Fickel, Norman

Working Paper

Messung supplementärer und partikularer Einflüsse mittels Sequenzialregression: Excel-Makropaket SeqReg

Diskussionspapier, No. 34/2000

Provided in Cooperation with:
Friedrich-Alexander-University Erlangen-Nuremberg, Chair of Statistics and Econometrics

Suggested Citation: Fickel, Norman (2000) : Messung supplementärer und partikularer Einflüsse mittels Sequenzialregression: Excel-Makropaket SeqReg, Diskussionspapier, No. 34/2000, Friedrich-Alexander-Universität Erlangen-Nürnburg, Lehrstuhl für Statistik und Ökonometrie, Nürnberg

This Version is available at:
https://hdl.handle.net/10419/29590

Messung supplementärer und partikularer Einflüsse mittels Sequenzialregression: Excel-Makropaket „SeqReg“

Norman Fickel[1]

Friedrich-Alexander-Universität Erlangen-Nürnberg
Wirtschafts- und Sozialwissenschaftliche Fakultät
Volkswirtschaftliches Institut
Lange Gasse 20
90403 Nürnberg
Deutschland

Zusammenfassung

Das neuartige Verfahren der Sequenzialregression ermöglich die Messung von supplementären und partikularen Einflüssen. Zur praktischen Anwendung werden durch das vorliegende Diskussionspapier dreizehn Tabellenfunktionen für Microsoft Excel zur Verfügung gestellt. Sie sind in fünf Gruppen eingeteilt:

1. *Auswahl von Reihenfolgen:* PERMLEXIKON, BESTZUWÄCHSE und ROSENBLUTH
2. *Supplementäre Koeffizienten:* SUPPLEMENTÄRKOEFF und SUPPLEMENTÄRTEILE
3. *Partikulare Koeffizienten:* DURCHSCHNITTSKOEFF und DURCHSCHNITTSTEILE
4. *Hierarchische Zerlegungen:* ATOMGEWICHTE, KOMBILEXIKON, KOMBIGEWICHTE und STUFENGEWICHTE
5. *Datenrauschen:* SUPPLEMENTÄRKOEFFRAUSCHEN und DURCHSCHNITTSKOEFFRAUSCHEN

Die Funktionen können jeder Excel-Installation ab Version 97 (bzw. 8.0) hinzugefügt werden, indem man die Datei „SeqReg.XLS“[2] öffnet. Anschließend lassen sie sich genauso verwenden wie die in Excel bereits vorhandenen Tabellenfunktionen.

[1] E-Mail: Norman.Fickel@wiso.uni-erlangen.de

[2] Internet: http://www.wiso.uni-erlangen.de/WiSo/VWI/s1/ (dort: Forschung, Diskussionspapiere)

1 Einleitung

Die Methoden der Sequenzialregression (Fickel 1999) sind nicht als eigenständige Software implementiert, sondern zu ihrer Benutzung wird die Standardsoftware Excel der Microsoft Corporation (1997) vorausgesetzt. Allerdings schränkt dies ihre Anwendbarkeit nicht erheblich ein, da Excel sowohl im Hochschulbereich als auch in der Wirtschaft weit verbreitet ist. Der Vorteil liegt besonders darin, dass man die zusätzlichen Funktionen nahtlos zusammen mit allen Möglichkeiten einer Tabellenkalkulationssoftware nutzen kann. Dies betrifft vor allem die Ein- und Ausgabe sowie die Aufbereitung der Daten. Dadurch ist es möglich, das neue Regressionsverfahren schnell und ohne größeren Aufwand auch auf eigene Datensätze anzuwenden.

Das vorliegende Diskussionspapier enthält den vollständigen Quellcode der Tabellenfunktionen, welche in der höheren Programmiersprache Visual Basic abgefasst sind. In den mit „Tabellenfunktionen“ betitelten Unterabschnitten ist dabei die Schnittstelle zu Excel beschrieben, wohingegen unter „Hilfsfunktionen“ die Programmteile mit den mathematischen Algorithmen zu finden sind. Diese Hilfsfunktionen können mit geringem Aufwand auch auf andere Dialekte von Basic angepasst werden oder sie lassen sich als Vorlage verwenden, um die Methoden in einer Sprache wie Pascal oder C zu programmieren.

Im Folgenden beziehen sich alle Verweise auf Fickel (1999).

