Bannert, Matthias

Working Paper

Gateveys: An R toolbox for re-traceable aggregation of business tendency survey data

KOF Working Papers, No. 326

Provided in Cooperation with:
KOF Swiss Economic Institute, ETH Zurich

Suggested Citation: Bannert, Matthias (2013) : Gateveys: An R toolbox for re-traceable aggregation of business tendency survey data, KOF Working Papers, No. 326, ETH Zurich, KOF Swiss Economic Institute, Zurich,
https://doi.org/10.3929/ethz-a-007606143

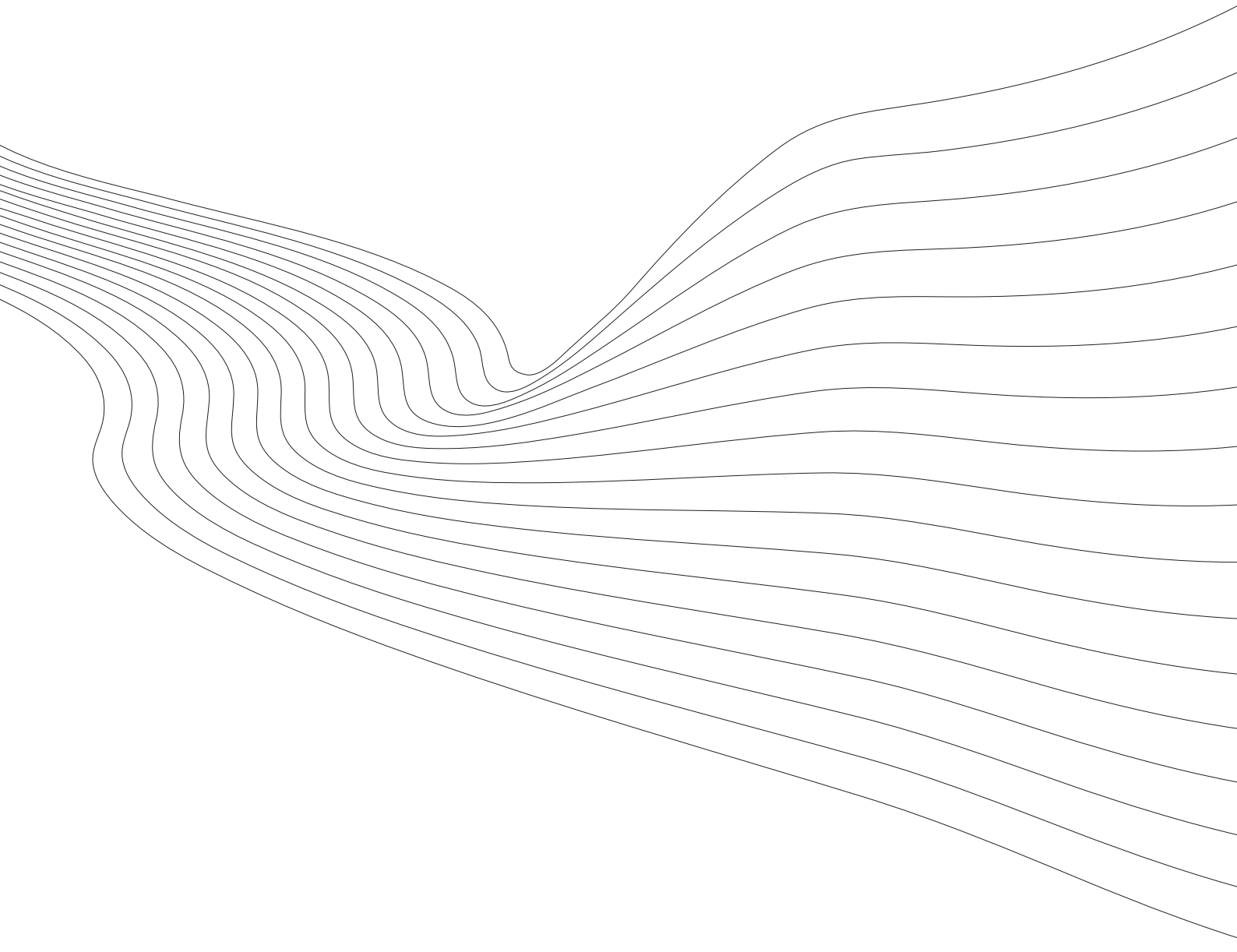
This Version is available at:
https://hdl.handle.net/10419/80843

KOF

KOF Working Papers

Gateveys – An R Toolbox for Re-traceable Aggregation of Business Tendency Survey Data

Matthias Bannert



No. 326
January 2013

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

KOF

ETH Zurich
KOF Swiss Economic Institute
WEH D 4
Weinbergstrasse 35
8092 Zurich
Switzerland

Phone +41 44 632 42 39
Fax +41 44 632 12 18
www.kof.ethz.ch
kof@kof.ethz.ch

gateveys - An R Toolbox for Re-traceable Aggregation of Business Tendency Survey Data

Matthias Bannert

January 28, 2013

Abstract

This paper describes how to use the R package `gateveys` to establish a transparent and reproducible aggregation work flow for longitudinal data stemming from business tendency surveys (BTS). Business tendency survey researchers are addressed in particular though the suggested work flow could also be applied to other processes that generate categorical data. The package has two main features: First, it provides functions to build an aggregation process that re-calculates all periods when a new survey wave is added and hence can be fully reproduced at any later stage. Second, the package can be used to dynamically add localized meta information to the resulting time series object during the aggregation process. Besides, the paper suggests a software architecture for use of the package in a scenario with regular, periodical survey waves.

