use our trained predictive model to match the remaining addresses in the dataset. This would work so long as the textual representations of addresses used in the prediction stage follow a similar structure to those addresses used to train the classification model.

Before we conclude, a benefit of using ensemble methods such as random forest classifiers is that we can return an indication of how useful and valuable each feature was in the construction of each decision tree. In a practical application, extracting a measure of feature importance might be a useful step in pruning redundant features from the comparison vectors. This might be a useful step in lowering computation times as we decrease the number of address field comparisons required to evaluate candidate address pairs.

Thus, in the following code block we rank feature importance of particular address fields to the match classification.

[35]:
```
# extract feature importances from random forest classifier
feature_importance_to_match = rf.feature_importances_

# calculate standard deviation of feature importances across trees
std = np.std([tree.feature_importances_ for tree in rf.estimators_], axis=0)
indices = np.argsort(feature_importance_to_match)[::-1]

# plot importances alongside feature labels
plt.figure(figsize=(10,6))
plt.title("Feature importances of address attributes to match", size=15)
plt.bar(range(X_train.shape[1]), feature_importance_to_match[indices],
       color="#3CB371", yerr=std[indices], align="center")
feature_labs = X_train.columns[np.argsort(feature_importance_to_match)[::-1]].values
plt.xticks(range(X_train.shape[1]), feature_labs, size=12)
plt.xlim([-1, X_train.shape[1]])
plt.show()
```

[35]:
```
Output in Figure 3
```

In our case, and as one might expect, the restaurant's house number, `house_number_jaro` is the most important feature used for resolving candidate pairs of addresses into a match while the suburb, `suburb_jaro`, is the least important feature and so could possibly be removed as an address field from the comparison step.

6 Conclusion

Address matching is a data enrichment process that is increasingly required in wide-ranging, real-world applications. For example, matching between census, commercial or lifestyle records has the potential benefit of improving data quality, enabling spatial data visualisation and joining data that would otherwise remain isolated in data silos. In absence of unique identifiers for directly linking data, practitioners have typically relied on statistical linkage methods for matching addresses. Linking address datasets in this way has the potential to unlock attributes that one would be unable to access in circumstances where no primary keys exist to join the two datasets. Thus, in this notebook, we documented the steps required to execute the work flow for an address matching exercise that utilised new and recent innovations in machine learning. While the dataset we used was low volume, the intention of the notebook was to demonstrate an approach that is reproducible within a self-contained environment, and which might be adapted by the interested user to larger data challenges. Training a predictive model to link restaurant addresses may seem a trivial problem to solve, but these addresses could easily be replaced by more meaningful address records in areas such as public health and socio-economic mobility studies. Therefore, the core contribution of this notebook sought to equip the regional scientist with skills necessary to extend the address matching work flow to their own (and far more interesting) use cases.

References

Baldovin T, Zangrando D, Casale P, Ferrarese F, Bertoncello C, Buja A, Marcolongo A, Baldo V (2015) Geocoding health data with geographic information systems: A pilot study in northeast Italy for developing a standardized data-acquiring format. *Journal of Preventive Medicine & Hygiene* 56: 88–94

Cayo R, Talbot TO (2003) Positional error in automated geocoding of residential addresses. *International Journal of Health Geographics* 2: 1–10

Christen P (2012) *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection.* Springer, New York, NY

Comber S, Arribas-Bel D (2019) Machine learning innovations in address matching: A practical comparison of word2vec and CRFs. *Transactions in GIS* 23: 334–348

Damerau F (1964) A technique for computer detection and correction of spelling errors. *Commununications of the ACM* 7: 171–176. CrossRef.

Diesner J, Carley M (2008) Conditional random fields for entity extraction and ontological text coding. *Computational and Mathematical Organization Theory* 14: 248–262

Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In: Brodley CE, Danyluk AP (eds), *Proceedings of the 18th International Conference on Machine Learning.* Morgan Kaufmann, San Francisco, CA, 282–289

Reynolds P, Behren JV, Gunier R, Goldberg D, Hertz A, Smith D (2003) Childhood cancer incidence rates and hazardous air pollutants in California: An exploratory analysis. *Environmental Health Perspectives* 111: 663–668

Ruggles S, Fitch C, Roberts E (2018) Historical census record linkage. *Annual Review of Sociology* 44[1]: 19–37

Yancey W (2005) Evaluating string comparator performance for record linkage. Research report series, statistics #2005-05, Bureau of the Census, Washington, DC

REGION
The Journal of ERSA
Powered by WU

Volume 7, Number 2, 2020, R15–R46
DOI: 10.18335/region.v7i2.295

journal homepage: region.ersa.org

A reproducible notebook to acquire, process and analyse satellite imagery: Exploring long-term urban changes

Meixu Chen, Dominik Fahrner, Daniel Arribas-Bel, Francisco Rowe[1]

[1] University of Liverpool, Liverpool, UK

Received: 22 December 2019/Accepted: 1 December 2020

Abstract. Satellite imagery is often used to study and monitor changes in natural environments and the Earth surface. The open availability and extensive temporal coverage of Landsat imagery has enabled to monitor changes in temperature, wind, vegetation and ice melting speed for a period of up to 46 years. Yet, the use of satellite imagery to study cities has remained underutilised in Regional Science, partly due to the lack of a practical methodological approach to capture data, extract relevant features and monitor changes in the urban environment. This notebook offers a framework to demonstrate how to batch-download high-resolution satellite imagery; and enable the extraction, analysis and visualisation of features of the built environment to capture long-term urban changes.

Key words: satellite imagery, image segmentation, urbanisation, cities, urban change, computational notebooks

1 Introduction

Sustainable urban habitats are a key component of many global challenges. Efficient management and planning of cities are pivotal to all 17 UN Sustainable Development Goals (SDGs). Over 90% of the projected urban population growth by 2050 will occur in less developed countries (United Nations 2019). Concentrated in cities, this growth offers an opportunity for social progress and economic development but it also imposes major challenges for urban planning. Prior work on urbanisation has identified the benefits of agglomeration and improvements in health and education, which tend to outweigh the costs of congestion, pollution and poverty (Glaeser, Henderson 2017). Yet research has remained largely focused on Western cities (e.g. Burchfield et al. 2006), developing a good understanding of urban areas in high-income, developed countries (Glaeser, Henderson 2017). Much less is known about the long-term evolution of urban habitats in less developed countries. Analysis of historical census data exist exploring changes at discrete points over time such as slum detection (e.g. Giada et al. 2003, Kit, Lüdeke 2013, Kohli et al. 2016). Less applications can be identified tracking changes in urban settings over a continuous temporal scale (Ibrahim et al. 2020). This gap is partly due to the lack of comprehensive and consistent data sources capturing the long-term dynamics of urban structures in less developed countries.

Cities in Asia provide a unique setting to explore the challenges triggered by rapid urbanisation. The share of urban population in Asia is currently at a turning point transitioning to exceed the share of rural population. Currently Asia is home to over 53% of the urban population globally and the share of urban population is projected to increase to 66% by 2050 (United Nations 2019). Developing tools to monitor and understand the past and current urbanisation process is key to guide appropriate urban planning and policy strategies.

Recent technological developments can help overcome the paucity in spatially-detailed urban data in less developed countries. The combination of geospatial technology, cheap computing and new machine learning algorithms has ushered in an age of new forms of data, producing brand new data sets and repurposing existing sources. Satellite imagery represents a key source of information. Photographs from the sky have existed for decades, but their use in the context of socioeconomic urban research has been limited. Image data has been hard to process and understand for social scientists. Yet recent developments in machine learning and artificial intelligence have made images computable and turned these data into brand new information to be explored by quantitative urban researchers. Further, satellite data has become more abundant and openly accessible in the past decade, and offers new possibilities for data exploration through increasing spatial and temporal resolution. This, together with more computational power being available, allows to process these data in an efficient and meaningful way.

This notebook illustrates an easy-to-use analytical framework based on Python tools which enables batch download, image feature extraction, analysis and visualisation of high-resolution satellite imagery to capture long-term urban changes. Our purpose is to fill in the absence of a systematic and reproducible framework to acquire, process and analyse satellite imagery in urban built environment related to the field of Regional Science. The source of satellite data and administrative boundaries data are from NASA's Landsat satellite programme and ArcGIS Online. The Python libraries used in this notebook are the following:

- Landsat images in Google Cloud Storage: The Google Cloud Storage is accessed using an API to download Landsat imagery (version used: 0.4.9)
- Matplotolib:A Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Numpy: Adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions
- Pandas: Provides high-performance, easy-to-use data structures and data analysis tools
- GeoPandas: Python library that simplifies working with geospatial data (version used: 0.6.2)
- Folium: Python library that enables plotting interactive maps using leaflet (version used: 0.10.0)
- Glob: Unix style pathname pattern expansion
- GDAL: Library for geospatial data processing (version used: 2.4.4)
- Landsat578: Simple Landsat imagery download tool
- L8qa: Landsat processing toolbox (version used: 0.1.1)
- Rasterio: Library for raster data processing (version used: 1.1.3)
- Scikit-image: Collection of algorithms for image processing
- Wget: Pure python download utility (version used: 3.2)
- OpenCV: Library for image processing
- scikit-learn: Machine learning in Python. Simple and efficient tools for data mining and data analysis.

We can import them all as follows:

[1]:
```
%matplotlib inline

#load external libraries
import matplotlib.pyplot as plt
from matplotlib import colors
import pandas as pd
import numpy as np
import geopandas as gpd
import folium
import os, shutil
import glob
import gdal
import wget
from landsat import google_download
from google_download import GoogleDownload
from l8qa.qa import write_cloud_mask
import rasterio
import rasterio as rio
from rasterio import merge
from rasterio.plot import show
from rasterio.mask import mask
from skimage import io,exposure, transform,data
from skimage.color import rgb2hsv, rgb2gray
from skimage.feature import local_binary_pattern
from sklearn.cluster import KMeans
import matplotlib.cm as cm
from sklearn import preprocessing
from rasterio.enums import Resampling
import seaborn as sns
import itertools

wdir= os.getcwd()
```

The remainder of this paper is structured as follows. The next section introduces the Landsat satellite imagery, study area Shanghai, and process on how to batch download and pre-process satellite data. Section 3 proposes our methods to extract different features including colour, texture, vegetation and built-up from imagery. Section 4 performs a clustering method on the extracted features, and section 5 interprets the results and gain insights from them. Finally, section 6 concludes by providing a summary of our work and avenues for further research using our proposed framework.

2 Data and Study Area

2.1 Landsat Imagery

We draw data from the NASA's Landsat satellite programme. It is the longest standing programme for Earth observation (EO) imagery (NASA 2019). Landsat satellites have been orbiting the Earth for 46 years providing increasingly higher resolution imagery. Landsat Missions 1-3 offer coarse imagery of 80m covering the period from 1972 to 1983. Landsat Missions 4-5 provides images of 30m resolution covering the period from 1983 to 2013 and Landsat Missions 7-8 are currently collecting enhanced images at 15m capturing Cirrus and Panchromatic bands, in addition to the traditional RGB, Near-, Shortwave-Infrared, and Thermal bands. The Landsat 6 mission was unsuccessful due to the transporting rocket not reaching orbit. Landsat imagery is openly available and offers extensive temporal coverage stretching for 46 years. Table 1 provides a summary overview of the operation, revisit time and image resolution for the Landsat programme, with other Earth observation satellite missions being shown in Table 2.

Additional Earth observation programmes exist. These programmes also offer freely accessible imagery at a higher resolution.

2.2 Study Area

In this analysis, we examine urban changes in Shanghai, China. Shanghai has experienced rapid population growth. Between 2000 and 2010, Shanghai's population rose by 7.4

Table 1: Overview of Landsat missions, their revisit time and spatial resolution

Mission	Operational time	Revisit time	Resolution
Landsat 1	1972-1978	18 d	80 m
Landsat 2	1975-1982	18 d	80 m
Landsat 3	1978-1983	18 d	80 m
Landsat 4	1983-1993	16 d	30 m
Landsat 5	1984-2013	16 d	30 m
Landsat 7	1999-present	16 d	15 m
Landsat 8	2013-present	16 d	15 m

Table 2: Overview of other Earth observation satellites, their revisit time and spatial resolution

Provider	Programme	Operational time	Revisit time	Resolution
European Space Agency	Sentinel	2015-present	5 d	10m
Planet Labs	Rapideye PlanetscopeSkysat	2009-present	4/5 d to daily	up to 0.8 m
NASA	Orbview 3	2003-2007	<3 d	1-4 m
NASA	EO-1	2003 -2017	–	10-30 m

million from 16.4 million to 23.8 million. It has an annual growth rate of 3.8 percent over 10 years. While the pace of population expansion has been less acute, Shanghai's population has continued to grow. In 2018, an estimated 24.24 million people were living in Shanghai experiencing a population expansion of approximately 8 million since 2010. The city is therefore a well suited example to explore long-term changes in urbanisation.

To extract satellite imagery, a first step is to identify the shape of the geographical area of interest. To this end, we use a polygon shapefile (https://www.arcgis.com/home/item.-html?id=105f92bd1fe54d428bea35eade65691b). These polygons represent the Shanghai metropolitan area, so they include the city centre and surrounding areas. These polygons will be used as a bounding box to identify and extract relevant satellite images. We need to ensure the shapefile is in the same coordinate reference system (CRS) as the satellite imagery (WGS84 or EPSG:4326).

[2]:
```
# Specify the path to your shapefile
directory = os.path.dirname(wdir)
shp = 'shang_dis_merged/shang_dis_merged.shp'
```

[3]:
```
# Certify that the shapefile is in the right coordinate system, otherwise reproject
# it into the right CRS
def shapefile_crs_check(file):
    global bbox
    bbox = gpd.read_file(file)
    crs = bbox.crs
    data = crs.get("init", "")
    if 'epsg:4326' in data:
        print('Shapefile in right CRS')
    else:
        bbox = bbox.to_crs({'init':'epsg:4326'})
    f,ax = plt.subplots(figsize=(5,5))
    plt.title('Fig.1: Shapefile of Shanghai urban area',y= -0.2)
    bbox.plot(ax=ax)
```

[4]:
```
shapefile_crs_check(shp)
```

[4]:
```
Shapefile in right CRS
image/png<Figure size 360x360 with 1 Axes>
```

The world reference system (WRS) from NASA is a system to identify individual satellite imagery scenes using path-row tuples instead of absolute latitude/longitude

Figure 1: Shapefile of Shanghai urban area

coordinates. The latitudinal centre of the image corresponds to the row, the longitudinal centre to the path. This system allows to uniformly catalogue satellite data across multiple missions and provides an easy to use reference system for the end user. It is necessary to note that the WRS was changed between Landsat missions, due to a difference in swath patterns of the more recent Landsat satellites (NASA 2019). The WRS1 is used for Landsat missions 1-3 and the WRS2 for Landsat missions 4,5,7,8. In order to obtain path-row tuples of relevant satellite images for an area of interest (AOI), it is necessary to intersect the WRS shapefile (either WRS1 or WRS2, depending on the Landsat satellite you would like to obtain data from) with the AOI shapefile. The resulting path-row tuples will later be used to locate and download the corresponding satellite images from the Google Cloud Storage. The output of the intersection between WRS and AOI files can be visualised using an interactive widget. The map below shows our area of interest in purple and the footprints of the relevant Landsat images on top of an OpenStreetMap basemap.

[5]:
```
# Donwload the WRS 2 file to later intersect the shapefile with the WRS path/row
# tuples to identify relevant Landsat scenes
#
def sat_path():

    url = 'https://prd-wret.s3.us-west-2.amazonaws.com/assets/palladium/production/
    ...s3fs-public/atoms/files/WRS2_descending_0.zip'
    # Create folder for WRS2 file
    if os.path.exists(os.path.join('Landsat_images','wrs2')):
        print('folder exists')
    else:
        os.makedirs(os.path.join('Landsat_images','wrs2'))

    WRS_PATH = os.path.join('Landsat_images','WRS2_descending_0.zip')
    LANDSAT_PATH = os.path.dirname(WRS_PATH)

    # The WRS file is only needed once thus we add this loop
    if os.path.exists(WRS_PATH):
        print('File already exists')
    # Downloads the WRS file from the URL given and unzips it
    else:
        wget.download(url, out = LANDSAT_PATH)
        shutil.unpack_archive(WRS_PATH, os.path.join(LANDSAT_PATH, 'wrs2'))
```

[6]:
```
%%time
# WARNING: this will take time the first time it's executed
# depending on your connection
sat_path()
```

[6]:
```
folder exists
File already exists
Wall time: 1e+03 mu s
```

[7]:
```
# Intersect the shapefile with the WRS2 shapefile to determine relevant path/row tuples
def get_pathrow():
    global paths,rows,path,row, wrs_intersection

    wrs=gpd.GeoDataFrame.from_file(os.path.join('Landsat_images','wrs2',
            'WRS2_descending.shp'))
    wrs_intersection=wrs[wrs.intersects(bbox.geometry[0])]
    paths,rows=wrs_intersection['PATH'].values, wrs_intersection['ROW'].values

    for i, (path,row) in enumerate(zip(paths,rows)):
        print('Image', i+1, ' -path:', path, 'row:', row)
```

[8]:
```
get_pathrow()
```

[8]:
```
Image 1  -path: 118 row: 38
Image 2  -path: 119 row: 38
```

[9]:
```
# Visualise the output of the intersection with the shapefile using Folium

# Get the center of the map
xy = np.asarray(bbox.centroid[0].xy).squeeze()
center = list(xy[::-1])

# Select a zoom
zoom = 8

# Create the most basic OSM folium map
m = folium.Map(location = center, zoom_start = zoom, control_scale=True)

# Add the bounding box (bbox) GeoDataFrame in red using a lambda function
m.add_child(folium.GeoJson(bbox.__geo_interface__, name = 'Area of Interest',
                      style_function = lambda x: {'color': 'purple', 'alpha': 0}))

loc = 'Fig 2.: Landsat satellite tiles that  cover the Area of Interest'
title_html = '''
         <figcaption align="center" style="font-size:12px"><b>{}</b></figcaption>
         '''.format(loc)
m.get_root().html.add_child(folium.Element(title_html))

# Iterate through each polygon of paths and rows intersecting the area
for i, row in wrs_intersection.iterrows():
    # Create a string for the name containing the path and row of this Polygon
    name = 'path: %03d, row: %03d' % (row.PATH, row.ROW)
    # Create the folium geometry of this Polygon
    g = folium.GeoJson(row.geometry.__geo_interface__, name=name)
    # Add a folium Popup object with the name string
    g.add_child(folium.Popup(name))
    # Add the object to the map
    g.add_to(m)
m
```

[9]:
```
text/html<folium.folium.Map at 0x1f0ea0d7dd8>
```

[10]:
```
+fvtextcolorcomment_color# Display number of images and Path/Row of the image
for i, (path,row) in enumerate(zip(paths,rows)):
    print('Image', i+1, ' -path:', path, 'row:', row)
```

[10]:
```
Image 1  -path: 118 row: 38
Image 2  -path: 119 row: 38
```

Note that here you have two options: 1) continuing and executing the code reported in the next two sections on data donwload and image cropping, or 2) skipping these sections and proceeding to the image mosaicing sections. We recommend 2) as the processing of unzipping every folder may take long causing the JupyterLab instance to crash.

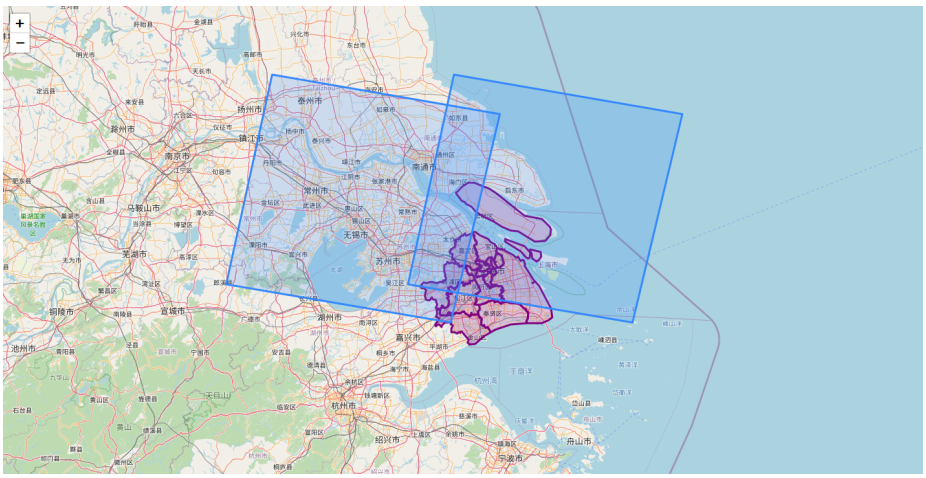


Figure 2: Landsat satellite tiles that cover the Area of Interest

2.3 Data download and pre-processing

We now have relevant path and row tuples for our area of analysis. So we can proceed to download satellite images, which are stored on the Google Cloud. To download images, we specify certain parameters: time frame, cloudcover in percentage (0-100 %) and satellite mission (1-5,7,8). The here used Landsat578 API automatically searches the Google Cloud for scenes with the specified parameters and downloads matching images. In order to search the Google Cloud for relevant images, a list of available needs to be downloaded when the code is run for the first time. The list provides basic information of the satellite images and since Landsat data acquisition is ongoing, is updated continuously. Thus, if data from the latest acquisition date is required, it is recommended to re-download the file list before running the code.

We use satellite imagery from a Landsat 5 scene taken in 1984 and a Landsat 8 taken in 2019 to determine neighbourhood changes over time. Landsat 5 scenes can be obtained from two different sensors, the Multispectral Scanner System and the Thematic Mapper, which provide 4 and 7 bands, respectively. The Multispectral Scanner System (MSS) is used in Landsat 1-3 and was superseded by the Thematic Mapper (TM). The MSS provides a green and red band (Band numbers: 1,2) and two infrared bands (Band numbers: 3,4), while the TM provides bands covering red, blue and green (Band numbers: 1,2,3), near-infrared (Band numbers: 4), short-wave infrared (Band numbers: 5,7) and thermal infrared (6). Each downloaded scene contains all bands with one image per band. The different bands can then be stacked in order to highlight various Earth surface processes. In this exercise, scenes from the MSS and TM are downloaded, but only data from the TM is used for analysis.

The Operational Land Imager (OLI) aboard Landsat 8 provides multispectral bands (bands 1-7 and 9) with a resolution of 30 metres and a panchromatic band (band 8) with a resolution of 15 metres (Barsi et al. 2014a). The Thermal Infrared Sensor (TIRS) provides thermal infrared images (bands 10 and 11) with a resolution of 100 meters (Barsi et al. 2014b). The Landsat 8 satellite has a swath width of 185 km for the OLI and TIRS instruments, so one scene usually captures the extent of a city. In other cases, the geographical area of interest may extend beyond one image so that multiple images may be needed (Barsi et al. 2014b, Knight, Kvaran 2014). Given the revisit time of 16 days, usually cloud free images can be retrieved for most cities on a bi-weekly or monthly basis (Roy et al. 2014). The folder and filename of each scene provides information about the satellite, instrument, path/row tuple and date.