2 Tabellenfunktionen in Microsoft Excel

Zur Benutzung der Tabellenfunktionen ist es nur nötig, ihre jeweiligen Parameter und die Form der Rückgabe zu kennen. Dazu geben die Parameterbezeichnungen *Y_Werte* und *X_Werte* jeweils die Tabellenbereiche an, welche die Regressandenwerte bzw. die Werte aller Regressoren enthalten. Dabei hat *Y_Werte* ein einspaltiger Vektor zu sein und die Spalten von *X_Werte* müssen zu den Regressoren in der Reihenfolge von links nach rechts gehören. Insbesondere besitzen dann sowohl *Y_Werte* als auch *X_Werte* gerade so viele Zeilen, wie Fälle im Datensatz vorhanden sind.

Man beachte, dass alle neuen Tabellenfunktionen (außer ROSENBLUTH) als Ergebnis nicht nur eine einzige Zahl, sondern eine ganze Matrix liefern. Es wird also ein bestimmter Tabellenbereich aufgefüllt, so wie es in der Excel-Hilfe unter dem Stichwort „Matrixformel“ näher beschrieben ist. Zum vereinfachten Aufruf listet Excels Funktionsassistent unter der Kategorie „Benutzerdefiniert“ alle Funktionsnamen zusammen mit den Parameterbezeichnungen auf.

Zur Auswahl von Reihenfolgen sind folgende Tabellenfunktionen verfügbar (vgl. Kapitel 6):

- PERMLEXIKON(*Anzahl*): Die Funktion liefert alle Permutationen von so vielen Elementen, wie es dem Parameter *Anzahl* entspricht. Dabei gehört jede Rückgabespalte zu einem Element und jede Rückgabezeile zu einer Permutation. Insbesondere gibt es also *Anzahl* Fakultät Rückgabezeilen.
- BESTZUWÄCHSE(*Y_Werte*; *X_Werte*): Es werden alle sequenziellen Zuwachsteile des multiplen Bestimmtheitsmaßes für sämtliche Permutationen der Spalten von *X_Werte* zurückgegeben, wobei die Rückgabespalten den Stufen der Regression entsprechen (vgl. Ab-

schnitt 6.2). Die Rückgabezeilen gehören zu den Permutationen, wie sie sich durch die Funktion PERMLEXIKON darstellen lassen.

- ROSENBLUTH(*Rho_Werte*): Die Funktion gibt das modifizierte Konzentrationsmaß nach Rosenbluth für die Werte des Parametervektors *Rho_Werte* zurück (vgl. Abschnitt 6.4.5). Wendet man sie auf die Ergebnisse der Funktion BESTZUWÄCHSE an, so kann mittels des in Excel vorhandenen Sortierbefehls die Permutation mit dem größten Rosenbluth-Maß gefunden werden.

Für die Ermittlung sequenzieller Koeffizienten gibt es zwei Tabellenfunktionen (vgl. Kapitel 7):

- SUPPLEMENTÄRKOEFF(*Y_Werte*; *X_Werte*): Die Rückgabematrix enthält in der ersten Spalte die supplementären Koeffizienten (vgl. Abschnitt 7.2.2) und in der zweiten Spalte die sequenziellen Bestimmtheitsmaßzuwächse (vgl. Abschnitt 7.3.1). Die Rückgabezeilen entsprechen den Spalten von *X_Werte*.

- SUPPLEMENTÄRTEILE(*Y_Werte*; *X_Werte*): Der totale Einfluss der letzten Spalte von *X_Werte* auf *Y_Werte* wird sequenziell zerlegt. Dazu enthält die Rückgabematrix in der ersten Spalte die Zerlegungsteile des totalen Regressionskoeffizienten (vgl. Abschnitt 7.3.2) und in der zweiten Spalte sind die zugehörigen Zerlegungsteile des totalen Bestimmtheitsmaßes (vgl. Abschnitt 7.3.3). Die Rückgabezeilen entsprechen den Spalten von *X_Werte*.

Die partikularen Koeffizienten können analog berechnet werden (vgl. Kapitel 8):

- DURCHSCHNITTSKOEFF(*Y_Werte*; *X_Werte*): Die Rückgabematrix enthält in der ersten Spalte die gemittelten Koeffizienten (vgl. Abschnitt 8.2.2) und in der zweiten Spalte die partikularen Komponenten des multiplen Bestimmtheitsmaßes (vgl. Abschnitt 8.3.1). Die Rückgabezeilen entsprechen den Spalten von *X_Werte*.