Keywords: R, aggregation, survey data, qualitative survey, categorical data, business tendency surveys, time series, meta data, meta information, software architecture, reproducible research.

1 Introduction

The term "Reproducible Research" has been establishing itself in a growing number of empirical research communities. Koenker and Zeileis (2009) elaborate the term relying on Buckheit and Donoho (1995) who claim that an article built on computational science is rather advertisement of a finding than a finding itself and that true results are explained in the software and the instructions to it. Later on de Leeuw (2001) calls this idea the Claerbout's Principle dating it back to Standford geophysics emeritus Jon Claerbout. Though it is hard to pinpoint exactly who started the movement in the respective fields of research we can easily see the growing attention. Numerous websites like `reproducibleresearch.net` or an own CRAN Task View[1] explicitly dedicated to the topic as well numerous journal articles show the significance of reproducible research to modern empirical science.

Nevertheless the enthusiasm seems to spill over inertly to survey based economic research so far. Surveys themselves continue to be an important source of timely information in empirical economics though. Especially periodical business tendency surveys (BTS) have established themselves worldwide as a reliable provider of longitudinal economic data. Even though their results are published at regular intervals and survey designs are similar across different sectors in most countries, implementations of their analysis and reporting are surprisingly different. This heterogeneity is striking in particular because business tendency survey results are not only published in professional journals but in recurring descriptive reports with a circulation and frequency beyond academic literature.

The following sections of this paper describe how to make use of the `gateveys` R package (Bannert, 2012) to setup a standard workflow for the aggregation of participant level results. Weighted aggregation of participant level information to different aggregation levels is a typical basis for BTS reports.

The following chapter describes the idiosyncratic situation that propelled the motivation to build an R toolbox for business tendency survey researchers. The third chapter explains how to use this toolbox to create a random example dataset that makes this paper reproducible itself. The fourth chapter describes the aggregation process of BTS data before the fifth chapter explains how to store and reference the results. Finally the paper concludes with a future outlook for the package and a suggestion for a software environment into which the package should be embedded to make the most of it.

[1] CRAN Task Views (CTV) are a group of maintained websites that monitor the activity of R developers for several fields of research. CTVs give an overview of package related to a particular field and try to summarize the work of the R community in a field. Max Kuhn manages the CTV on reproducible research, which can be found at: http://cran.r-project.org/web/views/ReproducibleResearch.html.

2 Motivation

The pioneering package `Sweave` (Leisch, 2002) and its more recent alternative `knitr` (Xie, 2012) brought comprehensive general frameworks for reproducible research to the R statistical environment. Due to their abstract and general nature these package are obviously not intended to cover specific problems in single fields of research by implementing specific models or classes.

This is exactly were the motivation for the `gateveys` package stems from. Despite all similarities between BTS, their long tradition and step-by-step rise to more and more sectors and countries has hampered a more co-ordinated and standardized implementation of their reporting. At the same time the strong demand for the exchange of statistical data puts pressure on survey conductors to find a solution between respecting participants' privacy and fulfilling the needs of the research community.

The first issue is rather straightforward to resolve. Outdated concepts like disproportionate saving of hard disk storage or computing time have not been re-engineered at a similar pace to technological change. The focus on optimization has often lead to implementations built on multiple intertwined general purpose languages like $C++$ or Java combined with SQL and statisical packages like SAS. This paper suggests an alternative aggregation process widely implemented in one single language, namely R, putting emphasis on reproducibility by trying to make the implementation transparent and accessible for a relatively broad audience within an economic research institute. Though this shift to a higher level programming language comes at the cost of performance it accounts for recent advances in computing and educational background of the researchers.

The second issue is not covered by an overhaul in the sense of a clean-up. A requirement of true reproducibility is the availability of the initial dataset to the person who reproduces the original report. While data can be fully exchanged among researchers within an institute is not possible most of the time to share participant level data with external researchers or even policy makers. In the case of BTS true anonymization on the micro level is difficult because some participants are exposed by their sheer size or sector combined with regional information. Thus BTS data is typically exchanged at different aggregation levels. Although true reproducibility is obviously not possible without the initital dataset a weaker analogue needs to be implemented: the ability to track data back to its provider and find context information on the data generating process. Thus the second main feature of `gateveys` is to generically add meta information to aggregated data and built a *re-traceable* basis. This data with meta information can be used to create analyses and reports with `knitr`.