Table 3 and Table 4 show which general information of the downloaded scenes can be inferred from the folder and file names of each individual scene:

Table 3: Overview of folder naming convention for Landsat images

Parameter	Meaning
L	Landsat
X	Sensor (“C”=OLI/TIRS combined, “O”=OLI-only, “T”=TIRS-only, “E”=ETM+, “T”=TM, “M”=MSS)
PPP	WRS path
RRR	WRS row
YYYY	Year
DDD	Julian day of year
GSI	Ground station identifier
VV	Archive version number

Note: Folder names are structured as `LXPPPRRRYYYYDDDGSIVV`

Table 4: Overview of file naming convention for Landsat images

Parameter	Meaning
L	Landsat
X	Sensor (“C”=OLI/TIRS combined, “O”=OLI-only, “T”=TIRS-only, “E”=ETM+, “T”=TM, “M”=MSS)
SS	Satellite (“0””=Landsat 7, “08”=Landsat 8)
LLL	Processing correction level (L1TP/L1GT/L1GS)
PPP	WRS path
RRR	WRS row
YYYYMMDD	Acquisition year, month, day
yyyymmdd	Processing year, month, day
CC	Collection number (01, 02, …)
TX	Collection category (“RT”=Real-Time, “T1”=Tier 1, “T2”=Tier 2)

Note: File names are structured as `LXSS_LLLL_PPPRRR_YYYYMMDD_yyyymmdd_CC_TX`

2.3.1 Landsat imagery download

We will now download two Landsat satellite images, one from 1984 and one from 2019. The starting year was chosen due to the increase in spatial resolution to 30 metres with Landsat 4, whereas the end year was chosen at random. The specific dates were selected as the cloud cover was below 5%, ensuring an unobstructed view of the urban area.

[11]:
```
# Download Tile list from Google - only needs to be done when first running the code
# NOTE this cell is using the ! magic, which runs command line processes from a Jupyter
# notebook. Make sure the ‘landsat‘ tool, from the ‘landsat578‘ package is installed
# and available

# Path to index file
Index_PATH = os.path.join(directory  ’/index.csv.gz’)
if os.path.exists(Index_PATH):
    print(’File already exists’)
else:
    !landsat --update-scenes yes
```

[12]:
```
# Define Download function to acquire scenes from the Google API
def landsat_download(start_date, end_date, sat,path,row,cloud,output):
    g=GoogleDownload(start=start_date, end=end_date, satellite=sat, path=path,
    ...row=row, max_cloud_percent=cloud, output_path=output)
    g.download()
```

[13]:
```
# Specify start/end date (in YYYY-MM-DD format), the cloud coverage of the image (in %)
# and the satellite you would like to acquire images from (1-5,7,8). In this case we
# acquire a recent scene from Landsat 8 with a cloud coverage of 5 %.

start_date = ’2019-01-01’
end_date = ’2019-02-20’
```

```
cloud = 5
satellites = [8]
output = os.path.join(directory  '/Lansat_images/')
```

[14]:
```
# Loop through the specified satellites for each path and row tuple
for sat in satellites:
    for i, (path,row) in enumerate(zip(paths,rows)):
        print('Image', i+1, ' -path:', path, 'row:', row)
        landsat_download(start_date, end_date,sat,path,row,cloud,output)
```

[15]:
```
# The above step is repeated to acquire a Landsat 5 scene from 1984 with 5 % cloud
# coverage.
start_date = '1984-04-22'
end_date = '1984-04-24'
cloud = 5
satellites = [5]
output = os.path.join(directory  '/Lansat_images/')
```

[16]:
```
# Loop through the specified satellites for each path and row tuple
for sat in satellites:
    for i, (path,row) in enumerate(zip(paths,rows)):
        print('Image', i+1, ' -path:', path, 'row:', row)
        landsat_download(start_date, end_date,sat,path,row,cloud,output)
```

[17]:
```
# Delete Scenes that were acquired using the MSS:
outdir = os.listdir(output)
for i in outdir:
    if 'LM' in os.path.basename(i):
        try:
            shutil.rmtree(os.path.abspath(os.path.join(output,os.path.basename(i))))
        except OSError as e:
            print ("Error: %s - %s." % (e.filename, e.strerror))
```

2.3.2 Image Cropping

Satellite imagery is large. The size per image can easily equate to 1 GB. It often makes the data processing and analysis computationally expensive. Cropping the obtained scenes to the relevant region of the image enables faster processing and analysing by significantly reducing the size of the input.

[18]:
```
# Define cropping function using command line gdalwarp.
## Note: The BQA band is the quality assessment band, which has a different no data
## value (1) than the other bands (0), which makes it necessary to us a different
## croping function.
def crop(inraster,outraster,shape):
        !gdalwarp -cutline {shape} -srcnodata 0 -crop_to_cutline {inraster} {outraster}
def crop_bqa(inraster,outraster,shape):
        !gdalwarp -cutline {shape} -srcnodata 1 -crop_to_cutline {inraster} {outraster}
```

[19]:
```
# Loop through every folder and a create an image cropped to the extent of the shapefile
# save it with the original name and the extension _Cropped
for t in range(0,12):
    for filename in glob.glob((output/**/*_B{}.tif').format(t), recursive=True):
        inraster = filename
        outraster = filename[:-4]  '_Cropped.tif'
        crop(inraster, outraster, shp)
for filename in glob.glob(output/**/*.tif'):
    if 'BQA.TIF' in i:
        inraster = i
        outraster = i[:-4]  '_Cropped.tif'
        crop_bqa(inraster,outraster,shp)
```

2.3.3 Image mosaic

As indicated above, a single Landsat scene may not cover the full extent of a city due to the satellite's flight path as can be observed from the interactive map. Creating a mosaic of two or more images is thus often needed to produce a single image that covers the entirety of the area under analysis.

[20]:
```
# Read in the relevant Landsat 8 files
output = 'Landsat_images/'
images = sorted(os.listdir(output))
dirpath1 = os.path.join(output, images[0])
dirpath2 = os.path.join(output, images[1])
mosaic_n = os.path.join(output,'Mosaic/')
search = 'L*_Cropped.tif'
query1 = os.path.join(dirpath1,search)
query2 = os.path.join(dirpath2,search)
files1 =  glob.glob(query1)
files2 =  glob.glob(query2)
files1.sort()
files2.sort()
if os.path.exists(mosaic_n):
    print('Output Folder exists')
else:
    os.makedirs(mosaic_n)
```

[21]:
```
# Match bands together and create a mosaic. Since the BQA band and the cloudmask have
# different denominations than the other bands, these images have to be merged
# together separately.
def mosaic_new(scene1,scene2):
    src_mosaic =[]
    string_list=[]
    for i,j in zip(scene1,scene2):
        for k in range(1,12):
            string_list.append('B{}_Cropped'.format(k))
        for l in range(0,11):
            if string_list[l] in os.path.basename(i) and os.path.basename(j):
                src1 = rasterio.open(i)
                src2 = rasterio.open(j)
                src_mosaic = [src1,src2]
                mosaic,out_trans = rasterio.merge.merge(src_mosaic)
                out_meta = src1.meta.copy()
                out_meta.update({"driver": "GTiff", 'height':mosaic.shape[1],
                                 'width':mosaic.shape[2],'transform':out_trans})
                outdata = os.path.join(mosaic_n,'B{}_mosaic.tif'.format(l))
                with rasterio.open(outdata,'w',**out_meta) as dest:
                    dest.write(mosaic)
        # Mosaic Quality Assessment Band
        if 'BQA_Cropped' in os.path.basename(i) and os.path.basename(j):
            bqa1 = rasterio.open(i)
            bqa2 = rasterio.open(j)
            bqa_mosaic = [bqa1,bqa2]
            mosaic_,out_trans = rasterio.merge.merge(bqa_mosaic,nodata=1)
            out_meta = bqa1.meta.copy()
            out_meta.update({"driver": "GTiff", 'height':mosaic_.shape[1],
                             'width':mosaic_.shape[2],'transform':out_trans})
            outdata = os.path.join(mosaic_n,'BQA_mosaic.tif')
            with rasterio.open(outdata,'w',**out_meta) as dest:
                dest.write(mosaic_)
            # Mosaic of Cloudmask
            search = 'cloudmask.tif'
            query3 = os.path.join(dirpath1,search)
            query4 = os.path.join(dirpath2,search)
            files3 =  glob.glob(query3)
            files4 =  glob.glob(query4)
            for i,j in zip(files3,files4):
                if 'cloudmask' in os.path.basename(i)and os.path.basename(j):
                    cloudmask1 = rasterio.open(i)
                    cloudmask2 = rasterio.open(j)
                    cloud_mosaic = [cloudmask1,cloudmask2]
                    mosaic_c,out_trans = rasterio.merge.merge(cloud_mosaic,nodata=1)
                    out_meta = cloudmask1.meta.copy()
                    out_meta.update({"driver": "GTiff", 'height':mosaic_c.shape[1],
                                     'width':mosaic_c.shape[2],'transform':out_trans})
                    outdata = os.path.join(mosaic_n,'Cloudmask_mosaic.tif')
                    with rasterio.open(outdata,'w',**out_meta) as dest:
                        dest.write(mosaic_c)
```

[22]:
```
mosaic_new(files1,files2)
```

[23]:
```
# Read in the relevant files for the Landsat 5 scenes
images = sorted(os.listdir(output))
dirpath_o1 = os.path.join(output, images[2])
dirpath_o2 = os.path.join(output, images[3])
mosaic_o = os.path.join(output,'Mosaic_old/')
query_o1 = os.path.join(dirpath_o1,search)
query_o2 = os.path.join(dirpath_o2,search)
files_o1 =  glob.glob(query_o1)
files_o2 =  glob.glob(query_o2)
files_o1.sort()
files_o2.sort()
if os.path.exists(mosaic_o):
    print('Output Folder exists')
else:
    os.makedirs(mosaic_o)
```

[24]:
```
# Match bands together and create a mosaic. Since the BQA band and the cloudmask have
# different denominations than the other bands, these images have to be merged together
# separately.
def mosaic_old(scene_o1,scene_o2):
    src_mosaic =[]
    string_list=[]
    for i,j in zip (scene_o1,scene_o2):

        for k in range(1,8):
            string_list.append('B{}_Cropped'.format(k))
        for l in range(0,7):
            if string_list[l] in os.path.basename(i) and os.path.basename(j):
                src1 = rasterio.open(i)
                src2 = rasterio.open(j)
                src_mosaic = [src1,src2]
                mosaic,out_trans= rasterio.merge.merge(src_mosaic)
                out_meta = src1.meta.copy()
                out_meta.update({"driver": "GTiff", 'height':mosaic.shape[1],
                                 'width':mosaic.shape[2],'transform':out_trans})
                outdata = os.path.join(mosaic_o,'B{}_mosaic.tif'.format(l))
                with rasterio.open(outdata,'w',**out_meta) as dest:
                    dest.write(mosaic)
        # Mosaic Quality Assessment Band
        if 'BQA_Cropped' in os.path.basename(i) and os.path.basename(j):
            bqa1 = rasterio.open(i)
            bqa2 = rasterio.open(j)
            bqa_mosaic = [bqa1,bqa2]
            mosaic_,out_trans= rasterio.merge.merge(bqa_mosaic,nodata=1)
            out_meta = bqa1.meta.copy()
            out_meta.update({"driver": "GTiff", 'height':mosaic_.shape[1],
                             'width':mosaic_.shape[2],'transform':out_trans})
            outdata = os.path.join(mosaic_o,'BQA_mosaic.tif')
            with rasterio.open(outdata,'w',**out_meta) as dest:
                dest.write(mosaic_)

            # Mosaic of Cloudmask
            search = 'cloudmask.tif'
            query_o3= os.path.join(dirpath_o1,search)
            query_o4 = os.path.join(dirpath_o2,search)
            files_o3 =  glob.glob(query_o3)
            files_o4 =  glob.glob(query_o4)
            for i,j in zip(files_o3,files_o4):
                if 'cloudmask' in os.path.basename(i)and os.path.basename(j):
                    cloudmask1 = rasterio.open(i)
                    cloudmask2 = rasterio.open(j)
                    cloud_mosaic = [cloudmask1,cloudmask2]
                    mosaic_c,out_trans= rasterio.merge.merge(cloud_mosaic,nodata=1)
                    out_meta = cloudmask1.meta.copy()
                    out_meta.update({"driver": "GTiff", 'height':mosaic_c.shape[1],
                                     'width':mosaic_c.shape[2],'transform':out_trans})
                    outdata = os.path.join(mosaic_o,'Cloudmask_mosaic.tif')
                    with rasterio.open(outdata,'w',**out_meta) as dest:
                        dest.write(mosaic_c)
```

[25]:
```
mosaic_old(files_o1,files_o2)
```

2.3.4 Natural-colour (True-colour) composition

Our downloaded data from Landsat 8 and Landsat 5 have different band designations. Combining different satellite bands are useful to identify features of the urban environment: vegetation, built-up areas, ice and water. We create a standard natural-colour composition image using Red, Green and Blue satellite bands. This colour composition best reflects the natural environment. For instance, trees are green; snow and clouds are white; and water is blue. Landsat 8 has 11 bands with bands 4, 3 and 2 corresponding to Red, Green and Blue respectively. Landsat 5 has 7 bands with bands 3, 2 and 1, corresponding to Red, Green and Blue. We perform layer stacking to produce a true colour image composition to gain understanding of the local area before extracting and analysing features of the urban environment.

[26]:
```
# Normalise the bands to so that they can be combined to a single image
def normalize(array):
    """Normalizes numpy arrays into scale 0.0 - 1.0"""
    array_min, array_max = array.min(), array.max()
    return ((array - array_min)/(array_max - array_min))
```

[27]:
```
# Adjust the intensity of each band for visualisation.
# This is a way of rescaling each band by clipping the pixels that are outside the
# specified range to the range we defined. By adjusting the gamma, we change the
# brightness of the image with gamma >1 resulting in a brighter image. However
# there are more complex methods such as top of the atmosphere corrections, which
# subtracts any atmospheric interference from the image.
# For the purpose of this notebook, this way is sufficient.
def rescale_intensity(image):
    p2, p98 = np.percentile(image, (0.2, 98))
    img_exp = exposure.rescale_intensity(image, in_range=(p2, p98))
    img_gamma = exposure.adjust_gamma(img_exp, gamma=2.5,gain=1)
    return(img_gamma)
```

[28]:
```
# Downsample image resolution with factor 0.5 for displaying purposes.
def downsample(file):
    downscale_factor=0.5
    data = file.read(1,
        out_shape=(
            file.count,
            int(file.height * downscale_factor),
            int(file.width * downscale_factor)
        ),
        resampling=Resampling.bilinear
    )
    # scale image transform
    transform = file.transform * file.transform.scale(
        (file.width / data.shape[-1]),
        (file.height / data.shape[-2])
    )
    return data
```

[29]:
```
# Use rasterio to open the Red, Blue and Green bands of the mosaic image from 1984
# to create an RGB image
# **NOTE**: The Mosaic names do not correspond to the actual band designations as
# python starts counting at 0!
with rasterio.open('Landsat_images/Mosaic_old/B0_mosaic.tif') as band1_old:
    b1_old=downsample(band1_old)
with rasterio.open('Landsat_images/Mosaic_old/B1_mosaic.tif') as band2_old:
    b2_old=downsample(band2_old)
with rasterio.open('Landsat_images/Mosaic_old/B2_mosaic.tif') as band3_old:
    b3_old=downsample(band3_old)
```

[30]:
```
# Normalise the bands so that they can be combined to a single image
red_old_n = normalize(b3_old)
green_old_n = normalize(b2_old)
blue_old_n = normalize(b1_old)

# Apply the function defined before to make more natural-looking image
red_adj = rescale_intensity(red_old_n)
green_adj = rescale_intensity(green_old_n)
blue_adj = rescale_intensity(blue_old_n)
```

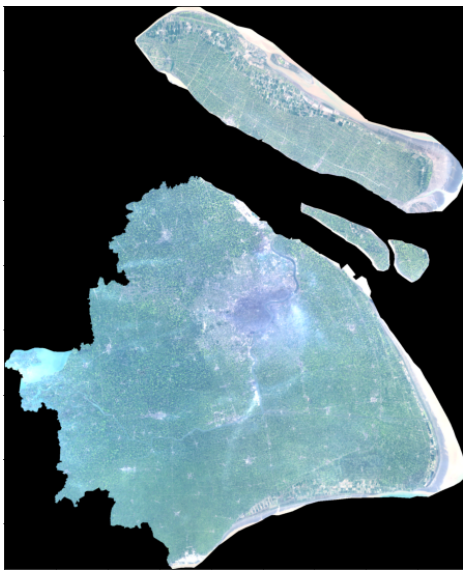


Figure 3: True colour Landsat image of the Shanghai urban area from 1984

```
# Stack the three different bands together
rgb_2 = np.dstack((red_adj,green_adj,blue_adj))

# Visualise the true color image
fig,ax = plt.subplots(figsize=(10,10))
ax.imshow(rgb_2)
plt.title('Fig.3: True color Landsat image of the Shanghai urban area from 1984',
          y=-0.1, fontsize=12)
plt.show()
plt.close()
del rgb_2,b1_old,b2_old,b3_old,red_adj,green_adj,blue_adj
```

[30]:
```
image/png<Figure size 720x720 with 1 Axes>
```

[31]:
```
# Use rasterio to open the Red, Blue and Green bands of the mosaic image from 2019
# to create an RGB image
# **NOTE**: The Mosaic names do not correspond to the actual band designations as
# python starts counting at 0!!
with rasterio.open('Landsat_images/Mosaic/B1_mosaic.tif') as band2_new:
    b2_new = downsample(band2_new)
with rasterio.open('Landsat_images/Mosaic/B2_mosaic.tif') as band3_new:
    b3_new = downsample(band3_new)
with rasterio.open('Landsat_images/Mosaic/B3_mosaic.tif') as band4_new:
    b4_new = downsample(band4_new)
```

[32]:
```
# Normalise the bands so that they can be combined to a single image
red_new_n = normalize(b4_new)
green_new_n = normalize(b3_new)
blue_new_n = normalize(b2_new)

# Apply the function defined before to make more natural-looking image
red_rescale = rescale_intensity(red_new_n)
green_rescale = rescale_intensity(green_new_n)
blue_rescale = rescale_intensity(blue_new_n)

# Stack the three different bands together
rgb = np.dstack((red_rescale, green_rescale, blue_rescale))

# Here we adjust the gamma (brightness) for the stacked image to achieve a more
# natural looking image.
rgb_adjust = exposure.adjust_gamma(rgb, gamma = 1.5, gain=1)

# Visualise the true color image
fig,ax = plt.subplots(figsize=(10,10))
```

Figure 4: True colour Landsat image of the Shanghai urban area from 2019

```
ax.imshow(rgb_adjust)
plt.title('Fig.4: True color Landsat image of the Shanghai urban area from 2019',
          y=-0.1, fontsize=12)
plt.show()
plt.close()
del rgb,red_new_n,green_new_n,blue_new_n,red_rescale,green_rescale,blue_rescale,
         rgb_adjust
```

[32]:
```
image/png<Figure size 720x720 with 1 Axes>
```

When comparing the true colour Landsat satellite images in Figures 3 and 4, the urbanisation of Shanghai between 1984 and 2019 is apparent. In the following steps, we will analyse and quantify these urban changes.

3 Feature extraction

Since the above two maps show that urban neighbourhoods of Shanghai have undergone dramatic changes over time in colour, texture, greenery, buildings, etc., the next stage is to gain valuable information out of satellite images and interpret these changes. Since the images we have downloaded are on a city-wide scale, which covers more than a thousand kilometre spatial resolution and less detailed. Therefore, feature extraction is performed to get a reduced representation of the initial image but informative and sufficiently accurate for subsequent analysis and interpretation.

We examine four sets of features based on the above two true colour maps and the scale, where the colour, texture, greenery, and buildings changed a lot during the past 25 years in Shanghai. Specifically, colour and texture features extracted from true colour imagery (i.e. RGB bands composition represented by bands 1-3 and bands 2-4 in 1984 and 2019), and vegetation features and built-up features extracted from Red, near infrared (NIR) and shortwave infrared (SWIR) bands, represented by bands 3-5 and bands 4-6 in 1984 and 2019. More detailed information about the meaning of each band can be found at https://www.usgs.gov/faqs/what-are-best-landsat-spectral-bands-use-my-research?qt-news_science_products=0#qt-news_science_products. In this analysis, colour features measure the colour moments of true colour imagery to interpret colour distribution; texture features apply LBP (Local binary patterns) texture spectrum model to show spatial distribution of intensity values in an image; vegetation features calculate the NDVI (Normalised difference vegetation index) to capture the amount of vegetation, and built-up features calculate NDBI (Normalised difference built-up index) to highlight artificially constructed areas.

Figure 5: Spatial distribution of all administrative divisions of Shanghai

The administrative divisions of Shanghai have experienced tremendous changes in the last tens of years (Ministry of Civial Affairs of the People's Republic of China 2018), thus, we will conduct feature extraction of imagery on the current administrative boundaries to explore if satellite imagery can be used to reflect and interpret urban changes. The figure below shows the spatial distribution of each administrative area with relative labels in Shanghai.

[33]:
```
# read administritive boundary shapefile of Shanghai
poly = gpd.read_file(shp)


f, ax = plt.subplots(1, figsize = (9,9))
poly.plot(ax = ax)
# create a new colomn, in order to plot polygon labels (i.e. name) in the map
poly['coords']=poly['geometry'].apply(lambda x:x.representative_point().coords[:])
poly['coords']=[coords[0] for coords in poly['coords']]
for idx, row in poly.iterrows():
    ax.annotate(text=row['Name'],xy=row['coords'],va='center',ha='center',alpha = 0.8,
                fontsize = 8)
plt.axis('equal')
plt.axis('off')
f.suptitle('Fig.5: Spatial distribution of all administrative divisions of Shanghai',
                y=-0.1,fontsize = 12)
```

[33]:
```
Text(0.5, -0.1, 'Fig.5: Spatial distribution of all administrative divisions of
Shanghai')
image/png<Figure size 648x648 with 1 Axes>
```

Figure 5 shows that administrative divisions of 'Chongming' in the north appear three geometries. Therefore, it is necessary to check if they belong to a single administrative unit.

[34]:
```
poly.loc[poly['Name']== 'Chongming','Name']
```

[34]:
```
0    Chongming
3    Chongming
5    Chongming
Name: Name, dtype: object
```

Chongming administrative division consist of three separate geometries, which may confuse our further analysis. As a result, we dissolved these geometries into a single geometric feature and take a look at the new dataset. The below table shows that the Chongming administrative division now consists of multipolygons which includes all polygons as a whole.

[35]:
```
# Dissolve geometries with the identical names together
poly = poly.dissolve(by = 'Name').reset_index()
# Have a look at the name of all administritive unit and we can see that chongming
# districts have been dissolved into a single administritive unit
poly['Name'].values
```

[35]:
```
array(['Baoshan', 'Changning', 'Chongming', 'Fengxian', 'Hongkou',
       'Huangpu', 'Jiading', 'Jinshan', 'Minhang', 'Pudong New', 'Putuo',
       'Qingpu', 'Songjiang', 'Xuhui', 'Yangpu', 'Zhabei'], dtype=object)
```

3.1 Image processing

Further pre-processing of satellite imagery is needed before feature extraction. This pre-processing involves three steps:

1. Masking (cropping) of raster files (i.e., Blue, Green, Red, Nir and SWIR bands) into each administrative district polygon;
2. Image enhancement to improve the quality and content of the original image; and,
3. Band stacking based on each neighbourhood unit.

[36]:
```
# open raster files
file_list_old = sorted(glob.glob('Landsat_images/Mosaic_old'  "/*.tif",recursive = True))
files_old = [rio.open(filename) for filename in file_list_old]
```

[37]:
```
file_list = sorted(glob.glob('Landsat_images/Mosaic'  "/*.tif"))
files = [rio.open(filename) for filename in file_list]
```

Before cropping all raster files into each polygon in the vector file (i.e. Shanghai administrative area shapefile), we have to ensure they have the same coordinate reference system (CRS). Once matched, the cropping process is prepared to go.