- DURCHSCHNITTSTEILE(*Y_Werte*; *X_Werte*): Der totale Einfluss der letzten Spalte von *X_Werte* auf *Y_Werte* wird in partikulare Komponenten aufgespalten: Die erste Spalte der Rückgabematrix enthält die Zerlegungsteile des totalen Regressionskoeffizienten und in der zweiten Spalte sind die zugehörigen Zerlegungsteile des totalen Bestimmtheitsmaßes (vgl. Abschnitt 8.3.3). Die Rückgabezeilen entsprechen den Spalten von *X_Werte*.

Zu den hierarchischen Zerlegungen sind vorhanden (vgl. Kapitel 8):

- ATOMGEWICHTE(*Y_Werte*; *X_Werte*): Alle Atome des multiplen Bestimmtheitsmaßes werden zurückgegeben (vgl. Abschnitt 8.4.2), wobei die Rückgabespalten zu den Spalten von *X_Werte* gehören. Die Rückgabezeilen entsprechen den Permutationen, wie sie sich durch die Funktion PERMLEXIKON darstellen lassen.

- KOMBILEXIKON(*Anzahl*): Die Funktion liefert alle Kombinationen von so vielen Elementen, wie der Parameter *Anzahl* angibt. Dabei gehört jede Rückgabespalte zu einem Element und ein Wert ungleich Null zeigt an, dass dieses in der Kombination der betreffenden Rückgabezeile enthalten ist. Da alle Kombinationen von einem oder mehreren Elementen ausgewiesen werden, gibt es insgesamt Zwei hoch *Anzahl* minus Eins Rückgabezeilen.

- KOMBIGEWICHTE(*Y_Werte*; *X_Werte*): Es werden die Gewichte aller Kombinationen in einem einspaltigen Vektor zurückgegeben (vgl. Abschnitt 8.4.2.1). Dabei entsprechen die Zeilen des Rückgabevektors den Kombinationen in der Anordnung, wie sie sich mit der Funktion KOMBILEXIKON darstellen lässt.
- STUFENGEWICHTE(*Y_Werte*; *X_Werte*): Die Rückgabematrix enthält die gleichzeitige Zerlegung des multiplen Bestimmtheitsmaßes nach Stufen und Regressoren (vgl. Abschnitt 8.4.2.3). Ihre Spalten entsprechen dabei den Spalten von *X_Werte* und ihre Zeilen den Stufen in aufsteigender Reihenfolge. Insbesondere hat sie so viele Zeilen wie Spalten. Mittels der Summenfunktion in Excel können Zeilen- und Spaltensummen gebildet werden, welche das Gesamtgewicht der einzelnen Stufen bzw. Regressoren anzeigen.

Schließlich ermöglichen es die letzten zwei Tabellenfunktionen, das Datenrauschen der supplementären und gemittelten Koeffizienten abzuschätzen (vgl. Abschnitt 7.4.3). Auch für die anderen Ergebnisse lassen sich theoretisch entsprechende Kennzahlen ermitteln. Jedoch wurde auf die Programmierung der zugehörigen Tabellenfunktionen verzichtet, um den ausgedruckten Quellcode überschaubar zu halten.

- SUPPLEMENTÄRKOEFFRAUSCHEN(*Y_Werte*; *X_Werte*; *Delta_Y_Wert*; *Delta_X_Werte*; *Iterationen*): Passend zu dem Ergebnis der Funktion SUPPLEMENTÄRKOEFF wird das Standardrauschen der Koeffizienten und Bestimmtheitsmaßzuwächse zurückgegeben (vgl. Abschnitt 7.4.3.3). Dabei stimmt die Anordnung der Rückgabematrix mit der von SUPPLEMENTÄRKOEFF überein. Als zusätzliche Parameter müssen angegeben werden: Die Standardabweichung des Regressanden *Delta_Y_Wert*, der einzeilige Vektor der Standardabweichungen der Regressoren *Delta_X_Werte* sowie die Anzahl der *Iterationen*, welche in der Zufallszahlensimulation durchgeführt werden sollen. Bei den in der vorliegenden Arbeit verwendeten Beispielsdaten liefert bereits ein Wert von 10 für *Iterationen* eine erste brauchbare Abschätzung.
- DURCHSCHNITTSKOEFFRAUSCHEN(*Y_Werte*; *X_Werte*; *Delta_Y_Wert*; *Delta_X_Werte*; *Iterationen*): Analog zur obigen Funktion wird das Standardrauschen zu dem Ergebnis der Funktion DURCHSCHNITTSKOEFF ermittelt (vgl. Abschnitt 8.5). Man beachte jedoch, dass die Rechenzeit erheblich höher ist, da in jeder Iteration gemittelte Koeffizienten bestimmt und dazu sämtliche Permutationen durchlaufen werden. Bei größeren Datensätzen empfiehlt es sich daher, zur Probe die Funktion zunächst für einen Wert von 1 für *Iterationen* aufzurufen.