3 The Data – a Realistic Random Example

The following sections use a reproducible example to illustrate how the `gateveys` R package can facilitate working with surveys. We use the `generateSamplePanel` function to create a random dataset that mimmics data stemming from a longi-

tudinal BTS. Though many properties of this sample dataset may hold for other categorical surveys to, this paper focuses on the case of BTS. Before we create the actual dataset we use the function `drawFromMixed` to draw a random vector of weights from a mixed distribution. BTS often use the number of employees to weight participants with a proxy of their relative importance to the economy. Hence we use a mix of distributions to draw a somewhat realistic weighting vector from three different distributions[2].

R code 3.1.

```
require(gateveys)
# draw weights from mixed distributions
w <- drawFromMixed(1000, list(rnorm = list(mean = 1000,
    sd = 250), rexp = list(), rchisq = list(df = 10)),
    c(0.1, 0.5, 0.4))
w <- ceiling(unlist(w))
```

Subsequently the dataset is created by handing the number of observations (n), the number of questions (q), the frequency, starting period, end and weight[3] to the `generateSamplePanel` function.

R code 3.2.

```
# example dataset,add some random NAs, add group
# information
sData <- generateSamplePanel(80, 3, "q", c(1995, 1),
    c(2001, 3), weight = w)
sData[, grep("quest", names(sData))] <- lapply(sData[,
    grep("quest", names(sData))], as.factor)
```

After the dataset is generated we randomly replace answers with item non-response (in R i.e. NA). Further we add some size classes based on the previously generated weights as well as some random groups.

R code 3.3.

```
sData <- generateRandomNAs(sData, c(1:3, 7), 30, 1)
sData <- merge(sData, generateRandomGroups(3, unique(sData$uid)),
    by = "uid")
altGroup <- generateRandomGroups(5, unique(sData$uid),
```

[2]For further information on the `drawFromMixed` function please have a look at the R context help of the package

[3]In practice the influence of employment based weights is often capped at a certain threshold to limit the influence of very large companies[4]. However, we do not employ such a limit in this example.

```
    variables = letters)
names(altGroup)[1] <- "altGroup"
sData <- merge(sData, altGroup, by = "uid")
# add size Class and show first 3 lines try this
# with slight modification to setSizeClass which
# returns sizeClass as factor for better summary
sData <- setSizeClass(sData, list(M = 50, L = 500),
    sectorColumn = "group", sizeColumn = "weight")
```

The following output shows a couple of observations stemming from our randomly generated dataset.

R output 3.1.

```
##   uid year period weight question_1 question_2
## 1   1 1995      1     12          2          3
## 2   1 1999      1     12          1          1
## 3   1 1998      3     12          1          3
##   question_3 group altGroup sizeClass
## 1          2     B        e         S
## 2          1     B        e         S
## 3          2     B        e         S
```

Finally we end up with a dataset that could very well stem from a business tendency survey. Our dataset contains 2160 observations ranging from the first quarter of 1995 to the third quarter of 2001. Table 1 shows some summary statistics of this sample dataset.

Table 1: Summary Statistics Random Dataset

weight	question_1	question_2	question_3	group	altGroup
Min. : 1	1 : 559	1 : 555	1 : 471	A:567	a:351
1st Qu.: 1	2 :1043	2 :1026	2 :1098	B:864	b:378
Median : 4	3 : 549	3 : 568	3 : 581	C:729	c:567
Mean : 156	NA's: 9	NA's: 11	NA's: 10		d:486
3rd Qu.: 13					e:378
Max. :1351					

4 Aggregate Data with gateveys R package

Just like the random sample generated above, most BTS result sets consist of several columns each of which contain categorical data. In our example the categorical items of an answer are mutually exclusive and can be observed over time on a monthly or quarterly basis. The analysis of BTS data typically

requires the weighted aggregation of company level data by different groups. Such groups can be periods, size classes, sector classifications, arbitrary classes based on categorical questions (e.g. export share categories) or a combination of multiple of these variables.

Thus a flexible interface is required to set groups. The `gateveys` R package makes use of the key based indexing of the `data.table`[5] package (Dowle, Short, and Lianoglou, 2012). The main workhorse function `calcShares` automatically aggregates over all variables defined in the key of a `data.table`. In addition to that the user may specify additional variables which will be added to the key in incremental fashion. The following example shows how to group over multiple categorical variables very much similar to GROUP BY used in SQL.

R code 4.1.

```
# convert data into data.frame using keys
sData.dt <- data.table(sData, key = c("year", "period",
    "group"))
# question 1 is used as an additional key
shares <- calcShares(sData.dt, "question_1", "weight")
```

The user can manually submit additional keys to the function the `calcShares` to be combined with `lapply` and `do.call`. This can be a very flexible and powerful combination as it allows for a fixed set of keys plus an additional, varying key.

R code 4.2.

```
# define vector of questions
questions <- grep("question", names(sData), value = TRUE)
#
l <- lapply(questions, function(X) do.call(calcShares,
    list(sData.dt, X, "weight")))
names(l) <- questions
```

The output below shows a result `data.frame` object for one question. The overall result of the previously described process is a list of multiple tables.