[38]:
```
poly.crs
```

[38]:
```
{'init': 'epsg:4326'}
```

[39]:
```
# check the crs of one band of satellite imagery
files[0].crs
```

[39]:
```
CRS.from_epsg(32651)
```

[40]:
```
# reproject the vector file to make it consistent with raster files
poly = poly.to_crs('EPSG:32651')
```

[41]:
```
# get each neighbourhood geographic boundary based on administritive area data
geo = [poly.__geo_interface__['features'][i]['geometry']
            for i in range(len(poly))]
```

[42]:
```
# clip R,G,B bands separately by each poly, so get pixel values in each poly and save
# them into a list
out_image = [[] for i in range(5)]
img_old = [[] for i in range(5)]

# x: Blue,Green,Red,NIR and SWIR bands, y: 16 polygons from vertor file
for x,y in itertools.product(range(5),range(len(geo))):
    # out_image[0] means masked Blue band polygon
    out_image[x].append(mask(files_old[0:5][x], [geo[y]], crop=True))
    # image enhancement: normalisation and Histogram Equalization
    img_old[x].append(exposure.equalize_hist(normalize(out_image[x][y][0][0])))
del out_image,files_old
```

[43]:
```
# clip R,G,B bands separately by each poly, so get pixel values in each poly and save
# them into a list
out_image = [[] for i in range(5)]
img_new = [[] for i in range(5)]

# x: Blue,Green,Red,NIR and SWIR bands, y: 16 polygons from vertor file
```

```
for x,y in itertools.product(range(5),range(len(geo))):
    # out_image[0] means masked Blue band polygon
    out_image[x].append(mask(files[0:5][x], [geo[y]], crop=True))
    # image enhancement: normalisation and Histogram Equalization
    img_new[x].append(exposure.equalize_hist(normalize(out_image[x][y][0][0])))
del out_image,files
```

[44]:
```
# have a look at the pixel values of one geographic area in blue band
img_new[0][0]
```

[44]:
```
array([[0.48515378, 0.48515378, 0.48515378, ..., 0.48515378, 0.48515378,
        0.48515378],
       [0.48515378, 0.48515378, 0.48515378, ..., 0.48515378, 0.48515378,
        0.48515378],
       [0.48515378, 0.48515378, 0.48515378, ..., 0.48515378, 0.48515378,
        0.48515378],
       ...,
       [0.48515378, 0.48515378, 0.48515378, ..., 0.48515378, 0.48515378,
        0.48515378],
       [0.48515378, 0.48515378, 0.48515378, ..., 0.48515378, 0.48515378,
        0.48515378],
       [0.48515378, 0.48515378, 0.48515378, ..., 0.48515378, 0.48515378,
        0.48515378]])
```

[45]:
```
# stack R,G,B bands together for later feature extraction
bb = [img_old[0][x].astype(np.float) for x in range(len(geo))]
bg = [img_old[1][x].astype(np.float) for x in range(len(geo))]
br = [img_old[2][x].astype(np.float) for x in range(len(geo))]
```

[46]:
```
rgb_old = [np.dstack((br[x],bg[x],bb[x])) for x in range(len(geo))]
```

[47]:
```
bb = [img_new[0][x].astype(np.float) for x in range(len(geo))]
bg = [img_new[1][x].astype(np.float) for x in range(len(geo))]
br = [img_new[2][x].astype(np.float) for x in range(len(geo))]
```

[48]:
```
rgb_new = [np.dstack((br[x],bg[x],bb[x])) for x in range(len(geo))]
```

3.2 Colour features

Colour features are used to extract the characteristics of colours from satellite imagery. A commonly used method to extract colour features is to compute colour moments of an image. Colour moments provide a measurement of colour similarity between images (Keen 2005). Basically, colour probability distributions of an image are characterised by a range of unique moments. The mean, standard deviation and skewness these three central moments are generally used to identify colour distribution. Here we extract colour features on HSV (Hue, Saturation and Value) colour space because it corresponds to human vision and has been widely used in computer vision. HSV colour space can be converted from RGB colour channels, Hue represents the colour portion, saturation represents the amount of grey in a particular colour (0 is grey), and Value represents the brightness of the colour (0 is black). Therefore, the true-colour imagery is characterised by a total of nine moments - three moments for each HSV channel in the same units.

[49]:
```
# interpret the color probability distribution by computing low order color
# moments(1,2,3)
def color_moments(img):
    if img is None:
        return
    # Convert RGB to HSV colour space
    img_hsv = rgb2hsv(img)
    # Split the channels - h,s,v
    h, s, v = [img_hsv[:,:,i] for i in [0,1,2]]
    # Initialize the colour feature
    color_feature = []
    # N = h.shape[0] * h.shape[1]
    # The first central moment - average
    h_mean = np.mean(h)  # np.sum(h)/float(N)
    s_mean = np.mean(s)  # np.sum(s)/float(N)
    v_mean = np.mean(v)  # np.sum(v)/float(N)
```

Table 5: Partial colour features identified in 1984

Name	h_mean	s_mean	v_mean	h_std	s_std	v_std	h_skew	s_skew	v_skew
Baoshan	0.27216	0.05208	0.64415	0.32709	0.07246	0.18531	0.35671	0.09006	0.19945
Changning	0.22141	0.05156	0.65989	0.28837	0.07518	0.17446	0.33080	0.09250	0.18763
Chongming	0.15381	0.01739	0.74231	0.27216	0.03563	0.10235	0.33292	0.05118	0.12060
Fengxian	0.33961	0.11292	0.60576	0.32194	0.12281	0.24362	0.34723	0.14439	0.25767
Hongkou	0.24951	0.06370	0.65073	0.30983	0.08744	0.18781	0.34797	0.10619	0.20013

```
    color_feature.extend([h_mean, s_mean, v_mean])
    # The second central moment - standard deviation
    h_std = np.std(h)  # np.sqrt(np.mean(abs(h - h.mean())**2))
    s_std = np.std(s)  # np.sqrt(np.mean(abs(s - s.mean())**2))
    v_std = np.std(v)  # np.sqrt(np.mean(abs(v - v.mean())**2))
    color_feature.extend([h_std, s_std, v_std])
    # The third central moment - the third root of the skewness
    h_skewness = np.mean(abs(h - h.mean())**3)
    s_skewness = np.mean(abs(s - s.mean())**3)
    v_skewness = np.mean(abs(v - v.mean())**3)
    h_thirdMoment = h_skewness**(1./3)
    s_thirdMoment = s_skewness**(1./3)
    v_thirdMoment = v_skewness**(1./3)
    color_feature.extend([h_thirdMoment, s_thirdMoment, v_thirdMoment])

    return color_feature
```

[50]:
```
# create and initialize a data table to store colour feastures
color_mom_old = pd.DataFrame(color_moments(rgb_old[0]))
# add the rest columns by assigning 9 color moments in each poly
for i in range(1,len(rgb_old)):
    color_mom_old[i]= color_moments(rgb_old[i])
    i = i+1
```

[51]:
```
# create and initialize a data table
color_mom_new = pd.DataFrame(color_moments(rgb_new[0]))
# add the rest columns by assigning 9 color moments in each poly
for i in range(1,len(rgb_new)):
    color_mom_new[i]= color_moments(rgb_new[i])
    i = i+1
```

[52]:
```
# Data manipulation
color_old_var = color_mom_old.T
# assign column names
color_old_var.columns =
   ['h_mean','s_mean','v_mean','h_std','s_std','v_std','h_skew','s_skew','v_skew']
# set geographic name as index
color_old_var= color_old_var.set_index(poly.Name)
```

[53]:
```
color_new_var = color_mom_new.T
color_new_var.columns =
   ['h_mean','s_mean','v_mean','h_std','s_std','v_std','h_skew','s_skew','v_skew']
color_new_var= color_new_var.set_index(poly.Name)
```

As we have created two new tables for colour features in the year 1984 and 2019, it would be helpful to have a view of the tables and see how they look like. Table 5 and Table 6 show nine variables (column) representing colour features within five administrative division of Shanghai (row).

[54]:
```
# check the information of colour feature
color_old_var.head().style.set_caption('Table 5: Partial colour features
... identified in 1984')
```

[54]:
```
text/html<pandas.io.formats.style.Styler at 0x1f0ed9c4be0>
```

[55]:
```
color_new_var.head().style.set_caption('Table 6: Partial colour features
... identified in 2019')
```

[55]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081f31518>
```

Table 6: Partial colour features identified in 2019

Name	h_mean	s_mean	v_mean	h_std	s_std	v_std	h_skew	s_skew	v_skew
Baoshan	0.23107	0.03587	0.63894	0.29785	0.05205	0.18019	0.33678	0.06787	0.19559
Changning	0.23185	0.03129	0.64924	0.30469	0.04847	0.16700	0.34439	0.06297	0.18248
Chongming	0.15731	0.01647	0.74240	0.28250	0.03184	0.10177	0.34529	0.04336	0.11980
Fengxian	0.29554	0.08620	0.60543	0.30273	0.09770	0.24360	0.32939	0.11570	0.25723
Hongkou	0.23994	0.03758	0.63861	0.30383	0.05505	0.18294	0.33962	0.07055	0.19762

3.3 Texture features

To extract texture features, we use a Local Binary Pattern (LBP) approach. LBP searches for pixels adjacent to a central point and tests whether these surrounding pixels are greater or less than the central pixel and generate a binary classification (Pedregosa et al. 2011) (https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_local_binary_pattern.html)). In theory, eight adjacent neighbour pixels in greyscale are set to compare with one central pixel value by 3 * 3 neighbourhood threshold, and consider the result as 1 or 0 (Ojala et al. 1996). Thus, these eight surrounding binary numbers correspond to LBP code for the central pixel value, determining the texture pattern of that threshold. Texture features are then the distribution of a collection of LBPs over an image.

[56]:
```
# convert a RGB image into Grayscale,which takes less space for analysis
gray_images_old = [rgb2gray(rgb_old[i]) for i in range(len(rgb_old))]
gray_images_new = [rgb2gray(rgb_new[i]) for i in range(len(rgb_new))]
```

[57]:
```
# settings for LBP
radius = 1 # radius = 1 refers to a 3*3 patch/window scale
n_points = 8 * radius #  the number of circularly symmetric neighbour set points
method = 'uniform' # finer quantization of the angular space which is gray scale and
                   # rotation invariant

lbps_old = [local_binary_pattern(gray_images_old[i],n_points,radius,method)
            ... for i in range(len(rgb_old))]
lbps_new = [local_binary_pattern(gray_images_new[i],n_points,radius,method)
            ... for i in range(len(rgb_new))]
```

[58]:
```
# n_bins are the same in each neighbourhood
n_bins = int(lbps_old[0].max()+1)
# define a function to count the number of points in a given bin of LBP distribution
# histogram
def count_hist(x):
    return np.histogram(lbps_old[x].ravel(),density=True, bins=n_bins,range=(0, n_bins))
# Assign counts to a new list, return the higtogram vector features in this cell(polygon)
hist_features_old = [count_hist(i)[0] for i in range(len(rgb_old))]
```

[59]:
```
# Extract texture features of another year based on same method
n_bins = int(lbps_new[0].max()+1)

def count_hist(x):
    return np.histogram(lbps_new[x].ravel(),density=True, bins=n_bins,range=(0, n_bins))

# Assign counts to a new list, return the higtogram vector features in this cell(polygon)
hist_features_new = [count_hist(i)[0] for i in range(len(rgb_new))]
```

Same with operations on colour features, this time we build two new tables (Table 7 and 8) for texture features, with each row present administrative division and each column represent texture feature.

[60]:
```
# The histogram features are the texture features
texture_old_var = pd.DataFrame([hist_features_old[a] for a in range(len(rgb_old))])
texture_old_var.columns = ['LBP'+ str(i) for i in range(n_bins)]
texture_old_var = texture_old_var.set_index(poly.Name)
# Have a look at the table with texture features of administrive division of
# Shanghai in 1984
texture_old_var.head().style.set_caption('Table 7: Partial texture features
  ... identified in 1984')
```

Table 7: Partial texture features identified in 1984

Name	LBP0	LBP1	LBP2	LBP3	LBP4	LBP5	LBP6	LBP7	LBP8	LBP9
Baoshan	0.03509	0.04196	0.04071	0.06839	0.07839	0.06748	0.04034	0.04110	0.52005	0.06648
Changning	0.03609	0.04608	0.04196	0.05979	0.06004	0.06442	0.03754	0.04339	0.53942	0.07129
Chongming	0.02582	0.02995	0.02176	0.03406	0.03958	0.03679	0.02404	0.02916	0.70944	0.04941
Fengxian	0.05551	0.06647	0.05123	0.07200	0.07377	0.07393	0.05291	0.06510	0.38211	0.10698
Hongkou	0.04202	0.05056	0.04354	0.05933	0.05676	0.07072	0.03969	0.04649	0.51043	0.08047

Table 8: Partial texture features identified in 2019

Name	LBP0	LBP1	LBP2	LBP3	LBP4	LBP5	LBP6	LBP7	LBP8	LBP9
Baoshan	0.04306	0.04774	0.04077	0.05862	0.06808	0.05681	0.03741	0.04557	0.52419	0.07777
Changning	0.04264	0.05012	0.03767	0.05137	0.05842	0.06153	0.03524	0.04708	0.53945	0.07648
Chongming	0.02547	0.02964	0.02333	0.03522	0.04762	0.03641	0.02359	0.02865	0.70442	0.04565
Fengxian	0.05121	0.06105	0.05288	0.08141	0.09716	0.07923	0.05152	0.06044	0.36993	0.09517
Hongkou	0.04703	0.05417	0.04294	0.05439	0.05501	0.06805	0.03879	0.04894	0.50732	0.08335

[60]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081ebe630>
```

[61]:
```
# The histogram features are the texture features
texture_new_var = pd.DataFrame([hist_features_new[a] for a in range(len(rgb_new))])
texture_new_var.columns = ['LBP' str(i) for i in range(n_bins)]
texture_new_var = texture_new_var.set_index(poly.Name)
# Have a look at the table with texture features of administrive division of
# Shanghai in 2019
texture_new_var.head().style.set_caption('Table 8: Partial texture features
  ... identified in 2019')
```

[61]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081ebeac8>
```

3.4 Vegetation and built-up features

Vegetation features and built-up features can be measured by calculating fundamental NDVI and NDBI indices in each administrative area respectively. The Normalized Difference Vegetation Index (NDVI) is a normalized index, using Red and NIR bands to display the amount of vegetation (NASA 2000). The use of NDVI maximizes the reflectance properties of vegetation by minimizing NIR and maximizing the reflectance in the red wavelength. The measure is used to distinguish vegetation in regions, as more vegetation will affect the ratio of visible light absorbed and near-infrared light reflected. The formula is as follows:

$$\text{NDVI} = (\text{NIR} - \text{Red})/(\text{NIR} + \text{Red})$$

The output value of this index is between -1.0 and 1.0. Close to 0 represents no vegetation, close to 1 indicates the highest possible density of green leaves, and close to -1 indicates water bodies.

The Normalized Difference Built-up Index (NDBI) uses the NIR and SWIR bands to highlight artificially constructed areas (built-up areas) where there is a typically a higher reflectance in the shortwave infrared region than the near infrared region (Zha et al. 2003). The index is a ratio type that reduces the effects of differences in terrain illumination and atmospheric effects. The formula is as follows:

$$\text{NDBI} = (\text{SWIR} - \text{NIR})/(\text{SWIR} + \text{NIR})$$

Also, the output value of this index is between -1 to 1. Higher values represent built-up areas whereas negative values represent water bodies.

After calculating these two indices, vegetation features and built-up features can be measured by calculating average values of index values within each administrative area.

Table 9: Partial vegetation and built-up features identified in 1984

Name	veg_mean	builtup_mean
Baoshan	-0.002218	0.000611
Changning	-0.002147	0.000582
Chongming	-0.000805	0.000190
Fengxian	-0.007201	0.001499
Hongkou	-0.004648	-0.000313

3.4.1 Vegetation features

[62]:
```
# identify red and NIR band to each neighbourhood unit in 1984
red_old, nir_old = img_old[2],img_old[3]
# Calculate ndvi, assign 0 to nodata pixels
ndvi_old = [np.where((nir_old[i]  red_old[i])==0, 0,
                     (nir_old[i]-red_old[i])/(nir_old[i]  red_old[i]))
            for i in range(len(poly))]
```

[63]:
```
# identify red and NIR band to each neighbourhood unit in 1984
red_new, nir_new = img_new[2],img_new[3]
# Calculate ndvi, assign 0 to nodata pixels
ndvi_new = list(map(lambda i: np.where((nir_new[i]  red_new[i])==0, 0,
                     (nir_new[i]-red_new[i])/(nir_new[i]  red_new[i])),
            list(range(len(poly)))
    ))
```

[64]:
```
veg_old_var = pd.DataFrame([np.mean(ndvi_old[i]) for i in range(len(poly))],
                       index = poly.Name, columns = ['veg_mean'])
```

[65]:
```
veg_new_var = pd.DataFrame([np.mean(ndvi_new[i]) for i in range(len(poly))],
                       index = poly.Name, columns = ['veg_mean'])
```

3.4.2 Built-up features

[66]:
```
# identify red and NIR band to each neighbourhood unit in 1984
nir_old, swir_old = img_old[3],img_old[4]
# Calculate ndvi, assign 0 to nodata pixels
ndbi_old = [np.where((nir_old[i]  swir_old[i])==0., 0,
                     (swir_old[i] - nir_old[i])/(nir_old[i]  swir_old[i]))
            for i in range(len(poly))]
```

[67]:
```
# identify red and NIR band to each neighbourhood unit in 1984
nir_new, swir_new = img_new[3],img_new[4]
# Calculate ndvi, assign 0 to nodata pixels
ndbi_new = list(map(lambda i: np.where((nir_new[i]  swir_new[i])==0., 0,
                     (swir_new[i] - nir_new[i])/(nir_new[i]  swir_new[i])),
            list(range(len(poly)))
                   ))
```

[68]:
```
builtup_old_var = pd.DataFrame([np.mean(ndbi_old[i]) for i in range(len(poly))],
                       index = poly.Name, columns = ['builtup_mean'])
```

[69]:
```
builtup_new_var = pd.DataFrame([np.mean(ndbi_new[i]) for i in range(len(poly))],
                       index = poly.Name, columns = ['builtup_mean'])
```

Table 9 and Table 10 created as shown below contain both vegetation features (NDVI) and builtup features (NDBI), with the mean value of vegetation features and built-up features (two columns) calculated at each administrative division (row).

[70]:
```
veg_built_old = pd.concat([veg_old_var,builtup_old_var], axis = 1)
veg_built_old.head().style.set_caption('Table 9: Partial vegetation and built-up
 ... features identified in 1984')
```

[70]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081e1b1d0>
```

Table 10: Partial vegetation and built-up features identified in 2019

Name	veg_mean	builtup_mean
Baoshan	-0.001801	0.001938
Changning	-0.001515	0.000774
Chongming	-0.000705	0.000318
Fengxian	-0.008185	-0.000408
Hongkou	-0.002057	-0.000277

[71]:
```
veg_built_new = pd.concat([veg_new_var,builtup_new_var], axis = 1)
veg_built_new.head().style.set_caption('Table 10: Partial vegetation and built-up
 ... features identified in 2019')
```

[71]:
```
text/html<pandas.io.formats.style.Styler at 0x1f0ed9c4828>
```

4 Feature clustering

Now we have four types of features: colour, texture, vegetation and built-up area for Shanghai in 1984 and 2019. These features are the embodiment of urban changes and vary greatly due to rapid urbanisation and development. Therefore, the subsequent task is to identify systematic patterns from these integrated features for analysis of urban changes, such as whether several administrative areas share similar patterns. A clustering method is required within this context to group these geographical divisions that are similar within each other but different between them. Considering the ease of computation and fast implementation, we use generalised and the most popular k-means clustering to identify representative types of neighbourhoods based on multiple features. K-means clustering partitions the data by creating k groups of equal variance, minimising the within-cluster sum of squares (Pedregosa et al. 2011). We can perform K-means using the package scikit-learn, which is a powerful machine learning package for Python.

[72]:
```
# merge all features together
features_old_var = pd.concat([color_old_var,texture_old_var,veg_old_var,
   builtup_old_var], axis = 1)
features_old_var.head().style.set_caption('Table 11: Four types of features
 ... (21 in total) identified in 1984')
```

[72]:
```
text/html<pandas.io.formats.style.Styler at 0x1f0ed9c4908>
```

[73]:
```
# merge all features together
features_new_var = pd.concat([color_new_var,texture_new_var,veg_new_var,
   builtup_new_var], axis=1)
features_new_var.head().style.set_caption('Table 12: Four types of features
 ... (21 in total) identified in 2019')
```

[73]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081f31438>
```

Table 11 and Table 12 reveal the integrated 21 features across our four sets of image features and their differences at geographical division in magnitude between 1984 and 2019. Since k-means clustering is one of the machine learning algorithms, which generally expect data transformation for preprocessing before fitting the algorithm. We therefore use one of the most popular rescale methods to standardise these features to lie between 0 and 1 based on MinMaxScaler() function in scikit-learn package. The motivation of this method relies on the robustness to very small standard deviation. This preprocess ensures individual features of dataset have the same scale that standard normally distributed.

[74]:
```
# Last preprocessing step before machine learning: data rescaling
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(features_old_var)
oldvar_scale = pd.DataFrame(np_scaled)
oldvar_scale.columns = features_old_var.columns
```

Table 11: Four types of features (21 in total) identified in 1984

Name	h_mean	s_mean	v_mean	h_std	s_std	v_std	h_skew
Baoshan	0.272161	0.052081	0.644148	0.327094	0.072457	0.185309	0.356713
Changning	0.221412	0.051564	0.659894	0.288368	0.075177	0.174455	0.330803
Chongming	0.153807	0.017394	0.742309	0.272162	0.035627	0.102347	0.332916
Fengxian	0.339613	0.112915	0.605758	0.321941	0.122805	0.243621	0.347226
Hongkou	0.249526	0.063704	0.650725	0.309825	0.087439	0.187805	0.347968

Name	s_skew	v_skew	LBP0	LBP1	LBP2	LBP3	LBP4
Baoshan	0.090057	0.199446	0.035093	0.041960	0.040705	0.068394	0.078389
Changning	0.092504	0.187627	0.036086	0.046078	0.041956	0.059792	0.060040
Chongming	0.051184	0.120603	0.025822	0.029946	0.021757	0.034058	0.039580
Fengxian	0.144392	0.257670	0.055508	0.066468	0.051230	0.072002	0.073767
Hongkou	0.106194	0.200131	0.042018	0.050562	0.043542	0.059326	0.056759

Name	LBP5	LBP6	LBP7	LBP8	LBP9	veg_mean	builtup_mean
Baoshan	0.067483	0.040339	0.041101	0.520053	0.066483	-0.002218	0.000611
Changning	0.064422	0.037538	0.043385	0.539416	0.071285	-0.002147	0.000582
Chongming	0.036787	0.024035	0.029158	0.709444	0.049413	-0.000805	0.000190
Fengxian	0.073928	0.052907	0.065099	0.382110	0.106981	-0.007201	0.001499
Hongkou	0.070718	0.039691	0.046490	0.510429	0.080465	-0.004648	-0.000313

[75]:
```
min_max_scaler = preprocessing.MinMaxScaler()
np_scaled = min_max_scaler.fit_transform(features_new_var)
newvar_scale = pd.DataFrame(np_scaled)
newvar_scale.columns = features_new_var.columns
```

Above two steps are the results of data transformation in 1984 and 2019. To identify robust and consistent clustering results, we merge them into a single one based on their common geographical units (see Table 13). The column names ended with '_x' and '_y' represent features extracted in 1984 and 2019, respectively. This table is the one prepared for the final k-mean clustering analysis. The dominant parameter in k-means clustering is the number of clusters (i.e., k), determining the optimal numbers of clusters is therefore a fundamental issue. We select a direct and popular elbow method as an example to assess the resulting partitions, testing nine different solutions varying k from 2 to 10. Basically, the idea of elbow method is to define clusters to minimise the total intra-cluster variation or total within-cluster sum of square (WSS). The optimal number can be determined by plotting the curve of WSS according to different k clusters and the location of a bend is considered as an indicator of the appropriate number for k.

[76]:
```
merged_var = pd.merge(oldvar_scale, newvar_scale, left_index = True, right_index = True)
merged_var.head().style.set_caption('Table 13: Integrated preprocessed features
 ... identified in 1984 and 2019 seperately')
```

[76]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081e1b6d8>
```