Auf die Tabellenfunktionen kann zugegriffen werden, wenn man in Excel die Datei „SeqReg.XLS“ als Arbeitsmappe öffnet. Alternativ dazu lässt sich auch der Quellcode im zugehörigen Visual-Basic-Editor als unformatierter Text eingeben oder über die Zwischenablage einfügen.

3 Quellcode in der Programmiersprache Visual Basic

Der Quellcode ist weitgehend selbsterklärend, da er aus relativ kurzen Funktionen und Subprozeduren besteht, deren jeweiliger Zweck aus ihrem Namen erschlossen werden kann. Dazu wurde ausgiebig die Möglichkeit genutzt, in Visual Basic fast beliebig lange Zeichenketten als

Bezeichner zu verwenden. An technisch schwierigen Stellen ist auf den entsprechenden Abschnitt eines mathematischen Kapitels verwiesen.

Bei der Programmierung ist von den objektorientierten Konzepten in Visual Basic kein expliziter Gebrauch gemacht worden, da sich dadurch der Quellcode schwerer auf andere Dialekte von Basic übertragen ließe. Um dennoch seine Lesbarkeit zu erhöhen, wurden gedanklich folgende Objekttypen gebildet:

- *Regressionsaufgabe* (RA): Ein Objekt vom Typ RA umfasst alle Variablen, durch welche eine Regressionsaufgabe und ihr Bearbeitungsstand festgelegt werden. Neben den in der Tabelle übergebenen Ursprungsdaten aus *Y_Werte* und *X_Werte* sind dies auch bereinigte und/oder synchronisierte Versionen der Spalten von *X_Werte*.
- *Regressionsaufgabe mit Datenrauschen* (DR): Zusätzlich zu allen Variablen einer RA enthält ein Objekt von Typ DR die Informationen über das Ausmaß des Rauschens in den Daten aus den Parametern *Delta_Y_Wert* und *Delta_X_Werte*. Außerdem sind eine Reihe von Variablen eingeschlossen, die den aktuellen Bearbeitungsstand beschreiben.
- *Permutation* (PM): Jedes Objekt ist eine bestimmte Reihenfolge von Elementen.
- *Kombination* (KB): Jedes Objekt ist eine bestimmte Teilmenge von Elementen.

Die zu den einzelnen Objekttypen gehörigen Funktionen und Subprozeduren sind dadurch gekennzeichnet, dass ihrem Namen das Kürzel des Objekttyps, gefolgt von einem Unterstrich, vorangestellt ist. So bezeichnet etwa der Name *RA_Initialisiere* die Subprozedur, welche eine Regressionsaufgabe mit den Startwerten versieht. Die zugehörige Parameterliste beginnt stets mit den zum Datentyp gehörigen Variablen und erst anschließend sind jene Parameter aufgeführt, die sich auf den jeweiligen Kontext des Aufrufs beziehen.

3.1 Auswahl von Reihenfolgen

(Vgl. Kapitel 6 und 13)

3.1.1 Tabellenfunktionen

```
Function PermLexikon(Anzahl As Integer) As Variant
    Dim a() As Integer

    Call Erstelle_ein_Lexikon_aller_Permutationen(Anzahl, a)
    PermLexikon = a
End Function

Function BestZuwächse(Y_Werte As Variant, X_Werte As Variant) As Variant
    Dim x() As Double, q As Integer, n As Integer
    Dim a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    ReDim a(Fakultät(q) - 1, q - 1) As Double
    Call RA_Ermittle_alle_Zuwächse_im_Bestimmtheitsmaß(x, q, n, a)
    BestZuwächse = a
End Function

Function Rosenbluth(Rho_Werte As Variant)
    Dim q As Integer, k As Integer
    Dim Summe As Double, s As Double
```

```
    q = Excel.Application.Count(Rho_Werte)
    Summe = Excel.Application.Sum(Rho_Werte)
    s = 0

    For k = 1 To q
       s = s + k * Rho_Werte(k) / Summe
    Next k

    Rosenbluth = 1 / (2 * s - 1)
End Function
```