R output 4.1.

```
##      year period group question_1 sumTest
##   1: 1995      1     A          3    1320
##   2: 1995      1     A          2    1114
```

[5]Except for calls of `calcShares` the distinction between `data.frame` and `data.table` does not matter in this paper. Hence we stick to `data.frame` notation.

```
##   3: 1995      1     A          1    1279
##   4: 1995      1     B          2    4134
##   5: 1995      1     B          1      26
##  ---
## 239: 2001      3     B          1    3332
## 240: 2001      3     B          3      20
## 241: 2001      3     C          1    2364
## 242: 2001      3     C          3    1277
## 243: 2001      3     C          2      25
##      sumTestTotal AN    share
##   1:         3713  4 0.355508
##   2:         3713 12 0.300027
##   3:         3713  5 0.344465
##   4:         5088 18 0.812500
##   5:         5088  2 0.005110
##  ---
## 239:         5088  9 0.654874
## 240:         5088  6 0.003931
## 241:         3666 10 0.644845
## 242:         3666  9 0.348336
## 243:         3666  8 0.006819
```

In practice surveys often need to be aggregated over multiple sets of key variables such as different regional levels or different sector classification schemes. The `gateveys` R package offers a function to apply `calcShares` based aggregation over all categorical variables and multiple classification columns in a dataset. This function is called `weighByMultiClasses` and lets users conveniently loop over multiple classifications levels and variables instead of calling `calcShares` multiple times. If the `multicore` package (Urbanek, 2011) is available `weighByMultiClasses` automatically makes use of the parallelized version of `lapply` called `mclapply`. On a multicore machine this could speed up the process significantly[6]. Assume an alternative classification column called '`altcolumn`' is introduced to the dataset. Code snippet 4.3 shows how to use `weighByMultiClasses`.

R code 4.3.

```
listOfResults <- weighByMultiClasses(sData.dt, c("group",
    "altGroup"), questions, "weight")
names(listOfResults) <- c("group", "altGroup")
```

Because R functions can only return one single object, `weighByMultiClasses` returns a single nested list. This result list already contains all relevant information though it is not arranged in a typical time series format.

[6]This feature has only been tested on OS X, but should work similarily well on any Unix machine.

R output 4.2.

```
## List of 2
##  group   :List of 3
##   ..$ question_1:Classes 'data.table' and 'data.frame': 243 obs. of ..
##   ..$ question_2:Classes 'data.table' and 'data.frame': 242 obs. of ..
##   ..$ question_3:Classes 'data.table' and 'data.frame': 243 obs. of ..
##  altGroup:List of 3
##   ..$ question_1:Classes 'data.table' and 'data.frame': 1027 obs. of..
##   ..$ question_2:Classes 'data.table' and 'data.frame': 1025 obs. of..
##   ..$ question_3:Classes 'data.table' and 'data.frame': 1002 obs. of..
```

At this point the aggregation process has finished and no further computation is needed. The next step is to organize the results in proper time series format.

5 Referencing Time Series

The main difference between the obtained list and a standard time series format is the fact that each question's respective `data.frame` contains all answer items of a particular question while time series typically just observe one item over time.

Hence we need to split the existing `data.frame` into multiple `data.frames` - one per item and question. First we use `burstList` to account for the two alternative aggregation levels (i.e. 'group', 'altGroup') in our example. In a second step we use `burstList` again to finally obtain a list of data.frames at item level.

R code 5.1.

```
dframes <- burstList(listOfResults)
names(dframes) <- c("group", "altGroup")

resultsByItem <- lapply(dframes, function(X) {
    nms <- names(X)
    res <- burstList(X)
    names(res) <- nms
    return(res)
})
```

Still, the object returned is a nested list which contains classification groups at first level (i.e. `group` or `altGroup`), questions (three questions) at the second level, selected groups (either 3 or 5 realizations depending on the classification level) at the third level. The fourth level contains a `data.frame` at item level over time. Put differently, the deepest level of nesting stores one `data.frame` per item (see also output 5.1).

R output 5.1.

```
## altGroup : List of 3
##  $ question_1:List of 5
##   ..$ a:List of 3
##   ..$ b:List of 3
##   ..$ c:List of 3
##   ..$ d:List of 3
##   ..$ e:List of 3
##  $ question_2:List of 5
##   ..$ a:List of 3
##   ..$ b:List of 3
##   ..$ c:List of 3
##   ..$ d:List of 3
##   ..$ e:List of 3
##  $ question_3:List of 5
##   ..$ a:List of 3
##   ..$ b:List of 3
##   ..$ c:List of 3
##   ..$ d:List of 3
##   ..$ e:List of 3
## group : List of 3
##  $ question_1:List of 3
##   ..$ A:List of 3
##   ..$ B:List of 3
##   ..$ C:List of 3
##  $ question_2:List of 3
##   ..$ A:List of 3
##   ..$ B:List of 3
##   ..$ C:List of 3
##  $ question_3:List of 3
##   ..$ A:List of 3
##   ..$ B:List of 3
##   ..$ C:List of 3
```

Though this arrangement of the data is very close to what we need, it's not a native R time series format in the sense that we could apply time series methods like seasonal decomposition or autocorrelation plots to it.