[77]:
```
# elbow analysis
cluster_range = range( 2, 11 )
cluster_errors = []

for num_clusters in cluster_range:
        clusters = KMeans( num_clusters )
        clusters.fit( merged_var)
        cluster_errors.append( clusters.inertia_ )
clusters_df = pd.DataFrame( { "num_clusters":cluster_range,
                              "cluster_errors": cluster_errors } )
plt.figure(figsize=(12,6))
plt.title('Fig.6: Elbow method to determine the optimal k for k-mean clustering',y=-0.2)
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[77]:
```
[<matplotlib.lines.Line2D at 0x1f0817c2550>]image/png<Figure size 864x432 with 1 Axes>
```

Table 12: Four types of features (21 in total) identified in 2019

Name	h_mean	s_mean	v_mean	h_std	s_std	v_std	h_skew
Baoshan	0.231070	0.035865	0.638941	0.297847	0.052048	0.180189	0.336779
Changning	0.231849	0.031294	0.649237	0.304689	0.048471	0.167002	0.344394
Chongming	0.157306	0.016473	0.742402	0.282495	0.031843	0.101771	0.345289
Fengxian	0.295539	0.086197	0.605431	0.302731	0.097695	0.243596	0.329388
Hongkou	0.239944	0.037582	0.638613	0.303830	0.055047	0.182938	0.339615

Name	s_skew	v_skew	LBP0	LBP1	LBP2	LBP3	LBP4
Baoshan	0.067866	0.195591	0.043059	0.047740	0.040768	0.058617	0.068075
Changning	0.062974	0.182483	0.042641	0.050118	0.037668	0.051370	0.058422
Chongming	0.043360	0.119800	0.025468	0.029637	0.023332	0.035217	0.047621
Fengxian	0.115702	0.257234	0.051206	0.061050	0.052882	0.081410	0.097157
Hongkou	0.070552	0.197624	0.047032	0.054172	0.042940	0.054392	0.055014

Name	LBP5	LBP6	LBP7	LBP8	LBP9	veg_mean	builtup_mean
Baoshan	0.056809	0.037410	0.045565	0.524189	0.077767	-0.001801	0.001938
Changning	0.061528	0.035235	0.047082	0.539452	0.076482	-0.001515	0.000774
Chongming	0.036412	0.023590	0.028650	0.704424	0.045649	-0.000705	0.000318
Fengxian	0.079233	0.051521	0.060437	0.369931	0.095174	-0.008185	-0.000408
Hongkou	0.068051	0.038789	0.048937	0.507320	0.083353	-0.002057	-0.000277

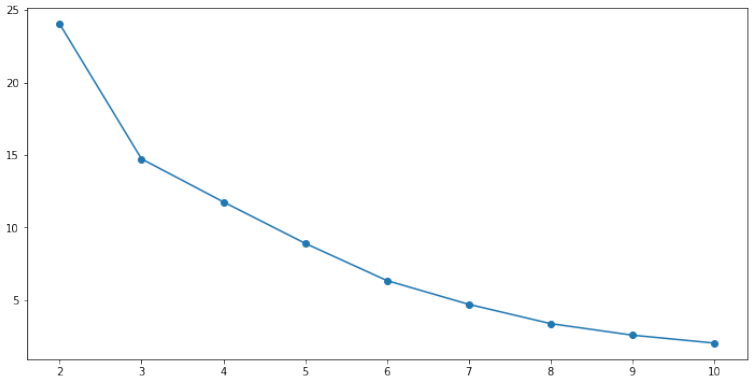


Figure 6: Elbow method to determine the optimal k for k-mean clustering

Figure 6 indicates that 2 and 6 (i.e. knee in the plot) can be the optimal numbers of k clusters for the features extracted from both years of satellite imagery. Considering the context of the paper, the number of 6 is finally assigned to k to fit the kmeans clustering model, varying labels are subsequently matched to features dataset.

[78]:
```
np.random.seed(0)
k = 6
cls = pd.Series(KMeans(n_clusters=k, max_iter = 1000, n_init = 1000,
                random_state = 24).fit_predict(merged_var))
```

After implementing k-means clustering on our constructed dataset, the label of each cluster is assigned to the last columns of data for further interpretation (as shown in Table 14).

[79]:
```
# Assign the each cluster number to the merged data
merged_var = merged_var.assign(lbls=cls)
merged_var.index = features_old_var.index
# last columns represent class labels
merged_var.head().style.set_caption('Table 14: Assign cluster number to each
 ... administritive area')
```

[79]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081e58550>
```

Table 13: Integrated preprocessed features identified in 1984 and 2019 seperately

	h_mean_x	s_mean_x	v_mean_x	h_std_x	s_std_x	v_std_x	h_skew_x	s_skew_x	v_skew_x	LBP0_x	LBP1_x	LBP2_x	LBP3_x	LBP4_x
0	0.636975	0.359694	0.281144	1.000000	0.422465	0.580121	1.000000	0.417053	0.568198	0.286043	0.328943	0.588494	0.717523	0.837853
1	0.363843	0.354334	0.396450	0.295018	0.453664	0.504220	0.000000	0.443305	0.483028	0.316677	0.441716	0.627346	0.537772	0.441710
2	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.081547	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	1.000000	0.990530	0.000000	0.906183	1.000000	0.987872	0.633860	1.000000	0.987806	0.915923	1.000000	0.915397	0.792921	0.738052
4	0.515153	0.480226	0.329302	0.685631	0.594329	0.597572	0.662480	0.590183	0.573137	0.499704	0.564467	0.676601	0.528034	0.370871

	LBP5_x	LBP6_x	LBP7_x	LBP8_x	LBP9_x	veg_mean_x	builtup_mean_x	h_mean_y	s_mean_y	v_mean_y	h_std_y	s_std_y	v_std_y	h_skew_y
0	0.613055	0.564716	0.332298	0.714937	0.032829	0.647652	0.766995	0.442195	0.278123	0.301061	0.463123	0.308221	0.547531	0.504942
1	0.551933	0.467707	0.395850	0.744082	0.042063	0.654899	0.764480	0.446860	0.212560	0.370616	0.659603	0.254002	0.455458	0.773771
2	0.000000	0.000000	0.000000	1.000000	0.000000	0.790941	0.730085	0.000000	0.000000	1.000000	0.022307	0.002006	0.000000	0.805338
3	0.741784	1.000000	1.000000	0.507310	0.110712	0.142243	0.844884	0.828667	1.000000	0.074677	0.603370	1.000000	0.990252	0.244056
4	0.677669	0.542270	0.482237	0.700451	0.059718	0.401200	0.686044	0.495392	0.302755	0.298842	0.634913	0.353666	0.566728	0.605083

	s_skew_y	v_skew_y	LBP0_y	LBP1_y	LBP2_y	LBP3_y	LBP4_y	LBP5_y	LBP6_y	LBP7_y	LBP8_y	LBP9_y	veg_mean_y	builtup_mean_y
0	0.338747	0.541837	0.526221	0.472274	0.548142	0.506561	0.410501	0.406220	0.494801	0.499487	0.726866	0.060848	0.856451	1.000000
1	0.271126	0.448129	0.513894	0.534312	0.450686	0.349686	0.216767	0.500209	0.416926	0.544291	0.749995	0.058415	0.893850	0.699181
2	0.000000	0.000000	0.007423	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.581435
3	1.000000	0.982533	0.766492	0.819500	0.928936	1.000000	0.994141	0.852811	1.000000	0.938629	0.493097	0.093827	0.020274	0.393903
4	0.375880	0.556374	0.643383	0.640062	0.616412	0.415111	0.148374	0.630101	0.544153	0.599054	0.701302	0.071432	0.822844	0.427538

5 Interpretation

To understand the analysis result, the mean of each feature across each cluster can be calculated to uncover the feature differences among clusters. A categorical bar-plot shown below presents how the average of all features changed between 1984 and 2019. Besides, a choropleth map is created to visualise the spatial distribution of categories/clusters by varying colours.

[80]:
```
# calculate the mean of features for each class
k6_mean = merged_var.groupby('lbls').mean()
k6_mean.style.set_caption('Table 15: Mean values of each feature at each cluster for
... different years')
```

[80]:
```
text/html<pandas.io.formats.style.Styler at 0x1f081654b00>
```

Table 15 displays the mean values of all features in two years at varying groups. For more interpretability, a few data munging steps are required to generate visual representations.

[81]:
```
# Rearrange our data in a way that every row is one feature in a class
k6_mean = k6_mean.stack()
k6_mean.head()
```

[81]:
```
lbls
0   h_mean_x    0.803863
    s_mean_x    0.749195
    v_mean_x    0.146109
    h_std_x     0.895068
    s_std_x     0.749907
dtype: float64
```

[82]:
```
# convert multi-indices into single index

k6_mean = k6_mean.reset_index()
# renmae the columns
k6_mean = k6_mean.rename(columns = {'lbls': 'Class','level_1': 'Features', 0: 'Values'})
# rename feature names in Feature column
old  = k6_mean.loc[k6_mean['Features'].str.contains('x') ==  True, :]
new  = k6_mean.loc[k6_mean['Features'].str.contains('y') ==  True, :]
# add a new column to represent time
old = old.assign(Time = 1984)
new = new.assign(Time = 2019)
# remove '_x' and '_y' in the table to make feature names for both years are the same
old['Features'] = old['Features'].str.replace('_x', '')
new['Features'] = new['Features'].str.replace('_y', '')
```

[83]:
```
# create a new dataframe to store the mean of each feature each cluster with time

data = pd.concat([old,new])
data.head().style.set_caption('Table 16: Tidy table represents mean values of features
 ... for each cluster at different years')
```

[83]:
```
text/html<pandas.io.formats.style.Styler at 0x1f08cee1d68>
```

Table 16 reveals different categorical information, with each row represents the number of class, the feature name, the mean value of the feature and the year when the feature is extracted. We can then visualise this table in the bar-plot in Figure 7 to understand the pattern from image features.

[84]:
```
# visualise the distribution of mean values by features, class and time

g = sns.catplot( data = data, x = 'Features', y = 'Values',row = 'Class',
            hue = 'Time',kind = 'bar', aspect = 5, height = 3, palette = 'Accent')
g.fig.suptitle('Fig.7: Visual representation of patterns extracted from k-mean
 ... clustering', y = -0.1, fontsize = 18)
```

[84]:
```
Text(0.5, -0.1, 'Fig.7: Visual representation of patterns extracted from k-mean
 clustering')
image/png<Figure size 1141.5x1296 with 6 Axes>
```

Table 14: Assign cluster number to each administrative area

Name	h_mean_x	s_mean_x	v_mean_x	h_std_x	s_std_x	v_std_x	h_skew_x	s_skew_x	v_skew_x	LBP0_x	LBP1_x	LBP2_x	LBP3_x	LBP4_x
Baoshan	0.636975	0.359694	0.281144	1.000000	0.422465	0.580121	1.000000	0.417053	0.568198	0.286043	0.328943	0.588494	0.717523	0.837853
Changning	0.363843	0.354334	0.396450	0.295018	0.453664	0.504220	0.000000	0.443305	0.483028	0.316677	0.441716	0.627346	0.537772	0.441710
Chongming	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.081547	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Fengxian	1.000000	0.990530	0.000000	0.906183	1.000000	0.987872	0.633860	1.000000	0.987806	0.915923	1.000000	0.915397	0.792921	0.738052
Hongkou	0.515153	0.480226	0.329302	0.685631	0.594329	0.597572	0.662480	0.590183	0.573137	0.499704	0.564467	0.676601	0.528034	0.370871

Name	LBP5_x	LBP6_x	LBP7_x	LBP8_x	LBP9_x	veg_mean_x	builtup_mean_x	h_mean_y	s_mean_y	v_mean_y	h_std_y	s_std_y	v_std_y	h_skew_y
Baoshan	0.613055	0.564716	0.332298	0.714937	0.032829	0.647652	0.766995	0.442195	0.278123	0.301061	0.463123	0.308221	0.547531	0.504942
Changning	0.551933	0.467707	0.395850	0.744082	0.042063	0.654899	0.764480	0.446860	0.212560	0.370616	0.659603	0.254002	0.455458	0.773771
Chongming	0.000000	0.000000	0.000000	1.000000	0.000000	0.790941	0.730085	0.000000	0.000000	1.000000	0.022307	0.002006	0.000000	0.805338
Fengxian	0.741784	1.000000	1.000000	0.507310	0.110712	0.142243	0.844884	0.828667	1.000000	0.074677	0.603370	1.000000	0.990252	0.244056
Hongkou	0.677669	0.542270	0.482237	0.700451	0.059718	0.401200	0.686044	0.495392	0.302755	0.298842	0.634913	0.353666	0.566728	0.605083

Name	s_skew_y	v_skew_y	LBP0_y	LBP1_y	LBP2_y	LBP3_y	LBP4_y	LBP5_y	LBP6_y	LBP7_y	LBP8_y	LBP9_y	veg_mean_y	builtup_mean_y	lbls
Baoshan	0.338747	0.541837	0.526221	0.472274	0.548142	0.506561	0.410501	0.406220	0.494801	0.499487	0.726866	0.060848	0.856451	1.000000	1
Changning	0.271126	0.448129	0.513894	0.534312	0.450686	0.349686	0.216767	0.500209	0.416926	0.544291	0.749995	0.058415	0.893850	0.699181	1
Chongming	0.000000	0.000000	0.007423	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	0.581435	2
Fengxian	1.000000	0.982533	0.766492	0.819500	0.928936	1.000000	0.994141	0.852811	1.000000	0.938629	0.493097	0.093827	0.020274	0.393903	3
Hongkou	0.375880	0.556374	0.643383	0.640062	0.616412	0.415111	0.148374	0.630101	0.544153	0.599054	0.701302	0.071432	0.822844	0.427538	1

Table 15: Mean values of each feature at each cluster for different years

lbls	h_mean_x	s_mean_x	v_mean_x	h_std_x	s_std_x	v_std_x	h_skew_x	s_skew_x	v_skew_x	LBP0_x	LBP1_x	LBP2_x	LBP3_x	LBP4_x
0	0.803863	0.749195	0.146109	0.895068	0.749907	0.834330	0.633322	0.720249	0.823005	0.454389	0.565946	0.821528	0.973220	0.949141
1	0.426195	0.381664	0.384955	0.543494	0.481686	0.523486	0.450601	0.478586	0.504149	0.382849	0.444543	0.574545	0.510795	0.408613
2	0.113097	0.088470	0.837305	0.210939	0.118478	0.127151	0.316506	0.107534	0.124469	0.070301	0.089850	0.162624	0.139945	0.089317
3	0.949856	0.995265	0.004559	0.876415	0.991048	0.993936	0.676387	0.975688	0.993903	0.882717	0.957562	0.957699	0.873867	0.798240
4	0.721478	0.979475	0.114017	0.567528	0.998762	0.933619	0.244557	0.971796	0.925791	1.000000	0.990273	0.845726	0.668671	0.407605
5	0.471719	0.427622	0.373358	0.634773	0.554207	0.559062	0.554391	0.560426	0.534555	0.426212	0.502654	0.647752	0.516126	0.345374

lbls	LBP5_x	LBP6_x	LBP7_x	LBP8_x	LBP9_x	veg_mean_x	builtup_mean_x	h_mean_y	s_mean_y	v_mean_y	h_std_y	s_std_y	v_std_y	h_skew_y
0	0.714188	0.767025	0.567415	0.603671	0.062757	0.336476	0.964021	0.621423	0.582989	0.125378	0.621145	0.621750	0.781063	0.466975
1	0.491368	0.480169	0.415924	0.747533	0.045784	0.579325	0.782732	0.400459	0.233562	0.370746	0.449325	0.279210	0.479726	0.520610
2	0.114726	0.140034	0.083018	0.941070	0.009137	0.739326	0.768474	0.081734	0.042801	0.835523	0.144022	0.062326	0.114256	0.757332
3	0.870892	0.996516	0.943707	0.495601	0.106946	0.071121	0.808680	0.914333	0.849361	0.087764	0.539909	0.845232	0.995126	0.122028
4	0.847049	0.819912	0.961713	0.538442	0.114071	1.000000	0.000000	0.883471	0.529215	0.000000	1.000000	0.546832	0.871779	0.698488
5	0.299768	0.528165	0.568006	0.000000	1.000000	0.549489	0.733977	0.428816	0.207932	0.329358	0.435917	0.246778	0.505767	0.409755

lbls	s_skew_y	v_skew_y	LBP0_y	LBP1_y	LBP2_y	LBP3_y	LBP4_y	LBP5_y	LBP6_y	LBP7_y	LBP8_y	LBP9_y	veg_mean_y	builtup_mean_y
0	0.629142	0.768836	0.559714	0.585206	0.760955	0.841492	0.838157	0.722830	0.794093	0.652172	0.600350	0.070401	0.714016	0.835832
1	0.300671	0.470766	0.465958	0.451277	0.510348	0.446913	0.309060	0.470514	0.466278	0.466282	0.748782	0.053443	0.852035	0.622854
2	0.077196	0.114218	0.050600	0.054194	0.140384	0.172060	0.158486	0.147885	0.115339	0.054215	0.939642	0.006619	0.959421	0.572280
3	0.870699	0.991266	0.883246	0.909750	0.964468	0.940470	0.808656	0.863404	0.987573	0.969315	0.488089	0.104292	0.010137	0.196952
4	0.575903	0.866917	0.981285	0.905575	0.907557	0.660840	0.487015	1.000000	0.950033	0.913027	0.521429	0.107485	0.713309	0.986139
5	0.280930	0.499205	0.621626	0.525659	0.570189	0.416572	0.116072	0.221324	0.445842	0.718360	0.000000	1.000000	0.895112	0.568988

Figure 7: Visual representation of patterns extracted from k-mean clustering

[85]:
```
# plot clustering results for two different years
f, ax = plt.subplots(1, figsize=(10, 12))
# plot cluster results
poly = poly.drop('coords', axis = 1)
poly.assign(lbls=cls)\
     .plot(column='lbls', categorical=True, linewidth=1, alpha=0.5, ax=ax,
           legend = True,cmap = 'Accent', edgecolor = 'black')
# add labels for geographical units
poly['coords']=poly['geometry'].apply(lambda x:x.representative_point().coords[:])
poly['coords']=[coords[0] for coords in poly['coords']]
for idx, row in poly.iterrows():
    ax.annotate(text=row['Name'],xy=row['coords'],va='center',ha='center',
                alpha = 0.8, fontsize = 10)
plt.title('Fig.8: Spatial distribution of classification results', y=-0.01)
# remove axes and set aspect ratio so that the data units are the same in every direction
ax.axis('off')
ax.axis('equal')
```

[85]:
```
(290053.0696196473, 407301.6741094636, 3389866.639388826, 3533566.430983904)
image/png<Figure size 720x864 with 1 Axes>
```

Table 16: Tidy table represents mean values of features for each cluster at different years

	Class	Features	Values	Time
0	0	h_mean	0.803863	1984
1	0	s_mean	0.749195	1984
2	0	v_mean	0.146109	1984
3	0	h_std	0.895068	1984
4	0	s_std	0.749907	1984

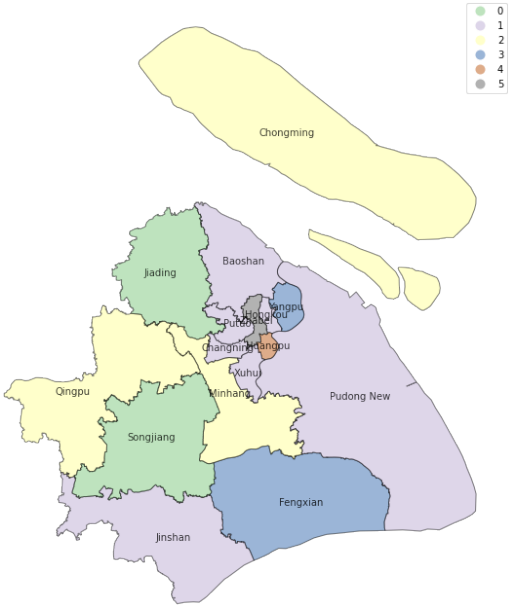


Figure 8: Spatial distribution of classification results

From Figures 7 and 8 we can see a few striking differences across clusters, or classes. For class 4, only one administrative area (i.e. Huangpu area) is grouped, displayed in the middle of north-east areas. The mean values for this class are mostly high in both years except a couple of features such as v_mean, LBP4 and LBP9 features. The brightness (v_mean) for this area is highly low and it became completely black over time. H_mean value is high in both years, demonstrating that the dominating colour is blue, which represent water. This corresponds to the famous area of The Bund, with its river skyline, which is part of this polygon. The vegetation built-up features indicate that this area has experienced a remarkable change, from more vegetation and few buildings to less vegetation and completely constructed/urbanisation.

Class 0 and Class 1 are relatively consistent compared to other classes, implying that the urban areas in purple and green colours almost remained unchanged during the past 35 years. Besides, these two classes have similar transformation such as more vegetation coverage and less buildings for the current year of 2019. However, Class 0 has more brightness and more green colour based on v_mean, h_mean and veg_mean features, and Class 1 has higher h_mean, h_std, h_skew and built-up_mean, implying these two areas have water covered and were highly constructed.

Class 2 distributed at north and middle-west areas in the map, which is extremely diverse and unique among all categories. It has the highest brightness features and LBP8 texture features, while the rest mean values of colour and texture features are highly low, especially for LBP9 where almost zero values in both years. The values for h_mean, s_mean and v_mean display that the primary colour for these areas is red with little grey and much brightness, representing that these areas include more bare ground or soil and thus probably rural areas. Adversely, Class 5 has zero values for LBP8 but highest

values for LBP9 in both years. It contains only one administrative area (i.e. Zhabei area), surrounded by Class 4 and Class 0. Similarly, the area in Class 5 has more vegetation but slightly less built-up areas over the past years. Class 3 contains two areas distributed at the south and surrounded by Class 1 from the map. The feature values in Class 3 are mostly extremely high, while the veg_mean and built-up_mean for current year are the least, thus indicating that these areas have more water over the time.

6 Conclusion

Urbanisation has significantly changed the interaction between humans and the surrounding environment, which poses new challenges in a multitude of fields including construction and city planning, hazard mitigation or disease control. It is essential to quantify and assess urbanisation over time to enable policy makers and planners to make informed decisions about future urban changes. The sustainability of urban spaces will become particularly important in the light of future climate change. Satellite imagery could play a vital role in assessing cities for their livability by i.e. quantifying the greenspace to built environment ratio. This notebook shows the potential of open source satellite imagery to exploring urban changes and proposes a simple method framework for automatic data collection and features extraction to determine urbanisation over time using Python as a tool.

References

Barsi JA, Lee K, Kvaran G, Markham BL, Pedelty JA (2014a) The spectral response of the Landsat-8 operational land imager. *Remote Sensing* 6: 10232–10251. CrossRef.

Barsi JA, Schott JR, Hook SJ, Raqueno NG, Markham BL, Radocinski RG (2014b) Landsat-8 thermal infrared sensor (TIRS) vicarious radiometric calibration. *Remote Sensing* 6: 11607–11626. CrossRef.

Burchfield M, Overman HG, Puga D, Turner MA (2006) Causes of sprawl: A portrait from space. *The Quarterly Journal of Economics* 121: 587–633. CrossRef.

Giada S, De Groeve T, Ehrlich D, Soille P (2003) Information extraction from very high resolution satellite imagery over Lukole refugee camp, Tanzania. *International Journal of Remote Sensing* 24: 4251–4266. CrossRef.

Glaeser E, Henderson JV (2017) Urban economics for the developing world: An introduction. *Journal of Urban Economics* 98: 1–5. CrossRef.

Ibrahim MR, Haworth J, Cheng T (2020) Understanding cities with machine eyes: A review of deep computer vision in urban analytics. *Cities* 96. CrossRef.

Keen N (2005) Color moments. School of informatics, University of Edinburgh

Kit O, Lüdeke M (2013) Automated detection of slum area change in Hyderabad, India using mulittemporal satellite imagery. *Journal of Photogrammetry and Remote Sensing* 83: 130–137. CrossRef.