Before we can turn all data.frames into time series objects and effectively add proper referencing to it we need to do some preliminary work: the nested list needs to be flattened. Because the list is nested multiple times we use a recursive function to linearize it. The function `linearizeNestedList`[7] is applied as described by the following code chunk.

R code 5.2.

```
listOfSeries <- linearizeNestedList(nList = resultsByItem,
    linearizeDataFrames = FALSE, nameSep = ".")
```

[7]This function was originally written by Akhil Behl and only slighty modified by the author of this package. For further information, please see the documentation of this function in the `gateveys` package.

By running `linearizeList` all nested elements are stored in a new un-nested list which displays the information contained in the former nesting in the names of its items. In other words: `linearizeList` generically creates unique names for all list elements (containing exactly one data.frame) of the new linear list. The newly created linear list enables us to use `lapply` to handle the list in a way that is very common in R.

Organization of time series

Though the obtained list of time series information has been named generically names should be adjusted to a more meaningful and standardized pattern. Thus we use `lapply` to rename the whole list in a manner inspired by the Statistical Data and Metadata Exchange (SDMX) format[8]. The new names contain country, data provider and source (survey) information. The code chunk below shows how this is done for a sample dataset. The new generic name of the `data.frame` is also added to the respective table as an attribute, so it can be accessed by attribute operations.

R code 5.3.

```
# rename the whole list
names(listOfSeries) <- checkKeys(names(listOfSeries),
    "CH", "KOF", "TESTSURVEY", listOfSeries)
# add a unique time series key to all elements of
# the list
for.setattr(listOfSeries)
```

A simple example shows why this kind of organization is crucial. Assume a single survey question has four possible answer items (positive, neutral, negative, not specified). Aggregated results typically show the share of participants that chose a particular item. Further assume that data is aggregated on six different levels each of which contains 10 classes. Imagine a raw version and a seasonally adjusted version is needed and the whole questionnaire has 10 questions. In this scenario we end up with 4′800 time series which all need to be referenced by a sound unique name. Though aggregation obviously reduces the number of observations, the number of variables can increase dramatically.

Once proper referencing is added to `data.frames` within the list, we can turn all `data.frames` into an R time series format by using `dataframe2ts`. This step also assigns all time series objects contained in `listOfSeries` to the global environment[9].

[8] see also http://sdmx.org/. The naming pattern suggested in this paper is simple and has not the intention to cover the complexity of SDMX by any means. Though SDMX has been influential when developing the time series organization proposed in this paper. Future version of the `gateveys` package might also go further into the direction of SDMX.

[9] Assignment to the global environment is the default though any other existing environment can be chosen using the env argument. Using other environments us helpful in particular of a large number of time series is assigned.

R code 5.4.

```
lapply(listOfSeries, dataframe2ts, period = "period",
    frequency = "Q")
```

Note that the `gateveys` package currently supports only `zoo` (Zeileis and Grothendieck, 2005) as a time series format and therefore depends on `zoo`[10].

5.1 Adding Meta Information

Referencing is clearly the most fundamental step in the organization of data. Still though supplementing it with meta information can become as important - in particular when data is shared with external researchers. Among others Dippo and Sundgren (2000) emphasize the importance of meta data in statistics. In programming, object names are limited in the information they can hold, because they need to make the ridgewalk between handling and information. External researchers might need a catalogue to understand a foreign referencing system and therefore would benefit from the ability to browse data based on meta information. But meta information cannot only be used to spot the right dataset in an haystack of data, it can also be used to describe data in generic reports or visualization.

5.1.1 Fixed Meta Data Information

The `gateveys` package can add two kinds of meta information to a time series. First so-called 'fixed' meta information can be added to a time series. The term 'fixed' is used as opposed to 'localized' meta information. Fixed meta information is information that does not need to be translated, e.g. because it is numeric. The `gateveys` package implements fixed meta data in object oriented manner, using R's S4 classes (R Core Team, 2012). The function `addFixedMetaData` is used to attach an attribute of S4 class `metaFixed` to the time series. Table 2 describes the slots of this class.

5.1.2 Localized Meta Data Information

Besides the type of meta information described previously meta data can contain information that needs to be translated. In business tendency survey context this can be the case when a country has more than one official language[11] or a dataset is used across borders. Again an S4 class is used to represent this type of meta information. Table 3 describes the slots of a localized class. Note that

[10] Future releases of the `gateveys`plan to support R's standard `ts` format too. For further information see the outlook section of this paper or the project website on github.com

[11] This package was developed in Switzerland where German, French and Italian are spoken. E.g. Canada could face similar issues with French and English.

Table 2: Fixed Meta Information

Slot Name	Description
tskey	time series key, same as object name
frequency	frequency information, Quarterly, annually and monthly is currently supported
status	a possibilty to add a status for internal use
unit	unit of observation
unitMultiplier	multiplier of observation, e.g. 1000 CHF
generatedOn	date of generation of the series
generatedBy	system user who generated this series
questionType	type of the question the series is based on, qualitative, quantitative and checkbox are allowed types
relatedSeries	series that are related to the series displayed. Relation is determined by tskey chunks.

time series object may have[12] a single fixed meta information class but multiple localized classes if the information is translated to multiple languages.

Table 3: Localized Meta Information

Slot Name	Description
title	descriptive title
selectedItem	categorical item of interest
description	elaborate description of the item
aLevel	aggregation Level, very typical for business tendency surveys
selectedGroup	group that was selected on the aggregation level chosen.
survey	indicates the survey from which the data stems
itemLevels	vector that contains all possible items.
weightingInformation	information about weights, typically either unweighted or employment based weighting

5.1.3 Visualization example

As stated previously, meta information can be very helpful to generically create legends for reports or graphs. The `gateveys` package provides a covenient high level plotting function called `plotTs` that accepts an arbitrary number of time series objects. The function uses meta information to label the time series object. Table 4 shows some foreign example labels and question wording. Such description tables could be created outside R by translators and read in as .csv files, Excel spreadsheets or be queried directly from a database. A table translating the questions can be passed to the function `addLocalizedMetaData` using the `qDict` argument.

[12]There is no strict technical restriction that prevents users from setting multiple fixed meta information classes, but within the concept presented in this paper it would not be meaningful.

Table 4: Examplary Variable Mapping with Localized Questions Wording

	qkey	questionShortLabel	questionWording
1	question_1	Urteil	Wie beurteilen Sie die Situation
2	question_2	Veränderung	Wie hat sich die Situation verändert?
3	question_3	Erwartung	Welche Siutation erwarten Sie?

Question wording is handled separetedly because the conduction of multi-mode surveys is often built on very modular architecture and can involve multiple third party tools with different translation handling. Table 4 completes the meta description of the example case presented in this paper. R Code 5.5 shows how to add meta information of both types including the question wording set as an attribute to the previously generated time series objects.

R code 5.5.

```
lapply(ls(pattern = "CH.KOF"), addFixedMetaData)
# lapply localized meta data to all series
lapply(ls(pattern = "CH.KOF"), addLocalizedMetaData)
```

The function `plotTs` creates a time series chart containing all time series passed to the function. Moreover `plotTs` exploits the meta information stored in the attributes of the series object to label the plots. A `ggplot2` (Wickham, 2009) object is created which can be extended by other layers just like any other ggplot2 object (see also Figure 1).

R code 5.6.

```
graph <- plotTs(CH.KOF.TESTSURVEY.group.question_3.C.item_3,
    CH.KOF.TESTSURVEY.group.question_3.C.item_2)
```

6 Example Software Architecture and Future Outlook

Though the example described in the previous chapters could also be applied to a single survey wave or a short panel dataset it is clearly designed to deal with the challenges of longer survey panels of moderate frequency. Typical properties of business tendency surveys like their re-ocurring nature, the decade long duration or changing official standards have been explicitly accounted for in the architecture of the package. Further `gateveys` accounts for the collaborative work flow that is common in research institutes. Thus the package has been

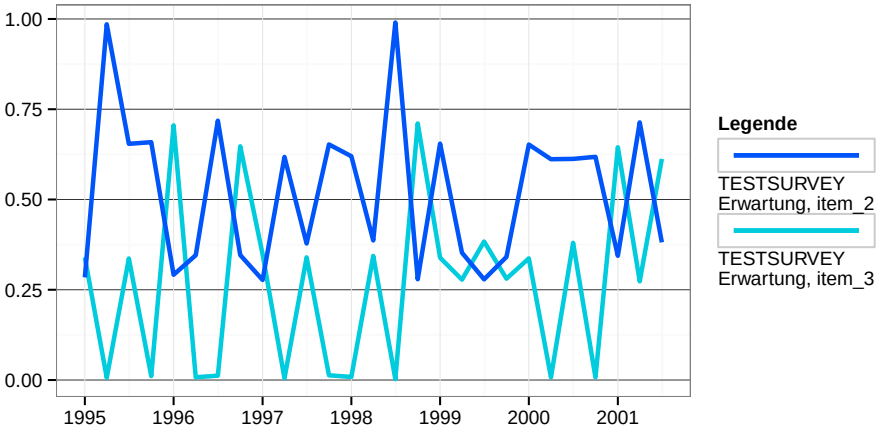
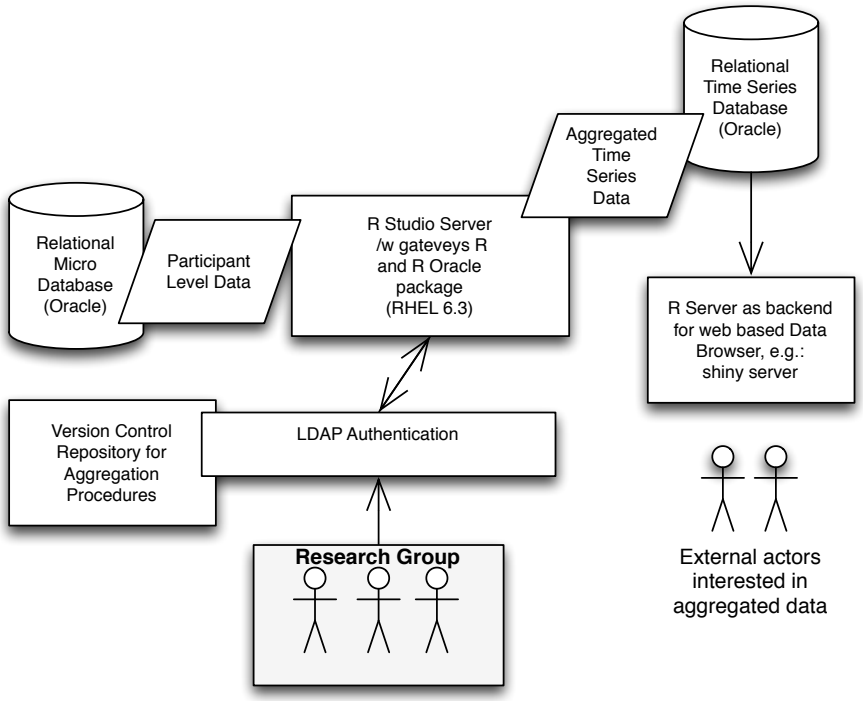