Knight EJ, Kvaran G (2014) Landsat-8 operational land imager design, characterization and performance. *Remote Sensing* 6: 10286–10305. CrossRef.

Kohli D, Sliuzas R, Stein A (2016) Urban slum detection using texture and spatial metrics derived from satellite imagery. *Journal of Spatial Science* 61: 405–426. CrossRef.

Ministry of Civial Affairs of the People's Republic of China (2018) Change of administrative divisions at or above the county level. Available at: http://202.108.98.30/description?-dcpid=1 [Accessed 10 Oct. 2019]

NASA (2000) Normalized difference vegetation index (NDVI). Available at: https://earthobservatory.nasa.gov/features/MeasuringVegetation/measuring_vegetation_2.php [Accessed 30 Oct. 2019]

NASA (2019) Landsat science. Available at: https://landsat.gsfc.nasa.gov/ [Accessed 10 Sep. 2019]

Ojala T, Pietikäinen M, Harwood D (1996) A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition* 19: 51–59. CrossRef.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J (2011) Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12: 2825–2830

Roy DP, Wulder MA, Loveland TR, Woodcock CE, Allen RG, Anderson MC, Helder D, Irons JR, Johnson DM, Kennedy R, Scambos TA, Schaaf CB, Schott JR, Sheng Y, Vermote EF, Belward AS, Bindschadler R, Cohen WB, Gao F, Hipple JD, Hostert P, Huntington J, Justice CO, Kilic A, Kovalskyy V, Lee ZP, Lymburner L, Masek JG, McCorkel J, Shuai Y, Trezza R, J. Vogelmann J, R.H. Wynne RH, Zhu Z (2014) Landsat-8: Science and product vision for terrestrial global change research. *Remote sensing of Environment* 145: 154–172. CrossRef.

United Nations (2019) World urbanization prospects 2018: Highlights. United Nations, Department of Economic and Social Affairs, Population Division (ST/ESA/SER.A/421)

Zha Y, Gao J, Ni S (2003) Use of normalized difference built-up index in automatically mapping urban areas from TM imagery. *International Journal of Remote Sensing* 24: 583–594. CrossRef.

REGION
The Journal of ERSA
Powered by WU

Volume 6, Number 3, 2019, 53–69
DOI: 10.18335/region.v6i3.277

journal homepage: region.ersa.org

Exploring long-term youth unemployment in Europe using sequence analysis: A reproducible notebook approach*

Nikos Patias[1]

[1] University of Liverpool, Liverpool, UK

Received: 28 August 2019/Accepted: 17 January 2020

Abstract. Youth unemployment is an important factor influencing the lifetime earnings and future job prospects of individuals, often resulting in deterioration in their health and well-being. Youth unemployment in Europe has been affected by the financial crisis of 2008. However, the magnitude of these effects varied across European countries. The objective of this notebook is to identify representative trajectories of youth unemployment change in Europe from 2008 to 2018. This notebook provides a self-contained research workflow that is fully reproducible and transparent. My findings suggest that northern Europe has high concentration of regions with stable low youth unemployment while southern Europe has high concentration of regions with stable high youth unemployment. Identifying key patterns of youth unemployment change among European countries can provide useful insights that help to understand migration patterns originating from the more "disadvantaged" regions to more "advantaged" ones, or beyond. Finally, I hope that data and regional scientists can benefit by the functionalities offered in this notebook and use it as a complementary guide for analysing their own data.

Key words: sequence analysis, unemployment, Europe, regional inequalities, reproducible research

1 Introduction

Youth unemployment is an important factor influencing the lifetime earnings and future job prospects of individuals, often resulting in deterioration in their health and well-being (Bell, Blanchflower 2011, O'Reilly et al. 2015). The effects of financial crisis of 2008 on youth unemployment were prominent and varied across European countries and regions. The European average for youth unemployment culminated in 2012 to more than 20%, but there were countries that scored much higher (i.e. more than 50% in Greece and more that 30% in Bulgaria and Italy) (Dietrich 2012). However, regional variations can help to contextualise and analyse patterns of youth unemployment more effectively (Pop et al. 2019). Understanding and tracking the evolution of regions that faced high levels of unemployment can help in planning future policies, as today's youth are going to be in the workforce for the next 50 years. Finally, the trajectories of youth unemployment change across regions (i.e. whether they have successfully recovered or not) and can be

*This paper is available as computational notebook on the REGION webpage.

linked to patterns of regional resilience against economic crises. In this notebook, I use a sequence analysis approach to identify representative trajectories of youth unemployment change by NUTS 2 regions in Europe from 2008 to 2018.

The idea of using reproducible analyses in computational research has a growing number of advocates (Peng 2011, Sandve et al. 2013, Rule et al. 2019). The development of computational notebooks such as R and Jupyter notebooks, allow scientists to incorporate code, documentation, graphs and text in a single document. Consequently, more than ever before, computational research has become more open, transparent and fully replicable. Peng (2011) developed a reproducibility spectrum to highlight the importance of incorporating publication standards text with linked code and data to achieve the "gold standard of reproducibility". The spectrum begins from the traditional static publications, which are not reproducible. They become more reproducible when code and data are incorporated in the publication. Finally, the full replication is achieved when linked and executable code and data are included. As Peng (2011) highlights, data is an integral part of reproducible research and should be clearly documented within the workflow. However, researchers often neglect to provide adequate information on the datasets used. As a result, other researchers have difficulties on replicating this piece of research. Linked datasets through direct web-links or Application Programming Interfaces (APIs) when available, contribute to the transparency of the research, by explicitly pointing the end-user to the source of information described in a research project.

The objective of this notebook is to identify key representative trajectories of youth unemployment change in Europe from 2008 to 2018. In the present notebook I provide a self-contained research workflow that is fully reproducible and transparent. Moreover, I make use of the functionalities offered by computational notebooks written in R markdown such as direct access to online tabular/spatial datasets, manipulation and linkage between these datasets as well as interactive plots/maps. Finally, this notebook aims to provide the sufficient tools that a data or regional scientist needs to perform similar types of analysis.

2 Packages and Dependencies

This section is used to report all the packages and dependencies required to run this notebook which are vital components of reproducible research. By reporting the R version under which I created this notebook and the packages used will ensure its replicability.

Firstly, is important to report the R version used in this notebook by running the following line of code.

[1]:
```
# to get the version of R used in the notebook
paste("The R Version used in this notebook is", getRversion())
```

[1]:
```
## [1] "The R Version used in this notebook is 3.5.1"
```

I then specify the CRAN repository where the packages have been downloaded from.

[2]:
```
# Define the CRAN repository for this session
r_rep = getOption("repos")
r_rep["CRAN"] = "http://cran.us.r-project.org"
options(repos = r_rep)
```

And install/load the packages required to run this notebook. Please note that the installation stage is required only the first time you run this notebook.

[3]:
```
# These are the packages required to run this notebook
# First should be installed
# install.packages("eurostat")
# install.packages("rvest")
# install.packages("knitr")
# install.packages("rgdal")
# install.packages("countrycode")
# install.packages("dplyr")
# install.packages("reshape2")
# install.packages("ggplot2")
# install.packages("TraMineR")
# install.packages("cluster")
```

```
# install.packages("factoextra")
# install.packages("RColorBrewer")
# install.packages("leaflet")
# install.packages("plotly")
# And then should be loaded
library(eurostat)
library(rvest)
library(knitr)
library(rgdal)
library(countrycode)
library(dplyr)
library(reshape2)
library(ggplot2)
library(TraMineR)
library(cluster)
library(factoextra)
library(RColorBrewer)
library(leaflet)
library(plotly)
```

Finally, I create a list of the available packages in my R environment and report the version of each package used.

[4]:
```
# Create a list with all the available packages in my R environment
pkg_used <- available.packages()
```

[5]:
```
# For every package print the version of the package, the version of R that depends
# on and the packages that imports
paste("eurostat Version is:", pkg_used["eurostat", "Version"])
paste("rvest Version is:", pkg_used["rvest", "Version"])
paste("knitr Version is:", pkg_used["knitr", "Version"])
paste("rgdal Version is:", pkg_used["rgdal", "Version"])
paste("countrycode Version is:", pkg_used["countrycode", "Version"])
paste("dplyr Version is:", pkg_used["dplyr", "Version"])
paste("reshape2 Version is:", pkg_used["reshape2", "Version"])
paste("ggplot2 Version is:", pkg_used["ggplot2", "Version"])
paste("TraMineR Version is:", pkg_used["TraMineR", "Version"])
paste("cluster Version is:", pkg_used["cluster", "Version"])
paste("factoextra Version is:", pkg_used["factoextra", "Version"])
paste("RColorBrewer Version is:", pkg_used["RColorBrewer", "Version"])
paste("leaflet Version is:", pkg_used["leaflet", "Version"])
paste("plotly Version is:", pkg_used["plotly", "Version"])
```

[5]:
```
## [1] "eurostat Version is: 3.4.20002"
## [1] "rvest Version is: 0.3.5"
## [1] "knitr Version is: 1.27"
## [1] "rgdal Version is: 1.4-8"
## [1] "countrycode Version is: 1.1.0"
## [1] "dplyr Version is: 0.8.3"
## [1] "reshape2 Version is: 1.4.3"
## [1] "ggplot2 Version is: 3.2.1"
## [1] "TraMineR Version is: 2.0-14"
## [1] "cluster Version is: 2.1.0"
## [1] "factoextra Version is: 1.0.6"
## [1] "RColorBrewer Version is: 1.1-2"
## [1] "leaflet Version is: 2.0.3"
## [1] "plotly Version is: 4.9.1"
```

In this notebook, I have installed the latest versions of the packages used. I understand that the analysis can be run by using previous versions too. However, using the versions of the packages as reported here ensures the reader that this notebook will run without any errors.

Now that all the packages have been correctly installed it is useful to provide a brief overview of the main functionalities of each package.

`eurostat` This package allows access to Eurostat data through their API.

`rvest` This package is used to scrape data from web pages.

`knitr` This package provides better visualisation of the results within the notebook (i.e. table formatting).

`rgdal` This package is used to read, merge and manipulate geospatial datasets.

`countrycode` This package is used to convert country codes (i.e. ISO 3166) to country names.

`dplyr` This package is used for more effective dataframes' manipulation.

`reshape2` This package is used to reshape tables from wide to long format and vice versa.

`ggplot2` This package is used for creating plots.

`TraMineR` This is the package is used to perform sequence analysis.

`cluster` This package is used to perform cluster analysis.

`factoextra` This package provides the functionalities to assess the optimal number of clusters.

`RColorBrewer` This package provides colour palettes to be used in maps.

`leaflet` This package is used for interactive mapping.

`plotly` This package is used for interactive plotting in conjuction with `ggplot2`.

3 Data and Methods

3.1 Data

Eurostat (https://ec.europa.eu/eurostat/data/database) has a large database providing a wide range of available datasets in varying geographies and time frames. In this notebook, I analyse youth unemployment in Europe from 2008 to 2018. Eurostat captures youth unemployment by measuring the percentage of "Young people neither in employment nor in education and training (NEET rates)". This dataset is available from 2000 to 2018 at NUTS 2 regions (more information on NUTS classification can be found at https://ec.europa.eu/eurostat/web/nuts/background). Eurostat defines youth unemployment either people aged 15-24 or 18-24 and are unemployed. In this study, I consider the percentage of people aged between 18 and 24. The original dataset used in this notebook can be accessed following https://ec.europa.eu/eurostat/web/products-datasets/-/edat_lfse_22. Eurostat has created its own R package (https://cran.r-project.org/web/packages/eurostat/index.html) to allow access in the database through an API with a comprehensive documentation (https://ropengov.github.io/eurostat/index.html) and tutorial (http://ropengov.github.io/eurostat/articles/eurostat_tutorial.html). In order to make the most of the functionalities offered by a notebook as well as enable researchers to replicate this approach I make use of Eurostat's API to access the dataset analysed in this notebook.

I first search for the datasets referring to young unemployed people.

[6]:
```
# search information about the datasets that are related to young unemployed people
kable(head(search_eurostat("Young people neither in employment")))
```

[6]:
```
Output in Table 1
```

Of the available datasets, I am interested in the first on this list with code `edat_lfse_22`. I specified my request to 11 time periods. Starting from the most current year (i.e. 2018) and going back to 2008. I also specified that I want total percentages (i.e. both male and female - `sex = "T"`) and finally the age group that covers people aged from 18 to 24.

[7]:
```
# Specify the ID of the dataset required
id <- "edat_lfse_22"
# Request of the dataset
young_unempl <- get_eurostat(id, filters = list(lastTimePeriod=11, sex = "T",
                     age = "Y18-24"), time_format = "num")
```

Table 1: Available datasets from Eurostat related to youth unemployment

title	code	type	last update of data	last table structure change	data start	data end	values
Young people neither in employment nor in education and training by sex and NUTS 2 regions (NEET rates)	edat_lfse_22	dataset	01.07.2019	13.08.2019	2000	2018	NA
Young people neither in employment nor in education and training by sex, age and degree of urbanisation (NEET rates)	edat_lfse_29	dataset	01.07.2019	13.08.2019	2000	2018	NA
Young people neither in employment nor in education and training by type of disability, sex and age	hlth_de030	dataset	21.03.2019	21.03.2019	2011	2011	NA
Young people neither in employment nor in education and training by sex, age and labour status (NEET rates)	edat_lfse_20	dataset	01.07.2019	13.08.2019	2000	2018	NA
Young people neither in employment nor in education and training by sex, age and citizenship (NEET rates)	edat_lfse_23	dataset	25.04.2019	13.08.2019	2004	2018	NA
Young people neither in employment nor in education and training by sex, age and country of birth (NEET rates)	edat_lfse_28	dataset	25.04.2019	13.08.2019	2004	2018	NA

I also make use of a spatial dataset to enable use of an interactive map to facilitate better presentation of results. To achieve that, I have downloaded NUTS 2 regions shapefile from the second version of Eurostat's spatial database (https://ec.europa.eu/eurostat/cache/GISCO/distribution/v2/). Eurostat provides the spatial data as a bulk download, so I first unzip the file and then select the shapefile that matches the tabular data that I have already downloaded. The name of the dataset "NUTS_RG_60M_2016_4326_LEVL_2.shp.zip" is self-explanatory showing that the spatial data is projected in the World Geodetic System of 1984 (i.e. WGS84 or EPSG:4326) for NUTS 2 regions in 2016.

[8]:
```
# Specify the url that links to the zipped spatial datasets
url = "http://ec.europa.eu/eurostat/cache/GISCO/distribution/v2/nuts/download/
...ref-nuts-2016-60m.shp.zip"
# Download the file
download.file(url, basename(url))
# Unzip the bulk file
unzip(basename(url))
# Unzip the specific shapefile needed
unzip(paste0(getwd(), "/NUTS_RG_60M_2016_4326_LEVL_2.shp.zip"))
# Read in the shapefile
geodata <- readOGR(dsn = getwd(), layer = "NUTS_RG_60M_2016_4326_LEVL_2")
```

Eurostat provides only country codes, which is not always helpful when presenting results. For this reason, I used the R package `countrycode` (https://cran.r-project.org/web/packages/countrycode/index.html) to convert country codes to country names. While in general, the European commission uses ISO 3166-1 alpha-2 codes, there are two exceptions. Greece is reported as "EL" (rather than "GR") and United Kingdom as "UK" (rather than "GB"). Thus, I recoded these two countries manually.

[9]:
```
# Create a new column for country names
geodata@data$cntr_name <- countrycode(geodata@data$CNTR_CODE, "iso2c", "country.name")
# Because European commission uses EL for Greece (in ISO 3166-1 alpha-2 codes is GR)
# and UK for United Kingdom (in ISO 3166-1 alpha-2 codes is GB) I should replace these
# two countries manually
geodata@data$cntr_name <- ifelse(geodata@data$CNTR_CODE=="EL","Greece",
ifelse(geodata@data$CNTR_CODE=="UK", "United Kingdom", geodata@data$cntr_name))
```

3.2 Methods

This notebook aims to identify representative trajectories of youth unemployment change across NUTS 2 regions in Europe. The methodological workflow followed here is similar to Patias et al. (2020) that analyses trajectories of neighbourhood change in Great Britain. Sequence analysis is a method that analyses sequences of categorical variables, and extracts information on their structure and evolution. Sequence analysis has its origins in biology, where it is used to analyse DNA sequences (Sanger et al. 1977). It can also be applied to analyse longitudinal individual-level family, migration and career trajectories (Brzinsky-Fay 2007, Rowe et al. 2017a,b). This method is also used on neighbourhood trajectory mining in the United States to identify patterns of socioeconomic change over

a period of time (Delmelle 2016). The key component of sequence analysis method is the optimal matching analysis which is used to measure pairwise dissimilarities between sequences and identifies "types of sequence patterns" (Studer, Ritschard 2016). In this notebook sequence analysis is used in a spatio-temporal concept assessing how youth unemployment in European regions (i.e. spatial) has changed from 2008 to 2018 (i.e. temporal). Sequence analysis has been used as it is a method capable of capturing multiple dimensions of spatio-temporal processes namely incidence, duration, timing and sequencing. The youth unemployment data downloaded from Eurostat is expressed in percentages (by NUTS 2 regions). Sequence analysis can be applied to categorical data, hence I had to classify the regions' youth unemployment percentages into quintiles so that they can be treated as categories. In this way, the multiple dimensions of spatio-temporal processes of youth unemployment change can be systematically measured. Thus, it explicitly captures for each region:

- The number of times in a particular quintile (i.e. incidence);
- The time span in a particular quintile (i.e. duration);
- The year at occurrence of change from one quintile to another (i.e. timing); and
- The chronological order of transitions between quintiles (i.e. sequencing).

The key stages followed in this notebook are described below with links to the particular sub-sections which provide some further technical clarifications:

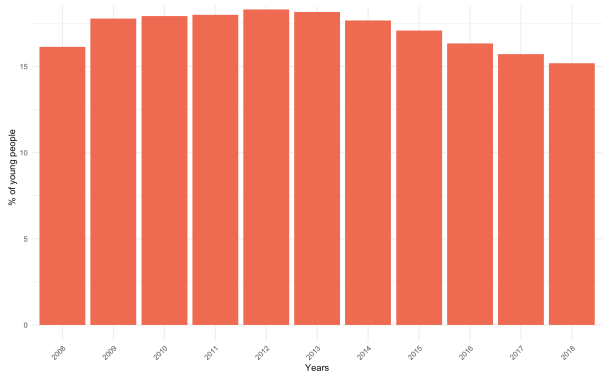
1. *Data pre-processing*, by classifying NUTS 2 regions into quintiles based on the percentage of youth unemployment in each year from 2008 to 2018 (regions with the lowest % youth unemployment belong to the 1st quintile and regions with the highest % youth unemployment belong to the 5th quintile).
2. Create a *sequence object* based on the quintile each region belongs to in every year (i.e. from 2008 to 2018).
3. *Measuring sequence dissimilarity* based on substitution costs which is the probability of transitioning from one quintile to another (i.e. higher transition rate from 1st quintile to 5th quintile rather than from 2nd quintile to 1st quintile). The substitution costs between quintiles i and j are calculated based on Equation (1).
4. Using the substitution costs calculated in the previous stage, I built a *dissimilarity matrix* including every pair of sequences. In this notebook I have used the Optimal Matching (OM) algorithm. The algorithm substitutes the elements of each sequence based on their substitution costs which in turn is the OM distance between each pair of sequences.
5. In the last stage I produce I typology of youth unemployment trajectories using the resulting dissimilarity matrix from stage 4. Partitioning Around Medoids (PAM) clustering algorithm is used for the *classification of sequences*

$$SubsCosts_{i,j} = 2 - p(i|j) - p(j|i) \tag{1}$$

where $p(i|j)$ is the transition rate between quintiles i and j. For the sequence analysis I have used R package `TraMineR` (https://cran.r-project.org/web/packages/TraMineR/index.html) which provides all the required functionalities.

4 Data Analysis

As shown in Figure 1, the average percentage of young people who are neither in employment nor in education and training in Europe increased from 16% in 2008 to 18.5% in 2012, followed by a decrease in 2018 (i.e. at around 15%). The results suggest that on average, regions show patterns of resilience against financial crises and that policies



Notes: Data from Eurostat, calculations by the author

Figure 1: European % average of young people neither in employment nor in education and training

targeting the decrease of youth unemployment have proven efficient. However, not all regions follow the same patterns.

This section of the notebook aims to provide an understanding on long-term youth unemployment patterns in NUTS 2 regions in Europe but also to guide the reader on the analytical process of sequence analysis and the functionalities offered by computational notebooks. Each of the following five sub-sections will present in detail each of the steps followed for the production of the results.

[10]:
```
# I change the years from numbers to characters so to be recongised as categorical
# rather than continuous variable
young_unempl$time <- as.character(young_unempl$time)
# Create a plot by showing the European % average of young people neither in employment
# nor in education and training
young_unempl %>%
  group_by(time) %>%
  summarise_all(mean, na.rm = TRUE) %>%
  ggplot()
  geom_bar(aes(x = time, y = values), stat = "identity", fill = "coral2")
  labs(title = "European average % of young unemployed people",
     x = "Years",
     y = "% of young people",
     caption = "Data downloaded from Eurostat\ncalculations made by the author")
  theme_minimal()
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

[10]:
```
Output in Figure 1
```

4.1 Data pre-processing

While the datasets provided by Eurostat are in clean format, they almost always require some data pre-processing. Here, there are three main tasks required to bring the data in the required format to proceed with the analysis. First, to subset the dataset to have only the NUTS 2 regions (the original dataset also includes country and NUTS 1 data). Second, to calculate the quintiles that every NUTS 2 region belongs to in every year. Third, to re-format the dataset from a long (each region represented in multiple rows – one for every year) to a wide format (each region represented by a single row and there are multiple columns containing the yearly young unemployment rates).

In coding terms, the first task is to calculate the number of characters of all geography codes and store them in a new column. I then create a subset of the dataset with only NUTS 2 regions which are those that contain four characters (i.e. the first two represent the country name followed by two numbers represent the NUTS 2 region).

Table 2: Preview of the data that will be used in sequence analysis

	sex	age	training	wstatus	unit	geo	n_char	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
2	T	Y18-24	NO_FE_NO_NFE	NEMP	PC	AT12	4	2	2	1	2	1	1	2	1	1	1	1
3	T	Y18-24	NO_FE_NO_NFE	NEMP	PC	AT13	4	2	2	2	2	2	2	2	3	3	2	3
5	T	Y18-24	NO_FE_NO_NFE	NEMP	PC	AT22	4	1	2	1	1	1	1	1	1	1	1	1
6	T	Y18-24	NO_FE_NO_NFE	NEMP	PC	AT31	4	1	1	1	1	1	1	1	1	1	1	1
8	T	Y18-24	NO_FE_NO_NFE	NEMP	PC	AT33	4	1	1	1	1	1	1	1	1	1	1	1
10	T	Y18-24	NO_FE_NO_NFE	NEMP	PC	BE10	4	5	4	4	4	4	4	4	4	4	4	4

[11]:
```
# Create a new column to store the number of characters of the geography
young_unempl$n_char <- nchar(as.character(young_unempl$geo))
# Subset only the NUTS 2 regions - their geography code contains 4 characters
young_unempl_NUTS2 <- young_unempl %>%
  filter(n_char == 4)
```

The next task is to calculate quintiles by year for each region based on their % of youth unemployment. This was done by looping through each year.

[12]:
```
# Calculate quintiles by year
# It is good to specify  the filter function to be used from dplyr function to avoid
# error messages
quant_data <- NULL
for (var in unique(young_unempl_NUTS2$time)) {
    young_unempl_NUTS2_temp <- young_unempl_NUTS2 %>%
      dplyr::filter(time == var) %>%
      mutate(quintiles = ntile(values, 5) )
    quant_data <- rbind(quant_data,young_unempl_NUTS2_temp)
}
```

The final task is to keep only the column containing the quintiles (the actual percentages will not be used in the rest of this notebook). The dataset will then be re-formatted to a wide format where each region will be represented by a single row. For each region there are multiple columns containing the corresponding yearly young unemployment quintiles. Finally, I delete all the rows that contain missing values. This is important as there are regions that have "gaps" in their data availability, meaning that there is no data in at least one year between 2008 and 2018. These regions are ignored in this analysis to speed up computational time and to have consistency across sequences.

[13]:
```
# I delete the column inlcuding the % as I will use the quintiles from now on in the
# analysis
quant_data <- subset(quant_data, select = -values)
# Re-format the data from long to wide format
# This means that every row will represent a region and every column represents a year
quant_data_wide <-
  dcast(quant_data,
  sex  age  training  wstatus  unit  geo  n_char ~ time,
  value.var = 'quintiles')
# We remove rows that  do not have values in at least one year so we have consistency
# between sequences
quant_data_wide <- na.omit(quant_data_wide)
# Have a look at the dataset
kable(head(quant_data_wide))
```

[5]:
```
Output in Table 2
```

4.2 Sequence object

Creating a sequence object is the initial point of sequence analysis. In this notebook I pass a subset of the columns of the dataset that contain the quintile values. Practically, it means that I create a sequence of quintiles from 2008 to 2018 which are from the 8th to 18th column for each region. As I have already mentioned, I have used the R package `TraMineR` (https://cran.r-project.org/web/packages/TraMineR/index.html). For more detailed information on sequence analysis and all the functionalities, please refer to the user guide (http://mephisto.unige.ch/pub/TraMineR/doc/TraMineR-Users-Guide.pdf) which provides detailed information on all the functionalities of the package.

Table 3: Substitution costs between quintiles

	1	2	3	4	5
1	0.000000	1.694604	1.985533	2.000000	2.000000
2	1.694604	0.000000	1.591372	1.960797	2.000000
3	1.985533	1.591372	0.000000	1.654430	2.000000
4	2.000000	1.960797	1.654430	0.000000	1.752492
5	2.000000	2.000000	2.000000	1.752492	0.000000

[14]:
```
# Create the sequence object using only the quintiles that every region belongs
seq_obj <- seqdef(quant_data_wide[,8:18])
```

4.3 Measuring sequence dissimilarity

A key element of sequence analysis is to calculate "distances" between each pair of sequences that can be used later for the Optimal Matching analysis. These distances are a measure based on how similar two sequences are. There are two components related to these distances. First is the insertion/deletion (indel) cost which is used when the length of sequences is not the same. This is the cost of deleting or inserting a state in a sequence so all the sequences have the same length. On this notebook, the time period covered is the same for every region (i.e. from 2008 to 2018) so the sequence length is fixed, an 11-state long sequence. Hence this step is not required.

The second component for calculating "distances" between each pair of sequences is to calculate the substitution costs for transforming one state (i.e. one quintile group) to another. Substitution costs can be theory-driven or empirically-driven (Salmela-Aro et al. 2011). Theory-driven costs are usually used when researchers define costs based on pre-determined concepts. Thus, the costs between states are solely dependent on the researchers' choices (i.e. how "far" is one state from another). On the other hand, empirically-driven costs are based on the observed transitions between states. Hence, two states are closer when there are more observed transitions between them. In this notebook I follow the empirically-driven approach as I intend to explicitly consider the observed transitions between states (i.e. quintile groups here).

By following the empirically-driven approach, there are two options to calculate substitution costs. The first is to assign a constant value for substituting sequence states (i.e. quintiles). The second option is to calculate transition rates which are the probabilities of transitioning from one state to another (between quintiles in this notebook). These transition rates are then used to calculate the substitution costs as shown in Equation (1). In this analysis, I have used transition rates (i.e. `method = "TRATE"`) because it is important to capture the higher probability of transitioning between 1st and 2nd quintile compared to 1st and 5th quintile. By assigning a constant value, this information would have been missed. For more detailed information on substitution costs please refer again to the `TraMineR` user guide (http://mephisto.unige.ch/pub/TraMineR/doc/TraMineR-Users-Guide.pdf).

Table 3 shows the substitution costs of this study. It is clear from the table that the probability of transitioning between 1st and 2nd quintile is higher than transitioning from 1st to 5th quintile. Hence, the substitution cost from 1st to 2nd is lower than the 1st to 5th. This information will then be used in the next step – the Optimal Matching.

[15]:
```
# Calculate substistution costs
subs_costs <- seqsubm(seq_obj, method = "TRATE")
# Print the substitution costs
kable(subs_costs)
```

[5]:
```
Output in Table 3
```

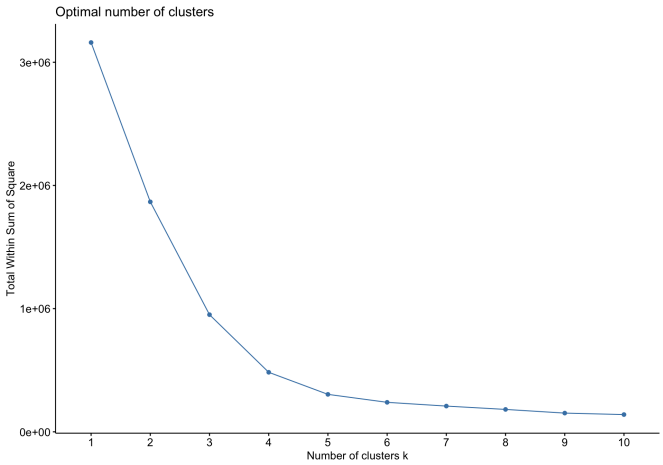


Figure 2: Within sum of squares to access optimal clustering solution

4.4 Dissimilarity matrix

The dissimilarity matrix is a symmetric matrix between all the regions. The matrix is populated by calculating the difference of the sequences between every pair of regions and is symmetrical because each row and column represents a NUTS 2 region (as it happens in any distance matrix). Each region consists of an 11-state long sequence (i.e. from 2008 to 2018). As in the previous stages there are different options to compare sequences. In this notebook I have used the simple Optimal Matching algorithm which uses the substitution costs between the quintiles and aggregates them for every pair of sequences. 'Dynamic Hamming' distance is an alternative method of Optimal Matching which calculates different substitution costs for every time period, assuming that the probabilites of transitioning between different states significantly change over time. Another variant of Optimal Matching is the 'Optimal Matching of Transition Sequences' that accounts for the sequencing of states by explicitly considering their order. In this notebook I have used the simple Optimal Matching algorithm as the aim of the notebook is to present how sequence analysis can be applied to the context of exploring youth unemployment change without making any assumptions that the timing or the ordering of states is considered more important which other Optimal Matching methods can explicitly capture. Studer, Ritschard (2016) provide a good review of different variants of Optimal Matching.

Hence, using the Optimal Matching method a dissimilarity matrix between all NUTS 2 regions has been built based on the substitution costs shown in Table 3. Lower costs mean that sequences are more similar, while larger costs mean that they are different. Hence, it is an abstract distance matrix, showing how "close" two sequences are.

[16]:
```
# Calculate the distance matrix
seq.OM <- seqdist(seq_obj, method = "OM", sm = subs_costs)
```

4.5 Classification of sequences

The final analytical step is to classify the sequences based on their similarities. There is a wide range of clustering algorithms to choose from, when it comes to object classification. Here, the Partitioning Around Medoids (PAM) clustering method was selected for classifying sequences. The PAM algorithm is similar to k-means, but is considered more robust (Kaufman, Rousseuw 1991). A dissimilarity matrix can be used as an index. The algorithm iterates to minimize the sum of dissimilarities within clusters, compared to k-means that aims to minimize the sum of squared Euclidean distances. PAM is based on finding k representative objects or medoids among the observations and then k clusters (that should be defined as in k-means) are created to assign each observation to its nearest medoid. There are different fit statistics to assess optimal clustering solutions.

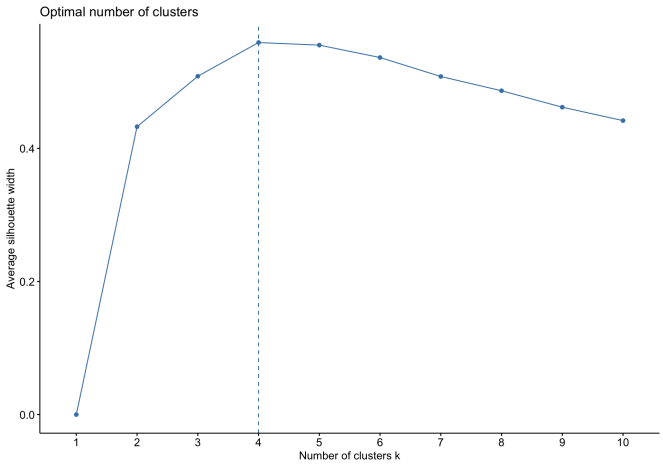


Figure 3: Average silhouette width to access optimal clustering solution

[17]:
```
# Assess different clustering solutions to specify the optimal number of clusters
fviz_nbclust(seq.OM, cluster::pam, method = "wss")
```

[17]:
```
Output in Figure 2
```

[18]:
```
# Assess different clustering solutions to specify the optimal number of clusters
fviz_nbclust(seq.OM, cluster::pam, method = "silhouette")
```

[18]:
```
Output in Figure 3
```

In this notebook I have used two fit statistics (see Figures 2 and 3) to assess various clustering solutions. The focus of this notebook is not to demonstrate the differences between fit statistics. Thus, I will not get into more detail on what every measure means. However, a useful tutorial can be found at https://rstudio-pubs-static.s3.amazonaws.-com/455393_f20bacf1329a49dab40eb393308b33eb.html. In short, they show how well separated each cluster is compared to other clusters (see Figure 2) but also how "compact" the observations are within each cluster (see Figure 3). The fit statistics here show that the optimal clustering solution is four clusters.

[19]:
```
# Run clustering algorithm with k = 4
pam.res <- pam(seq.OM, 4)
```

Having classified the sequences, it is then important to visualise the results to understand differences between the groups. Figure 4 and 5 show the four resulting transition patterns of young unemployment in NUTS 2 regions based on the quintiles they belonged from 2008 to 2018. In Figure 4 each line represents a region, each colour a quintile group and the x-axis represents each year. Figure 5 displays the year-specific distribution of each sequence group. Finally, the y-axis in Figure 4 represents the total number of sequences within each sequence group, while in Figure 5 it represents the distribution of sequences that belong to each sequence group at thus it ranges from 0 to 1.

[20]:
```
# Assign the cluster group into the tabular dataset
quant_data_wide$cluster <- pam.res$clustering
# Then rename clusters
quant_data_wide$cluster <- factor(quant_data_wide$cluster, levels=c(1, 2, 3, 4),
                                  labels=c("Stable Low youth unemployment",
                                           "Stable Moderate youth unemployment",
                                           "Increasingly High youth unemployment",
                                           "Stable High youth unemployment"))
```

For convenience and better communication of the results I assigned names to the four groups starting from the top left plot as:

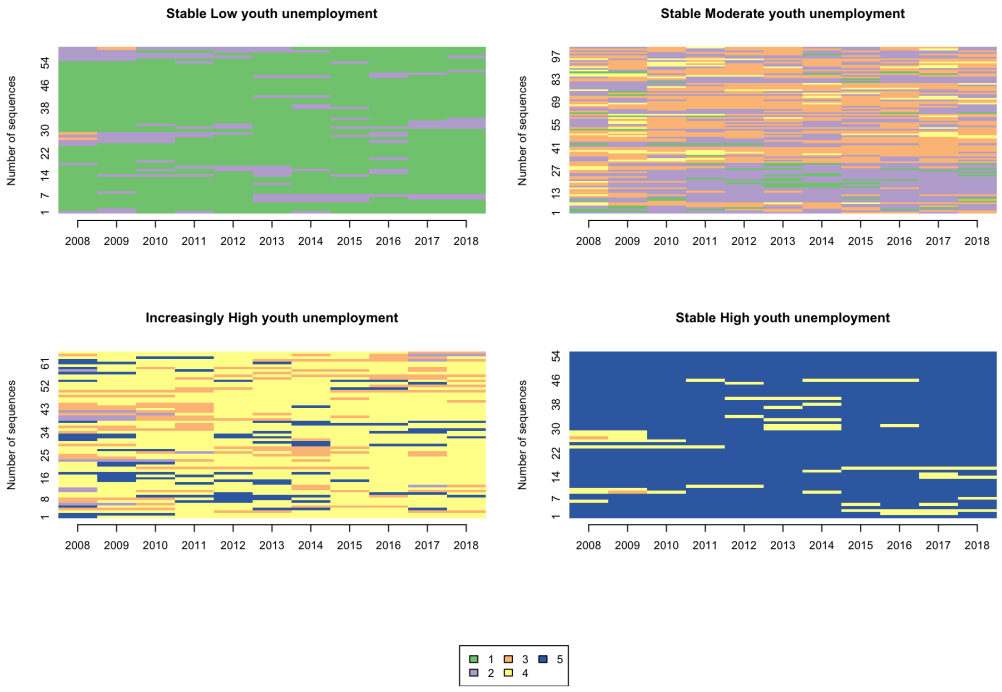


Figure 4: Individual sequences by sequence group

- Group 1 ... Stable Low youth unemployment
- Group 2 ... Stable Moderate youth unemployment
- Group 3 ... Increasingly High youth unemployment
- Group 4 ... Stable High youth unemployment

[21]:
```
# Plot of individual sequences split by sequence group
seqIplot(seq_obj, group = quant_data_wide$cluster, ylab = "Number of sequences")
```

[21]:
```
Output in Figure 4
```

[22]:
```
# Distribution plot by sequence group
seqdplot(seq_obj, group = quant_data_wide$cluster, border=NA,
ylab = "Distribution of sequences")
```

[22]:
```
Output in Figure 5
```

The results of this analysis mainly show patterns of stability in terms of youth unemployment. Group 1 contains regions that are in the lowest quintile, which means they have the lowest youth unemployment ratios over time. Group 4 is exactly the opposite of group 1, containing the regions that belong to the highest quintile over time (highest youth unemployment ratios). Group 2 consists of regions that classified either in the 2nd or 3rd quintile in the last 10 years. Finally, group 3 consists of regions that initially (i.e. 2008) belonged to 3rd, 4th, 5th and few on the 2nd quintile but gradually transformed to the 4th quintile, thus now having a higher percentage of young unemployed people.

5 Exploring spatio-temporal trends of youth unemployment in Europe

Youth unemployment as a socioeconomic phenomenon is of main concern in European policy. Thus, it is important to visualise the findings of this notebook, so that they can be

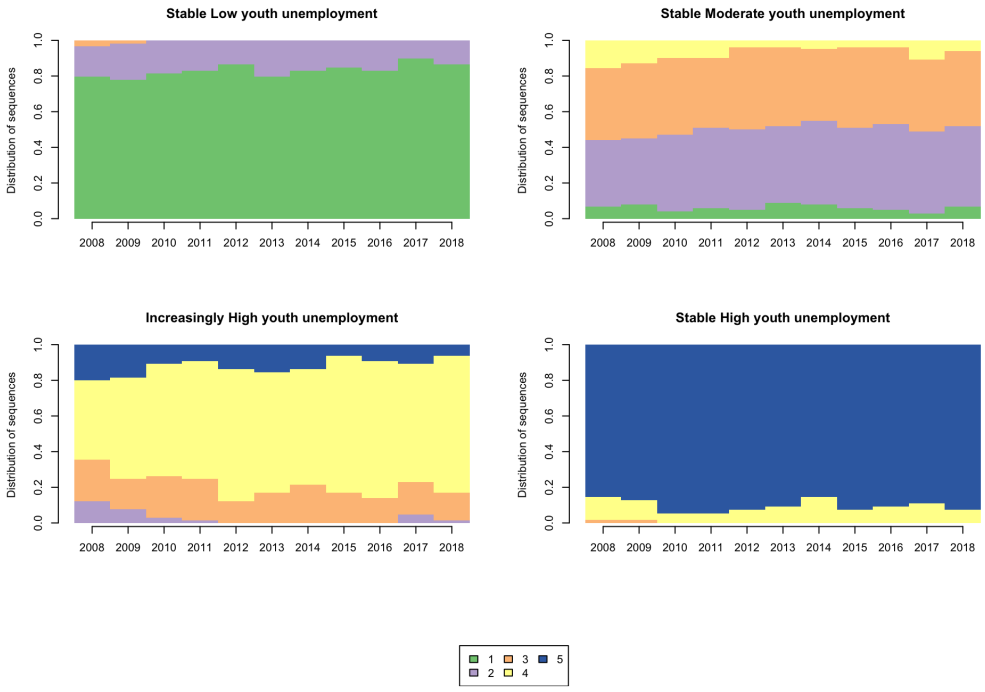


Figure 5: Distribution plot by sequence group

easily explored by the reader. To achieve this, I linked the results of the sequence analysis to the NUTS 2 region geographies so to create an interactive map. I then calculated the frequency of each trajectory group in every European country and created an interactive plot. In this way, the results are more accessible to everyone interested[1].

The map (see Figure 6) offers the opportunity to hover over the regions. Then by clicking on any region, information on the trajectory group, the region name and the country name is shown. The interactive plot (see Figure 7) offers an overview of the frequencies of trajectory groups across European countries. By hovering over the plot, one can observe the exact frequency of each group. It also offers the opportunity to zoom in on particular countries and to manually navigate through the graph (i.e. pan option on the toolbox on the right top of the plot). Finally, by clicking on the legend, particular group(s) can be selected to be shown.

[23]:
```
# Merge the spatial to the tabular dataset which includes the cluster names
map_data <- merge(geodata, quant_data_wide, by.x="FID", by.y="geo", all.x=TRUE)
```

Figures 6 and 7 show spatio-temporal variations of youth unemployment within and across European countries. As illustrated in the map, Mediterranean and Balkan countries (i.e. Greece, Italy, Spain, Turkey, Bulgaria and Romania) have stable high youth unemployment over time. On the other hand, northern countries and central European countries (i.e. Norway, Sweden, Netherlands, Germany, Austria and Switzerland) have stable low youth unemployment over time. Finally, the majority of central European countries and the United Kingdom have followed moderate levels of youth unemployment change. However, there are regional differences highlighting that socioeconomic inequalities are not only apparent between countries but also within their national boundaries. There is a clear split between the stable high youth unemployment in south Italy compared to lower but still increasingly high youth unemployment in the north. Spain has three tiers, split geographically, where the more northern the region, the lower the youth unemployment level. A similar pattern appeared in the United Kingdom, where northern regions have higher youth unemployment levels than southern regions over time.

[1]The interactive figures are included in the HTML-version of the paper or can be generated from the Rmd-file.

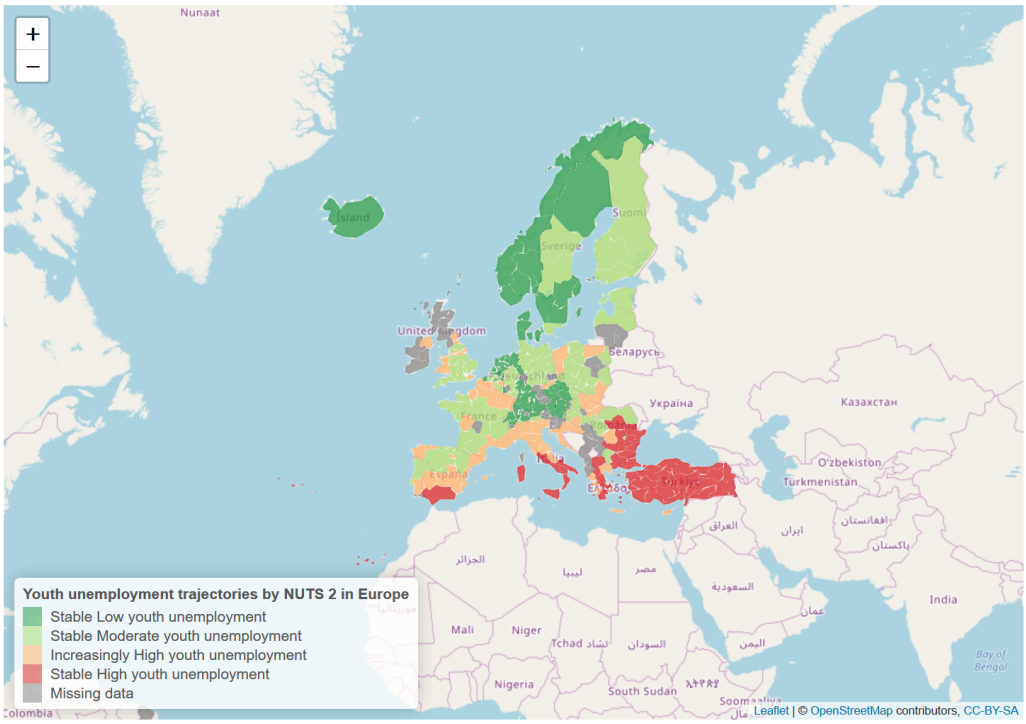


Figure 6: Interactive map of youth unemployment trajectories in NUTS 2 from 2008 to 2018

When looking in more detail at some major metropolitan regions, we can observe deviations from their neighbouring regions. Bucharest and Sofia seem to have lower unemployment levels compared to adjacent regions in Romania and Bulgaria respectively. On the other hand, while Belgium has moderate or low levels of youth unemployment on average, Brussels, its biggest city and capital, has stable higher youth unemployment. Austria follows similar pattern where the country has on average high concentration of 'stable low youth unemployment' regions but its biggest city (and capital) Vienna is classified as 'stable high youth unemployment'. This highlights that higher levels of socioeconomic inequalities and more disadvantaged groups are often aggregated in large metropolitan areas.

[24]:
```
# Create a map showing the distribution of sequence clusters
# Specify the colour palette
myColors <- rev(brewer.pal(4,"RdYlGn"))
pal <- colorFactor(myColors, domain = unique(map_data$cluster))
# Create the initial background map, zooming in Europe
colourmap <- leaflet() %>%
  addTiles() %>%
  setView(lat = 55, lng = 1, zoom = 3)
# Create the interactive map showing the sequence clusters
colourmap %>%
  addPolygons(data = map_data,
    fillColor = ~pal(cluster),
    weight = 0.2,
    opacity = 0.8,
    color = "white",
    dashArray = "3",
    fillOpacity = 0.7,
    popup = paste("Cluster: ", map_data$cluster, "<br>",
                  "NUTS 2 Name: ", map_data$NUTS_NAME, "<br>",
                  "Country Name: ", map_data$cntr_name, "<br>"),
    highlight = highlightOptions(
      weight = 5,
      color = "#666",
      dashArray = "",
```

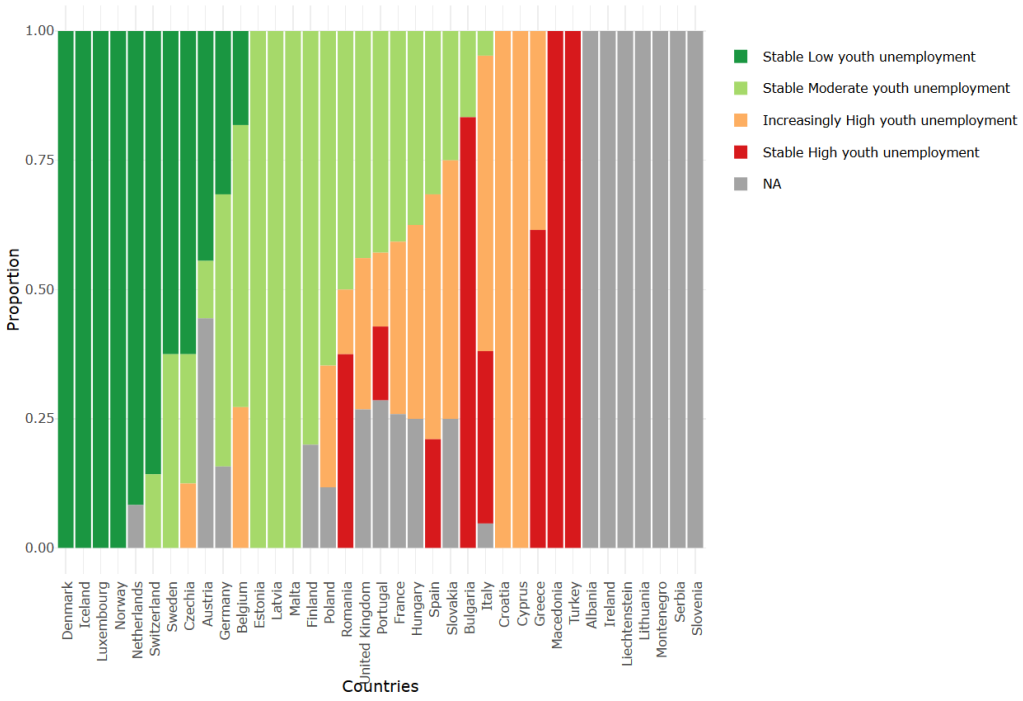


Figure 7: Distribution of youth unemployment trajectories across Europe from 2008 to 2018