Figure 1: Example Plot with Localized German Labels based on meta information

tested in a collaborative environment in which multiple researchers were working on different periodical surveys at the same time. Figure 2 shows a software architecture in which a packaged aggregation process comes to its own – particularly when compared to single scripts of varying languages running locally. As a central piece R Studio Server provides a web based interface that enables

Figure 2: Software architecture with a server-side `gateveys` installation



researchers to access and trigger aggregation procedures installed on the server. The fact that `gateveys` has been bundled to a package helps users to install the whole process locally and fiddle with the aggregation procedures on their own desktop PC while ensuring similar conditions as in production. This is helpful for newcomers in particular. A version controlled code repository stores the code of the aggregation procedures and tracks changes. The central aggregation server gets its code from this repository.

One of the most enticing combinations in this architecture is the direct connection of R Studio server to a relational micro database. This can be easily achieved with the `ROracle` (Mukhin, James, and Luciani, 2012) package which builds on Oracle's Oracle Call Interface (OCI) respectively the open source library `OCIlib` (Rogier, Online Resource last visted in 2013). With the help of these packages users can read participant level data conveniently using either a set of convenience functions or minimal SQL[13].

If `gateveys` is used to aggregate participant level data, sound meta information is assigned in the process. By tranforming the aggregated data as described in chapter 5 of this paper it is prepared to be stored in a database driven archive. However, these time series results could be stored in a relational database as well. Mapping these results including fixed and localized metadata to a database structure is probably the most promising future perspective for the `gateveys` package.

Gilbert (2012) has build a base package called `TSdbi` plus a group of packages around it that provide a common interface to time series resources. This package family includes – amongst others – packages such as TSPostreSQL, TSMySQL, TSOracle or TSfame. The SQL based variants within this group suggest a database structure which is able to map time series information to a relational database system. The mapping written in R has to respect the structure of the database. Though the structure proposed by Gilbert (2012) also accounts for meta information one of its shortcomings is the lacking ability to map localized meta information. Thus it would be an interesting to extend the database structure suggested by the `TSdbi` family with a possibility to store localized meta information. Apart from this structural difference[14], data generated by the `gateveys` could be mapped to a relational database using `TSdbi`.