```
      fillOpacity = 0.7,
      bringToFront = TRUE)) %>%
  addLegend(pal = pal,
            values  = map_data$cluster,
            na.label = "Missing data",
            position = "bottomleft",
            title = "Youth unemployment trajectories by NUTS 2 in Europe")
```

[24]:
```
Output in Figure 6
```

[25]:
```
# Calculate country summary statistics
freq_reg <- map_data@data %>%
  group_by(cntr_name, cluster) %>%
  summarise(n = n()) %>%
  mutate(freq = n / sum(n))
```

[26]:
```
# reformat the data to order them by clusters frequency
data_wide <- dcast(freq_reg, cntr_name ~ cluster, value.var="freq")
data_wide <- data_wide[order(-data_wide$`Stable Low youth unemployment`,
                             -data_wide$`Stable Moderate youth unemployment`,
                             -data_wide$`Increasingly High youth unemployment`,
                             -data_wide$`Stable High youth unemployment`),]
# Create a bar plot for country distribution of clusters
distribution_plot <- ggplot()
  geom_bar(aes(y=freq, x=cntr_name, fill=cluster), data=freq_reg, stat="identity")
  labs(title = "Distribution of youth unemployment trajectories across Europe",
       x = "Countries", y = "Proportion", fill = "")
  theme_minimal()
  theme(axis.text.x=element_text(angle = 90, hjust = 1))
  scale_x_discrete(limits=c(data_wide$cntr_name))
  scale_fill_brewer(palette="RdYlGn", na.value = "grey64", direction = -1)

# Set an interactive mode to the plot
ggplotly(distribution_plot)
```

[26]:
```
Output in Figure 7
```

6 Conclusion

Sequence analysis offers the opportunity to understand long-term socioeconomic trends over various levels of geographic regions. Clustering regions that follow similar socioeconomic trajectories can guide local, regional, national or European policy making by identifying and reducing the socioeconomic segregation of disadvantaged population groups. The first aim of this notebook was to highlight (NUTS 2) regions in Europe that maintain high, moderate or low youth unemployment levels, as well as regions that have transitioned from one level to another over the last decade. The findings of this notebook showed that northern Europe has high concentrations of regions with stable low youth unemployment, while southern Europe has high concentrations of regions with stable high youth unemployment. It is observed that southern countries struggled to adapt to the financial crisis of 2008. These findings can be used as a starting point to understand migration patterns that originated from these "disadvantaged" regions within or outside European boundaries. The second aim of this notebook was to provide a self-contained reproducible and transparent analytical workflow. This aim was achieved by providing detailed steps for successful data manipulation and make use of sequence analysis which is not a commonly used method in regional studies. Hence, I hope that data and regional scientists can benefit from the functionalities offered in the notebook and use it as a complementary guide when analysing their own data.

Acknowledgment

I would like to acknowledge the useful and constructive feedback received from my PhD supervisor Dr. Francisco Rowe throughout this research project. I would also like to thank my fellow PhD students at the University of Liverpool Krasen Samardzhiev and Patrick Ballantyne as well as the two anonymous reviewers for their useful comments on earlier versions of the notebook.

References

Bell DNF, Blanchflower DG (2011) Young people and the great recession. *Oxford Review of Economic Policy* 27[2]: 241–267. CrossRef.

Brzinsky-Fay C (2007) Lost in transition? Labour market entry sequences of school leavers in Europe. *European Sociological Review* 23[4]: 409–422. CrossRef.

Delmelle EC (2016) Mapping the DNA of urban neighborhoods: Clustering longitudinal sequences of neighborhood socioeconomic change. *Annals of the American Association of Geographers* 106[1]: 36–56. CrossRef.

Dietrich H (2012) Youth unemployment in Europe – Theoretical considerations and empirical findings. Friedrich ebert stiftung, bonn

Kaufman L, Rousseuw PJ (1991) Finding groups in data: An introduction to cluster analysis. Vol. 47. 2. CrossRef.

O'Reilly J, Eichhorst W, Gábos A, Hadjivassiliou K, Lain D, Leschke J, McGuinness S, Kureková LM, Nazio T, Ortlieb R, Russell H, Villa P (2015) Five characteristics of youth unemployment in Europe: Flexibility, education, migration, family legacies, and EU policy. *SAGE Open* 5[1]: 1–19. CrossRef.

Patias N, Rowe F, Cavazzi S (2020) A scalable analytical framework for spatio-temporal analysis of neighborhood change: A sequence analysis approach. In: Kyriakidis P, Hadjimitsis D, Skarlatos D, Mansourian A (eds), *Geospatial Technologies for Local and Regional Development*. Springer International Publishing, Cham, 223–241. CrossRef.

Peng RD (2011) Reproducible research in computational science. *Science* 334[6060]: 1226–1227. CrossRef.

Pop A, Kotzamanis B, Muller E, McGrath J, Walsh K, Peters M, Girejko R, Dietrich C (2019) YUTRENDS – Youth unemployment: Territorial trends and regional resilience. ESPON, Luxemburg

Rowe F, Casado-Díaz JM, Martínez-Bernabéu L (2017a) Functional labour market areas for Chile. *REGION* 4[3]: R7–R9. CrossRef.

Rowe F, Corcoran J, Bell M (2017b) The returns to migration and human capital accumulation pathways: Non-metropolitan youth in the school-to-work transition. *Annals of Regional Science* 59[3]: 819–845. CrossRef.

Rule A, Birmingham A, Zuniga C, Altintas I, Huang C, Knight R, Moshiri N, Nguyen MH (2019) Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLoS Computational Biology* 15[7]. CrossRef.

Salmela-Aro K, Kiuru N, Nurmi J, Eerola M (2011) Mapping pathways to adulthood among finnish university students: Sequences, patterns, variations in family- and work-related roles. *Advances in Life Course Research* 16[1]: 25–41. CrossRef.

Sandve GK, Nekrutenko A, Taylor J, Hovig E (2013) Ten simple rules for reproducible computational research. *PLoS Computational Biology* 9[10]. CrossRef.

Sanger F, Nicklen S, Coulson AR (1977) DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences* 74[12]: 5463–5467. CrossRef.

Studer M, Ritschard G (2016) What matters in differences between life trajectories: A comparative review of sequence dissimilarity measures. *Journal of the Royal Statistical Society. Series A: Statistics in Society* 179[2]: 481–511. CrossRef.

REGION
The Journal of ERSA
Powered by WU

Volume 7, Number 1, 2020, 21–34
DOI: 10.18335/region.v7i1.282

journal homepage: region.ersa.org

Teaching on Jupyter – Using notebooks to accelerate learning and curriculum development

Jonathan Reades[1]

[1] King's College London, London, United Kingdom

Received: 7 October 2019/Accepted: 5 January 2020

Abstract. The proliferation of large, complex spatial data sets presents challenges to the way that regional science and geography more widely is researched and taught. Increasingly, it is not 'just' quantitative skills that are needed, but computational ones. However, the majority of undergraduate programmes have yet to offer much more than a one off 'GIS programming' class since such courses are seen as challenging not only for students to take, but for staff to deliver. Using the evaluation criterion of minimal complexity, maximal flexibility, interactivity, utility, and maintainability, we show how the technical features of Jupyter notebooks particularly when combined with the popularity of Anaconda Python and Docker enabled us to develop and deliver a suite of three 'geocomputation' modules to Geography undergraduates, with some progressing to data science and analytics roles.

1 Introduction

The growth of data from sources that are both 'accidental, open, and everywhere' (Arribas Bel 2014), and characterised by volume, velocity, variety, and questions of veracity (Gorman 2013) has opened up new possibilities, and challenges, for researchers. This, in turn, calls for new conceptual, methodological, and technical approaches since 'acquiring data is no longer a strongly limiting factor to completing analytical tasks' (Bowlick, Wright 2018), working with it is. It is not particularly important whether these skills are framed as an informed empirical social science (Ruppert 2013) or as a computational social science (Lazer et al. 2009); authoritative reviews of the social sciences and humanities by The British Academy (2012), and of human geography by the Economic and Social Research Council (Ley et al. 2013), have concluded that many graduates are poorly prepared to engage with this world of 'big data'. The Royal Society (2019) has called for curriculum change at Higher Education Institutions (HEIs) with a view to encouraging interdisciplinarity and the effective integration of data science skills.

This presents something of a problem for a nascent 'geographic data science' (Singleton, Arribas Bel 2019) of the sort that regional science, and regional studies and geography more widely, require since a surprisingly large number of university programmes continue to teach proprietary, mostly point-and-click software. So many students' principal exposure to quantitative methods, let alone computational ones, comes in a standalone 'quantitative methods module' that provides little in the way of meaningful interaction with the underlying issues of spatial data and spatial data analysis at scale. And while the issue may be particularly acute for students in the U.K. (Johnston et al. 2014), even in more technically-oriented countries there is often not much more on offer than a

straightforward 'GIS course' (Wikle, Fagin 2014). Consequently, students progressing to higher levels of study or the professional realm often find that 'the skills least developed in undergraduate GIS courses are those related to programming and computer science' (Bowlick et al. 2017).

2 Dependencies

This notebook requires the GeoJSON labextension to be installed in JupyterLab. All other packages should be part of a default Python 3 installation.

3 Context

The long history of computers in geography has not been without controversy (Arribas Bel, Reades 2018, Barnes 2013, Cresswell 2014, Johnston et al. 2014), although many have actively engaged with recent developments (e.g. Torrens 2010) and expect impacts on the very fabric of the discipline (González Bailón 2013). So although our experience with teaching computational skills using Jupyter notebooks is clearly rooted in the 'geography of geography' (Bradbeer 1999) in the sense that we speak to particular challenges here in the U.K., it is part and parcel of a wider skills gap at the undergraduate level in general. In short, too few students are gaining the skills needed to engage with this deluge of data or to take advantage of cutting-edge tools developed outside of the field, either as researchers or as end-users in the public or private sectors (Singleton 2014).

This is where we believe that the pedagogical potential of Project Jupyter (Kluyver et al. 2016) is revolutionary: reflecting on our experience of trying to roll out exactly this type of programme, we seek to highlight the transformative potential of notebooks for student and researcher development. Jupyter removes significant barriers to teaching by providing a flexible and familiar interface that hides, or even postpones indefinitely, some of the complexity of managing local programming language installations whilst also allowing instructors to provide rich media and contextual information next to the code where it is needed the most. Making coding accessible is not simply about allowing students to 'hack away' at data, it can actually *help* students to better understand spatial analytic methods by linking concepts to code as Xiao's outstanding text on algorithms demonstrates (Xiao 2016).

3.1 Teaching Programming to Non Programmers

Given the interaction effects between pedagogical and subsequent practice, it is therefore worth placing the challenge of teaching programming in the context of the shifting terrain for quantitative research and researcher development. These challenges start early: many students already demonstrate what Spronken-Smith (2013) calls 'equation phobia': "students not linking numbers, and problems with visualisation of quantities." Hodgen et al. (2014) suggest just some of the reasons for this: limited prior knowledge and attainment; time elapsed since last study of maths; a failure to see relevance; and the wide range of attainment levels within each cohort (Hodgen et al. 2014). Whatever its origins, a general lack of confidence and/or competence creates a feedback loop fuelling further avoidance (Chapman 2010).

In the context of maths instruction Macdonald, Bailey (2000) have also noted the challenge inherent in delayed gratification given that 'maths is the tool, not the goal.' Given the apparent gulf between `print('Hello world.')` and being able to write useful analytical code, the issue is no less serious in programming. There is no reason why the familiarity of so-called 'Digital Natives' with computers should have any bearing on their understanding of how they actually work; indeed, today's students may well be *more* detached from the underlying processes – metaphorical and actual – thanks to 'the sophistication of modern Graphical User Interfaces' (Muller, Kidd 2014). In the long run, programming requires an ability to envision and manipulate abstract entities such as data structures sitting, in turn, on top of additional layers of abstraction such as the application and its state(s), the file system and its structure(s), the operating system and even the underlying hardware.

Figure 1: Barron Stone memorably demonstrates *for* and *while* loops (Stone 2013)

There are many differing views of how programming should be taught (Pears et al. 2007), though we come down firmly on the side of Lukkarinen, Sorva (2016) that there are advantages to 'contextualising programming practice in the field of application'. In general, it seems that introductory programming courses should strive simultaneously for richness and simplicity: richness in the 'constructs' associated with programming, and simplicity in terms of the foundation being laid (Lukkarinen, Sorva 2016). Unfortunately, the expertise of teachers is not always a plus for effective teaching (Chapman 2010) since concepts that seem intuitive and are easily connected to a range of related problems by the instructor may yield no such benefit to the novice. As we developed our teaching materials, we found that videos created by other learners could, at times, capture student attention more effectively than our own demonstrations; for example, Stone's instructional video for students at Rice University on the difference between for and while loops, shown in Figure 1. Using Jupyter notebooks this kind of content can be embedded directly in the task explanation.

3.2 Course Structure

The work reported here draws on methodological and pedagogical research conducted over the past five years in the Department of Geography at King's College London; it seeks both to position learning to code as essential to further student and staff development, and to examine the reasons why Jupyter notebooks have been selected as the best means of achieving this goal. As such, this research is necessarily caught up in a wider debate about quantitative skills amongst students; however, our undergraduate 'pathway' in *Geocomputation & Spatial Analysis* (which could be understood as an optional 'minor' in the North American tradition) seeks to go beyond the kinds of statistical skills training encouraged by funders (see brief discussion in Johnston et al. 2014) and to tackle these in conjunction with computational skills. We want to take students with a variety of social, economic, ethnic, and computational backgrounds and cultivate in (and with) them an appreciation of, and ability to undertake, interdisciplinary work with a strong computational element (see Mir et al. 2017, for a discussion of the *CS+X* format).

Based on our own experience, we felt that shoe horning exposure to 'computational geography' into a single module – as seems to occur in many American programmes (Bowlick et al. 2017) – would only reinforce student aversion to such approaches, so we opted to 'unpack' the concepts across three modules:

1. Geocomputation
2. Spatial Analysis and Modelling, and
3. Applied Geocomputation.

These modules must be taken in sequence, the preceding module acting as a pre-requisite for admission to the next, although students are free to exit the sequence at any time. We also provide an optional 'Code Camp' (Reades et al. 2019) to be undertaken over the summer before the first module begins so that students begin the term familiar with basic concepts: variables, lists/arrays, dictionaries/hashes, and functions/subroutines, provided they have done the work.

3.3 Contextualised Computing

To our knowledge, there is no other undergraduate programme like it with important differences in both style and substance from what would be covered in an Economics, Statistics, or Computer Science (CS) degree in terms of its spatial and applied focus. In this sense, the modules are an extended test of 'contextualised computing' instruction (see Lukkarinen, Sorva 2016, for a review) which seeks to emphasise relevance to 'real-world' applications and to avoid "general CS content, such as how one might go about sorting an array of any type for an unspecified purpose" (Lukkarinen, Sorva 2016). We also recognise, however, that "contextualized computing education cannot help students learn more in less time" (Guzdial 2010) and that the *transferrable* aspects of this learning need to be emphasised: in our case we try to highlight how the same approach can be applied to human and physical geography problems.

Consequently, wherever possible these exercises are grounded in spatial examples, even where these are very simple indeed, on the basis that connecting them to the learner's existing knowledge and interests will improve retention at the introductory level (Guzdial 2010). For example, a notebook on dictionaries (taken from Reades et al. 2019) can start with creating and querying a phone book of national emergency numbers where the student has to replace the `???` in `eNumbers = { ??? }` with functioning Python code:

[1]:
```
eNumbers = {
        'IS': 112,
        'US': 911
}
print(f"The Icelandic emergency number is {eNumbers['IS']}")
print(f"The American emergency number is {eNumbers['US']}")
```

[1]:
```
The Icelandic emergency number is 112
The American emergency number is 911
```

Students then progress towards a task involving a dictionary-of-dictionaries:

[2]:
```
cityData = {
        'London': {
                'population': 8673713,
                'location': [51.507222, -0.1275],
                'country': 'UK'
        },
        'Paris': {
                'population': 2140526,
                'location': [48.8567, 2.3508],
                'country': 'FR'
        }
}

for city, data in cityData.items():
  print(f"The population of {city} ({data['location'][0]:0.3f}ºN,
   {data['location'][1]:0.3f}ºE) is {data['population']:,}")
```

[2]:
```
The population of London (51.507ºN, 0.128ºE) is 8,673,713
The population of Paris (48.857ºN, 2.351ºE) is 2,140,526
```

This work is building towards a GeoJSON example in which they have to complete missing attributes in order to show a marker centred on the university's central London campus. Since GeoJSON is essentially a dictionary-of-dictionaries, this is a good test of their understanding, but with Jupyter they receive immediate feedback on this because GeoJSON can be embedded directly into the notebook: an interactive web map shows up

as soon as they've run the code, reinforcing the contextual aspect – that this is *all* about geography – of their learning.

[3]:
```
# King's College London's coordinates...
# What format are they in? Does it seem appropriate?
# How would you convert them back to numbers if you
# needed to do so?
longitude = '-0.11596798896789551'
latitude  = '51.51130657591914'

# Notice how we set up a data type and location
# here where it's easy to see where the lat/long
# values are being used we could also use these
# in a loop as a _template_ for creating many points
# from a data file! Notice too that it's a dictionary
# containing a mix of string and list values...
the_geometry = {
        "type": "Point",
        "coordinates": [longitude, latitude],
}

# Now we set up the larger 'data file' this is harder
# to read but is *still* basically a dictionary! A
# 'collection' implies more than one feature, and in this
# case the list of 'features' is nothing more than a list
# of dictionaries so that our data stays in order!
the_position = {
        "type": "FeatureCollection",
        "features": [
        {
                "type": "Feature",
                "properties": {
                        "marker-color": "\#7e7e7e",
                        "marker-size": "medium",
                        "marker-symbol": "building",
                        "name": "KCL"
                },
                "geometry": the_geometry
        }
        ]
}

# And show the points on an interactive map!
# You don't need to know what's happening here *yet*, but
# see if you can make sense of the main elements...
try:
from IPython.display import GeoJSON
from IPython.display import display
import json
parsed = json.loads(str(the_position).replace("\'", "\""))
display(GeoJSON(parsed))
except ImportError:
print("You seem to be missing either the GeoJSON extension or json library.")
```

[3]:
```
The output is shown in Figure 2
```

4 How We Reached Jupyter

Since the pathway pushes students both conceptually and technically, finding ways to take the deployment and management of the software stack out of the picture has been a priority. Our review of the pedagogical literature and practical experience gained in the private and HEI sectors—including several failures during the first few years of teaching—led us to the ultimate conclusion that a useful geospatial programming environment should possess the following characteristics:

Minimal Complexity : it does not require students to load and learn a new Operating System or large number of new applications/platforms at the same time as they are

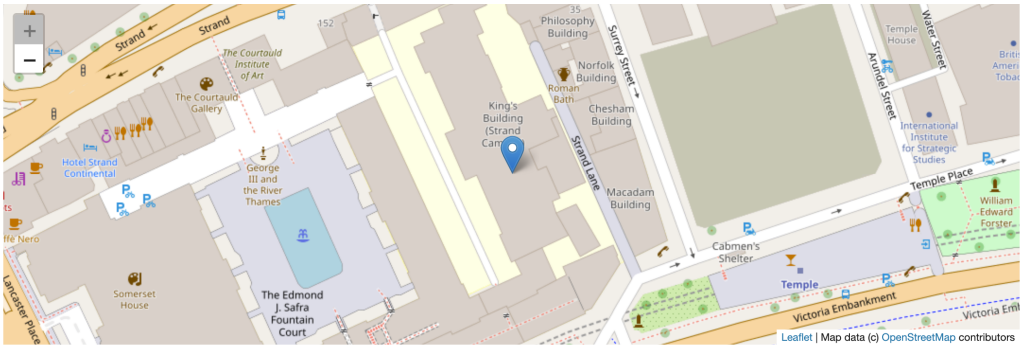


Figure 2: Output of code 3

learning to code; it should also be reasonably 'performant' on a mix of student and HEI hardware.

Maximal Flexibility : it is simple, if not always easy, to configure and install on a range of hardware, but is not 'sandboxed' or 'packaged' in ways that constrain our freedom to install what we need to teach effectively.

Interactivity : it allows us to keep commentary, 'rich' media, and other scaffolding material together with the code so that students can move between code and explanations easily, and can add their own annotations as needed.

Utility : it supports life-long learning by providing a 'real world' development environment that would be both familiar, and accessible, to students after graduation in personal and professional contexts.

Maintainability : it can be easily updated by the instructor(s) and supports version control and easy distribution mechanisms.

These five features can, at times, appear to cut against each other: maximal flexibility and minimal complexity are difficult to reconcile since the former tends to expose more 'options' to the user, while the latter seeks to mask those same options. However, a strong advantage of Jupyter is that it meets all of these criteria to some extent, and in most cases meets them fully!

4.1 Pretty Walled Gardens

The desired set of features ruled out commonly-used proprietary platforms: at the time we began developing the curriculum, MATLAB was still a *de facto* standard for many but its pricing and sandboxing approach made it both less flexible and less useful for students once they graduated and lost access to the HEI license. Like Etherington (2016), we were therefore attracted by the fact that Python presented 'no financial or hardware obstacles to teaching' and that, consequently, "students [would] always be able to use their Python programming skills..." Etherington (2016). However, in developing the early iterations of the course we also, again like Etherington (2016), encountered significant challenges in 'getting a working installation of Python together with its associated geospatial packages'.

We discovered that the existing, IT-supported Enthought Canopy Python distribution provided few of geospatial libraries, and that updating it with packages from outside of their 'walled garden' caused all manner of issues. This situation was not entirely unexpected since geospatial analysis is not a key component of Enthought's offering to universities; however, the challenges of keeping up with the state-of-the-art are such that additional barriers to software update management are undesirable. Indeed, the pace of change in the field can be gauged from Wise's review of 'geospatial technologies' in U.K. universities (Wise 2018): it not only questions the utility of 'free' programmes (presumably meaning Free Open Source Software, or FOSS) which now dominate in the data sciences and in many research projects, but it also contains not a single mention of programming—in Python or any other language.

4.2 The Wrong Kind of Flexibility

Like Muller, Kidd (2014), who sought to 'debug geographers' with an introduction to a holistic computing context alongside programming skills *tout court*, we next attempted to provide our students with virtualised Linux desktop systems in the belief that this would empower them not only with a better understanding of what was going on 'under the hood' but also with a computer on which they could experiment without fear of damaging their existing installation. For good measure, we included other useful analytics tools such as the latest version of QGIS with all of the 'bindings' for low-level packages such as GDAL (the Geospatial Data Abstraction Layer).

Using VMWare and Ubuntu 16 LTS with a full Python installation configured largely 'by hand' provided us with a fully FOSS 'solution' that students could take with them and update in the future as they gained confidence in using such software. However, we soon found that in-memory and on-disk bottlenecks, together with students' tendency to actually try to install Ubuntu's suggested updates and render their systems inoperable, made this a profoundly alienating and frustrating experience. For students already working hard to master the basics of programming, having to 'drop' into the Terminal in order to resolve installation errors when they were used to seamless updates on their host operating systems simply represented an unnecessary hassle that detracted from the real focus of the modules: learning to use code to perform spatial analyses.

4.3 Escape Velocity

While we had been tinkering with different Linux and Python distributions, a set of three connected developments had been transforming the landscape for teaching:

1. A few academics who had taken very different approaches began, rather bravely, to publish their teaching methods and materials freely for others to use (e.g. Arribas Bel 2019);

2. Data scientists not only adopted Python *en masse*, driving the rapid development of new analytical and visualisation libraries (*e.g.* pandas, seaborn, bokeh), but they had also quickly settled on the use of a then-novel technology called 'iPython notebooks' to widely share their tutorials online;

3. Since many of these data scientists were paid by firms interested in moving their work into production systems as smoothly and quickly as possible, this also led to improvements in the way that Python distributions and notebooks were managed.

Rather unexpectedly, the kinds of practical problems that data scientists were trying to solve mirrored quite closely the kinds of challenges that we, as teachers, were trying to solve in terms of being able to replicate installations across multiple systems and share code/commentary quickly and easily.

The iPython platform ultimately gained the ability to run other programming languages and was rebranded 'Project Jupyter', but this means that it has become a viable, general purpose teaching platform. So although the term 'Virtual Learning Environment' (VLE) is typically understood to refer to a full-featured, client-server system such as Moodle or Blackboard (see Britain 1999), it could also apply to Jupyter: not only does it have a client/server architecture (with the web-based interface allowing the server to run locally or on a remote system with no discernible difference to the student), but it has been progressively enriched with tools for grading and other common teaching tasks. Although we are not yet making full use of these new features, it is clear that Jupyter is well on its way to becoming an important teaching platform.

5 Discussion

Perhaps the single greatest benefit of working with Jupyter notebooks is that development is *not* being driven by educational needs: this is a full-featured development environment used day-in and day-out by professional software developers and large firms such as Netflix

(Ufford et al. 2018). So, unlike both expensive proprietary systems that are rarely used by small or innovative firms, and instructional systems whose functionality is limited to teaching purposes, students are able to seamlessly progress from learning to code, to competent coders, and on to practicing data scientists (as a few of our students have done), using a single environment. This is a platform with the capacity to grow with the student, following them out of the 'ivory tower' and into gainful employment.

An additional benefit flowing from the professional use of Jupyter is that many researchers, not least the others included in this special issue, use notebooks as a normal part of their research practice; this allows lecturers to remain abreast of technical developments on the platform without 'updating my installation' being a separate overhead in a congested working week. This pattern of usage is in sharp contrast to tools – such as SPSS or ArcGIS – that are less-used by active researchers but often still taught in standalone modules, with the quality and timeliness of teaching materials often suffering accordingly. Jupyter breaches the historical divide between computational research and teaching, not only allowing students to benefit from active research, but also for research to build on student outputs (see, for example Reades et al. 2019).

5.1 Cloning Around

Jupyter becomes particularly powerful when combined with other recent developments in the management and distribution of computing platforms. Anaconda Python's enhanced support for the configuration of virtual environments (in essence, multiple distributions of Python on the same system) allows specific versions of Python and sets of required libraries to be specified in a simple text file following the 'Yet Another Markup Language' (YAML) standard. The code below downloads and prints out part of the YAML file that we use to configure both student machines *and* our Docker container (about which more below); here the virtual environment is named `gsa2019`:

[4]:
```
import urllib

url  = 'https://raw.githubusercontent.com/kingsgeocomp/gsa_env/gsa2019/gsa.yml'
with urllib.request.urlopen(url) as resp:
file = resp.read().decode('utf8').split('\n')

# Don't output everything...
to_print = list(range(0,5))  list(range(39,48))  list(range(110,116))

print("=" * 50)
for line in to_print:
print(file[line])
print("=" * 50)
```

[4]:
```
==================================================
# OVERVIEW
# This YAML script will attempt to install a Python virtual environment able to
# support the requirements of all three of King's College London's 'Geocomputation'
# pathway in the BA/BSc Geography programme.
#
# CONFIGURATION PARAMETERS
name: gsa2019
channels:
 - conda-forge
 - defaults
dependencies:
 - python=3.7
 - pip
 - git
 - xlrd
 - xlsxwriter
 - pip:
   - six
#- git\+http://github.com/sevamoo/SOMPY#egg=sompy # Doesn't run in Python3
   - git\+http://github.com/kingsgeocomp/SOMPY#egg=sompy
==================================================
```

The use of YAML configuration files makes it easier to install a teaching instance of

Python and to expose this as a named 'iPython kernel'. The connection between virtual environments and kernels allows researchers to manage multiple research and teaching installations of Python on the *same* system, to access them through the same Jupyter interface, and to do so without changes to one Python installation impacting any others.

5.2 Docking Safely

The emergence of containerisation platforms such as Docker now makes it much simpler to distribute a pre-configured virtual machine[1] (such as a pre-packaged teaching or research environment) that will run on almost any host operating system: Mac, Windows, or Linux. Because the virtual machines are fully specified at the time of creation, students can download and install a working version with one command, while instructors can be confident that every student is working with the *same* version of every library. This year we provided students with a Docker image that leveraged the work of Arribas Bel (2019) but that had been customised to provide only the features that we wished to teach.

The combined popularity of Python and Docker has led to the creation of novel, web-based platforms such as Binder (mybinder.org); these take notebooks stored on the GitHub code-sharing web site to build a Docker image serving those notebooks on Binder's servers. Students may now learn to code without installing any software whatsoever. Local installation can be deferred to the point at which specialist requirements or load on the server require it. In a stroke, one of the most pernicious barriers to entry, needless technical issues associated with installation and configuration of programming software, has been eliminated.

5.3 Houston, We Have a Problem

Of course, no single solution is without drawbacks and Jupyter is no exception. It is worth noting that there are quite specific technical, conceptual, and development issues raised by Jupyter that are difficult to circumvent without both know-how and some careful thinking about assessment and teaching. The principal technical challenge relates to user permissions on managed machines (*e.g.* in computer clusters) since Python, Jupyter, and Docker all struggle to different degrees with 'locked down' Windows systems. Indeed, Docker does not currently run at all without administrator privileges. We worked closely with university-level IT staff to install and provision Anaconda Python and Jupyter. Provision of the YAML configuration script assisted with both installation and isolation of our teaching environment from their existing installation, easing institutional barriers to adoption.

From a teaching standpoint, an additional issue is that Git – the dominant version control software that we use to manage and share notebook changes – sees notebooks in a way that means just re-running code registers as a local modification of the file that needs to be committed to the version control system. So although 'GitHub' provides support for the online display of Jupyter notebooks, the use of Git can lead to a large number of essentially meaningless commits. This can make tracking meaningful content changes over time more difficult, and it means that we've shied away from teaching students about version control on the basis that they may not perceive the value of commits that seem to record little of value.

A final and rather unexpected disbenefit was uncovered the year after we moved from the Spyder IDE to Jupyter: weaker student understanding of execution flow. Unlike a traditional script that clearly executes from top-to-bottom (typically in its entirety), Jupyter notebooks freely intermingle code blocks and text/rich media blocks allowing – and even encouraging – the user both to jump between widely separated blocks without executing intervening code and to edit and re-run earlier blocks. This leads to: a) difficult-to-diagnose bugs because the code *looks* like it should execute properly but doesn't, and b) to a weaker student understanding of system 'state' in terms of instantiated variables, loaded libraries, and available functions. We typically seek to cultivate this understanding by stressing that the *real* test, whether directly assessed or not, of whether their code

[1] It should be noted that, technically, Docker containers are not virtual machines in the traditional sense.

Table 1: Evaluating Jupyter

	Pros	Cons
Minimal Complexity	Deploying a full geographic data science 'stack' requires installing one application (Docker or Anaconda Python) and running two lines of code in a Terminal/Shell to install and configure Jupyter, its dependencies, and the analytical libraries. Environment requires no configuration.	Persistent challenges with student understanding of file system interaction and paths. Some confusion around multiple Python instances manifesting as different 'kernels' in notebooks.
Maximal Flexiblity	Combination of Binder, Docker, and Anaconda Python allows us to install on nearly any hardware/operating system mix. Docker uses same YAML configuration script as Anaconda Python so maintaining compatibility and consistency is straightforward.	Students cannot update Docker containers and do not gain understanding of package management or dependency conflict resolution.
Interactivity	Students can view/edit/add rich media, code, and other content directly within the Jupyter notebook environment. Textual and graphical outputs from code cells in notebooks are saved between restarts of Jupyter.	Students do not develop a strong understanding of execution flow and system state.
Utility	Growth of Jupyter has made it the 'tool of choice' for data scientists, and students are able to continue working with a fully functioning development environment. Students can edit installation and configuration scripts incrementally, as expertise grows.	Relative ease of installation may not prepare students for managing their own development and production environments. Students remain unfamiliar with IDEs and code-completion.
Maintainability	Docker and Anaconda update mechanisms are straightforward. GitHub works well for distribution, previewing, and (to a lesser extent) version control.	Nature of notebooks makes it harder for instructors to track incremental changes in version control, and for students to see value of such an approach.

'works' is that a notebook can be run in full (`Restart Kernel and Run All Cells`) without user intervention.

We should note that, in the absence of an Integrated Development Environment (IDE), students are unlikely to benefit from test suites and other tools that support developer best-practice. However, such an approach can also have the effect of deterring new students by pushing back the point at which they appear to be achieving anything concrete: "Because learning in computer science and programming is challenged by numerous barriers, students need to be motivated about the purpose, value, and utility of concepts within course work" (Bowlick et al. 2017) So while knowledge of professional tools and practices is desirable, we nonetheless feel that these kinds of ideas and issues are best tackled when students have progressed further with their studies and are motivated to tackle more abstract challenges.

6 Conclusion: Back Here on Earth

In order to understand why the practical benefits of teaching with Jupyter notebooks outweigh the technical and conceptual challenges encountered, it is worth returning to the evaluation criteria outlined near the start of this work. Table 1 summarises the pros and cons observed across the five dimensions identified by our review of the state-of-the-art nearly six years ago.

From this, the principal technical recommendation is that a flexible mix of platforms should be used to deliver Jupyter-based learning. We recommend Binder to deliver foundational material using few non-core Python libraries, and now strongly recommend that students use Docker in subsequent modules. However, a critical issue is that Windows 10 Home Edition does not support Docker, and it is therefore *still* necessary to support direct installation of Anaconda Python and associated configuration of the 'kernel' using a

YAML text file. We are also investigating the use of a containerised JupyterHub running on our own hardware: this would allow students to mimic using Binder while benefiting from the ability to save work and make full use of Python's capabilities. All of the code supporting these configurations is available as a Github repository, as is Arribas-Bel's resource.

6.1 And Back to the Future

A failure to engage directly with computational approaches and tools poses long-term risks: while ours 'has always been a following discipline' (Burton 1963), what is new is that other disciplines have now taken an interest in cities and regions (O'Sullivan, Manson 2015). Ruppert (2013) warns, "if social scientists do not step forward, then computational social science risks becoming the exclusive domain of ... computing scientists" (Ruppert (2013). However, there is also an enormous opportunity for students equipped with both domain knowledge and programming skills to act as 'knowledge brokers' (Bowlick, Wright 2018). As Mir et al. (2017) note: "truly transformative work at the intersection of computing and ... other disciplines requires ... people with heterogeneous skill-sets (both computational and non-computational) who, despite their differences in training, can work collaboratively." In other words, facing the future requires both translators and explorers: individuals who understand the broader terrains across which knowledge moves and the frontiers at which new knowledge is generated.

We have also come to believe that the use of Jupyter-like platforms in non-STEM disciplines may have a role to play in addressing a deeper problem: the widening participation challenge in computationally-oriented disciplines such as data science (The Royal Society 2019). A particular contribution is these other disciplines' capacity to provide an applied context for computational training that helps to motivate further study and engagement (see Bort et al. 2015, for a creative application in literary studies). It should not be the responsibility of Geography and allied fields to plug the so-called 'leaky pipeline' (Berryman 1983), but they may yet create novel pathways for a more diverse cohort of students to enter computationally intensive fields. Such an outcome would not only be to the benefit of Computer Science, it would very much be to the benefit of an innovative Regional Science as well.

Acknowledgements

This work builds on the input of many – staff and students – to the Geocomputation and Spatial Analysis pathway at King's College London; however, I wish to particularly acknowledge the critical contributions of Dr. James Millington, Michele Ferretti, Dr. Chen Zhong, and Dr. Yijing Li. Finally, Dr. Arribas-Bel has donated many hours of his time – directly and by example – to helping me to develop and migrate our teaching environment.

References

Arribas Bel D (2014) Accidental, open and everywhere: Emerging data sources for the understanding of cities. *Applied Geography* 49: 45–53. CrossRef.

Arribas Bel D (2019) A course on geographic data science. *The Journal of Open Source Education* 2: 42. CrossRef.

Arribas Bel D, Reades J (2018) Geography and computers: Past, present, and future. *Geography Compass* e12403

Barnes T (2013) Big data, little history. *Dialogues in Human Geography* 3: 297–302

Berryman S (1983) Who will do science? Trends, and their causes in minority and female representation among holders of advanced degrees in science and mathematics. A special report. Rockefeller Foundation, New York, NY

Bort H, Czarnik M, Brylow D (2015) Introducing computing concepts to non majors: A case study in gothic novels. Proceedings of the 46th ACM Technical Symposium on Computer Science Education, 132–137. ACM. CrossRef.

Bowlick F, Goldberg D, Bednarz S (2017) Computer science and programming courses in geography departments in the United States. *The Professional Geographer* 69: 138–150. CrossRef.

Bowlick F, Wright D (2018) Digital data centric geography: Implications for geography's frontier. *The Professional Geographer* 70: 687–694. CrossRef.

Bradbeer J (1999) Barriers to interdisciplinarity: Disciplinary discourses and student learning. *Journal of Geography in Higher Education* 23: 381–396. CrossRef.

Britain S (1999) A framework for pedagogical evaluation of virtual learning environments. Report, Joint Information Systems Committee. https://www.webarchive.org.uk/wayback/archive/20140613220103/http://www.jisc.ac.uk/media/documents/programmes/jtap/jtap-041.pdf

Burton I (1963) The quantitative revolution and theoretical geography. *The Canadian Geographer/Le Géographe Canadien* 7: 151–162. CrossRef.

Chapman L (2010) Dealing with maths anxiety: How do you teach mathematics in a geography department? *Journal of Geography in Higher Education* 34: 205–213. CrossRef.

Cresswell T (2014) Déjà vu all over again: Spatial science, quantitative revolutions and the culture of numbers. *Dialogues in Human Geography* 4: 54–58

Etherington T (2016) Teaching introductory GIS programming to geographers using an open source python approach. *Journal of Geography in Higher Education* 40: 117–130. CrossRef.

González Bailón S (2013) Big data and the fabric of human geography. *Dialogues in Human Geography* 3: 292–296. CrossRef.

Gorman S (2013) The danger of a big data episteme and the need to evolve geographic information systems. *Dialogues in Human Geography* 3: 285–291. CrossRef.

Guzdial M (2010) Does contextualized computing education help? *ACM Inroads* 1: 4–6. CrossRef.

Hodgen J, McAlinden M, Tomei A (2014) Mathematical transitions: A report on the mathematical and statistical needs of students undertaking undergraduate studies in various disciplines. Report, The Higher Education Academy

Johnston R, Harris R, Jones K, Manley D, Sabel C, Wang W (2014) Mutual misunderstanding and avoidance, misrepresentations and disciplinary politics: Spatial science and quantitative analysis in (United Kingdom) geographical curricula. *Dialogues in Human Geography* 4: 3–25. CrossRef.

Kluyver T, Ragan Kelley B, Pérez F, Granger B, Bussonnier M, Frederic J, Kelley K, Hamrick J, Grout J, Corlay S, Ivanov P, Avila D, Abdalla S, Willing C, Jupyter Development Team (2016) Jupyter notebooks – a publishing format for reproducible computational workflows. In: Loizides F, Schmidt B (eds), *Positioning and power in academic publishing: Players, agents and agendas*. IOS Press, 97–90

Lazer D, Pentland A, Adamic L, Aral S, Barabási A, Brewer D, Christakis N, Contractor N, Fowler J, Gutmann M, Jebara T, King G, Macy M, Roy D, Van Alstyne M (2009) Life in the network: The coming age of computational social science. *Science* 323: 721–723. CrossRef.

Ley D, Braun B, Domosh M, Elliott S, Le Heron R, Peake L, Willekens F, Yeoh B (2013) International benchmarking review of UK human geography. Report, Economic and Social Research Council, in partnership with the Royal Geographical Society (with IBG) and the Art and Humanities Research Council. https://esrc.ukri.org/files/research/research-and-impact-evaluation/international-benchmarking-review-of-uk-human-geography/

Lukkarinen A, Sorva J (2016) Classifying the tools of contextualized programming education and forms of media computation. Proceedings of the 16th Koli Calling International Conference on Computing Education Research, 51–60. ACM. CrossRef.

Macdonald R, Bailey C (2000) Integrating the teaching of quantitative skills across the geology curriculum in a department. *Journal of Geoscience Education* 48: 482–486. CrossRef.

Mir D, Mishra S, Ruvolo P, Pollock L, Engen S (2017) How do faculty partner while teaching interdisciplinary CS+X courses: Models and experiences. *Journal of Computing Sciences in Colleges* 32: 24–33

Muller C, Kidd C (2014) Debugging geographers: Teaching programming to non computer scientists. *Journal of Geography in Higher Education* 38: 175–192. CrossRef.

O'Sullivan D, Manson S (2015) Do physicists have geography envy? and what can geographers learn from it? *Annals of the Association of American Geographers* 105: 704–722. CrossRef.

Pears A, Seidman S, Malmi L, Mannila L, Adams E, Bennedsen J, Devlin M, Paterson J (2007) A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin* 39: 204–223. CrossRef.

Reades J, De Souza J, Hubbard P (2019) Understanding urban gentrification through machine learning. *Urban Studies* 56: 922–942. CrossRef.

Reades J, Ferretti M, Millington J (2019) Code camp: 2019. Github repository, King's College London

Ruppert E (2013) Rethinking empirical social sciences. *Dialogues in Human Geography* 3: 268–273. CrossRef.

Singleton A (2014) Learning to code. *Geographical Magazine* 77

Singleton A, Arribas Bel D (2019) Geographic data science. *Geographical Analysis*: 1–15. CrossRef.

Spronken-Smith R (2013) Toward securing a future for geography graduates. *Journal of Geography in Higher Education* 37: 315–326. CrossRef.

Stone B (2013) Differences between for & while loops (in Python). Video, YouTube. https://www.youtube.com/watch?v=9AJ0uoxtdCQ

The British Academy (2012) Society counts. Report, The British Academy, https://www.-thebritishacademy.ac.uk/sites/default/files/BA Position Statement - Society Counts.pdf

The Royal Society (2019) Dynamics of data science skills: How can all sectors benefit from data science talent? Report, The Royal Society, https://royalsociety.org/-/media/policy/projects/dynamics of data science/dynamics of data science skills report.pdf

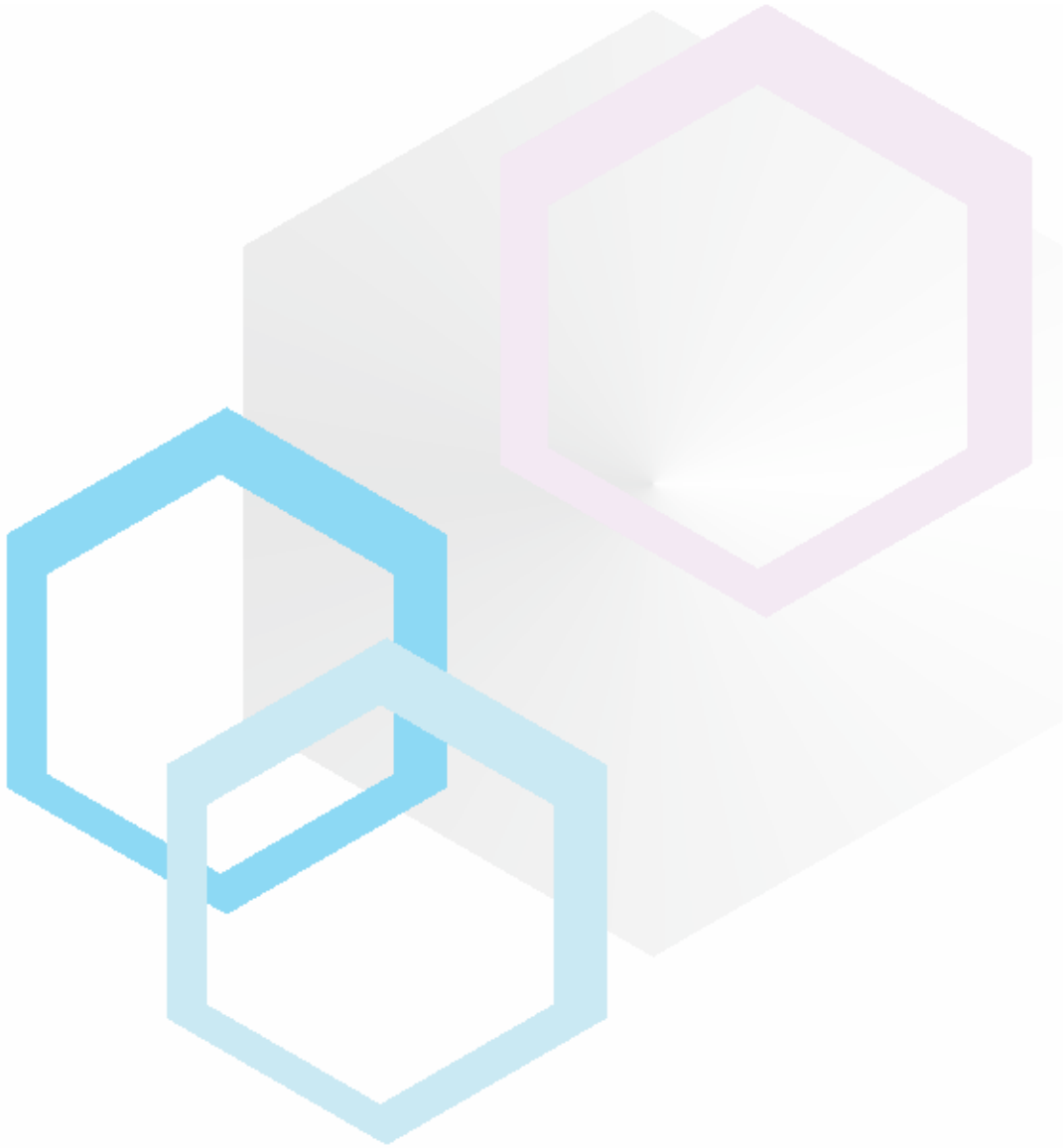
Torrens P (2010) Geography and computational social science. *GeoJournal* 75: 133–148. CrossRef.

Ufford M, Pacer M, Seal M, Kelley K (2018) Beyond interactive: Notebook innovation at Netflix. Blog post, Netflix, https://netflixtechblog.com/notebook-innovation-591ee3221233. [last checked: 3 October 2019]

Wikle T, Fagin T (2014) GIS course planning: A comparison of syllabi at US college and universities. *Transactions in GIS* 18: 574–585. CrossRef.

Wise N (2018) Assessing the use of geospatial technologies in higher education teaching. *European Journal of Geography* 9

Xiao N (2016) *GIS Algorithms: Theory and Applications for Geographic Information Science & Technology*. Research Methods. SAGE. CrossRef.



Funded by

ersa

WU

FWF