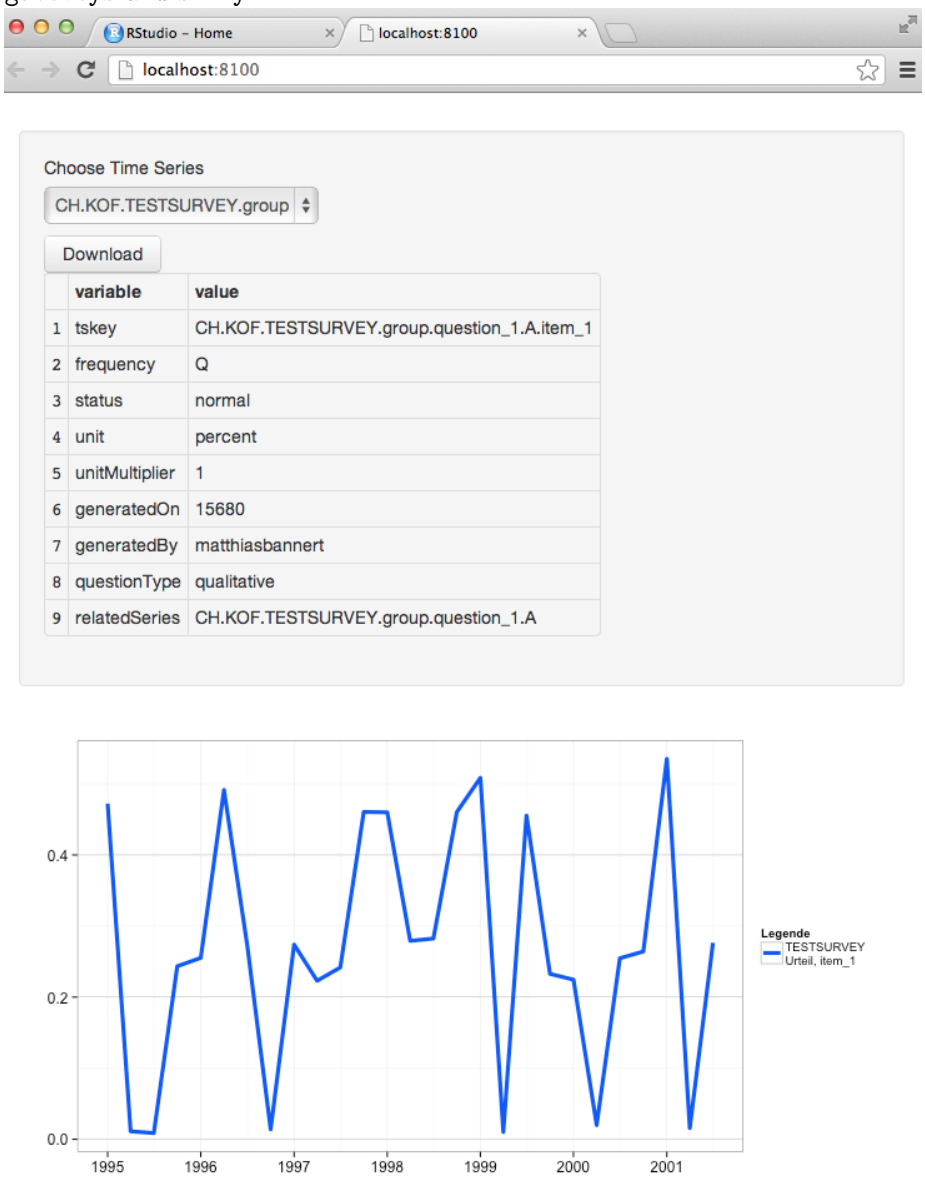
This separated time series archive could be combined with a web interface as shown in figure 2. RStudio Inc. (2012) has developed a websocket based package to deploy web interfaces using R. This package called `shiny` was only available as local standalone and beta version at time of writing this paper but a open server version has already been announced for the nearby future. Figure 3 shows a screenshot of an initial browser based data browser built with `shiny`.

The example web interface shows a dropdown menu to select a time series, a chart panel to display the selected series and a table containing fixed meta information. Moreover users can download the selected series as a comma separated

[13]This architecture works well with other SQL database management systems such as the more lightweighted MySQL or PostgreSQL.

[14]Currently `gateveys` depends on `zoo`, this dependency should also be dissolved in the future. Future users should be able to choose between `zoo` and `ts (base)` format.

Figure 3: Web based data browser with data and meta information based on `gateveys` and `shiny`



file. With the help of such a webinterface the server based construction does not only give R users access to the results but opens up these results to a broader audience. Moreover a web browser based interface could be used to let selected non-R users access the micro database without installing a database client.

References

Bannert, M. (2012): "gateveys: An R Toolbox for Business Tendency Survey Researchers," https://github.com/mbannert/gateveys/, R package version 0.1.

Buckheit, J. B., and D. L. Donoho (1995): "WaveLab and Reproducible Research," *In Antoniadis A, Oppenheim G (eds.) Wavelets in Statistics, Lecture Notes in Statistics,*.

de Leeuw, J. (2001): "Reproducible research: The bottom line. Technical Report 2001031101," Discussion paper, Department of Statistics Papers, University of California, Los Angeles.

Dippo, C. S., and B. Sundgren (2000): "The Role of Metadata in Statistics," paper presented at the International Conference on Establishment Surveys II, Buffalo, New York.

Dowle, M., T. Short, and S. Lianoglou (2012): *data.table: Extension of data.frame for fast indexing, fast ordered joins, fast assignment, fast grouping and list columns.*R package version 1.8.2.

Gilbert, P. (2012): *TSdbi: TSdbi (Time Series Database Interface),* R package version 2012.8-1.

Koenker, R., and A. Zeileis (2009): "On reproducible econometric research," *Journal of Applied Econometrics*, 24(5), 833–847.

Leisch, F. (2002): *Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis*. Physica Verlag, Heidelberg, ISBN 3-7908-1517-9.

Mukhin, D., D. A. James, and J. Luciani (2012): *ROracle: OCI based Oracle database interface for R,* R package version 1.1-7.

R Core Team (2012): *R: A Language and Environment for Statistical Computing*R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

Rogier, V. (Online Resource last visted in 2013): "OCIlib - C Driver for Oracle," http://orclib.sourceforge.net/.

RStudio Inc. (2012): *shiny: Web Application Framework for R,* R package version 0.2.3.

Urbanek, S. (2011): *multicore: Parallel processing of R code on machines with multiple cores or CPUs,* R package version 0.1-7.

Wickham, H. (2009): *ggplot2: elegant graphics for data analysis*. Springer New York.

Xie, Y. (2012): *knitr: A general-purpose package for dynamic report generation in R*R package version 0.8.

Zeileis, A., and G. Grothendieck (2005): "zoo: S3 Infrastructure for Regular and Irregular Time Series," *Journal of Statistical Software*, 14(6), 1–27.