3.1.2 Hilfsfunktionen

```
Option Private Module

Public Const Rechnernull As Double = 0.00000000000001

Sub PM_Addiere_Eins(r() As Integer, q As Integer)
    Dim addiert As Boolean, i As Integer

    addiert = False
    For i = q To 1 Step -1
        If r(i) < q Then
            r(i) = r(i) + 1
            addiert = True
            Exit For
        Else
            r(i) = 1
        End If
    Next i

    If Not addiert Then
        For i = 1 To q
            r(i) = i
        Next i
    End If
End Sub

Function PM_Ist(r() As Integer, q As Integer) As Boolean
    Dim i As Integer, j As Integer, ist As Boolean

    ist = True
    For i = 1 To q - 1
        For j = i + 1 To q
            If r(i) = r(j) Then
                ist = False
                Exit For
            End If
        Next j

        If Not ist Then Exit For
    Next i

    PM_Ist = ist
End Function

Function Fakultät(q As Integer) As Long
    Dim p As Long, i As Integer

    p = 1
    For i = 1 To q
        p = p * i
    Next i
```

```
    Fakultät = p
End Function

Sub PM_Initialisiere(r() As Integer, q As Integer)
    Dim i As Integer

    For i = 1 To q
        r(i) = i
    Next i
End Sub

Sub PM_Nächste(r() As Integer, q As Integer)
    Do
        Call PM_Addiere_Eins(r, q)
    Loop Until PM_Ist(r, q)
End Sub

Sub Erstelle_ein_Lexikon_aller_Permutationen(q As Integer, a() As Integer)
    Dim p0 As Long, p As Long, r() As Integer, k As Integer

    p0 = Fakultät(q)
    ReDim r(q) As Integer
    Call PM_Initialisiere(r, q)
    ReDim a(p0, q)

    For p = 1 To p0
        For k = 1 To q
            a(p - 1, k - 1) = r(k)
        Next k
        Call PM_Nächste(r, q)
    Next p
End Sub

Sub RA_Zentriere(x() As Double, q As Integer, n As Integer, k As Integer)
    Dim s As Double, i As Integer

    s = 0
    For i = 1 To n
        s = s + x(i, k)
    Next i

    s = s / n
    For i = 1 To n
        x(i, k) = x(i, k) - s
    Next i
End Sub

Sub RA_Vervollständige_die_Initialisierung( _
        x() As Double, q As Integer, n As Integer)
    Dim i As Integer, k As Integer

    Call RA_Zentriere(x, q, n, 0)
    For k = 1 To q
        Call RA_Zentriere(x, q, n, q + k)
    Next k

    For k = 1 To q
        For i = 1 To n
            x(i, k) = x(i, q + k)
        Next i
    Next k

    x(0, q) = RA_Varianz(x, q, n, 2 * q)
    x(0, 0) = RA_Varianz(x, q, n, 0)
End Sub

Sub RA_Initialisiere(x() As Double, q As Integer, n As Integer, _
```

```
        Y_Werte As Variant, X_Werte As Variant)
    Dim i As Integer, k As Integer

    n = Excel.Application.Count(Y_Werte)
    q = Excel.Application.Count(X_Werte) / n
    ReDim x(n, 2 * q) As Double

    For i = 1 To n
        x(i, 0) = Y_Werte(i)
    Next i

    For k = 1 To q
        For i = 1 To n
            x(i, q + k) = X_Werte(i, k)
        Next i
    Next k

    Call RA_Vervollständige_die_Initialisierung(x, q, n)
End Sub

Function RA_Bestimmtheitsmaß(x() As Double, q As Integer, n As Integer, _
                                        k As Integer, l As Integer) As Double
    ' vgl. Abschnitt 11.5.6
    Dim s As Double, sk As Double, sl As Double, i As Integer
    Dim Produkt As Double

    s = 0
    sk = 0
    sl = 0
    For i = 1 To n
        s = s + x(i, k) * x(i, l)
        sk = sk + x(i, k) ^ 2
        sl = sl + x(i, l) ^ 2
    Next i

    Produkt = sk * sl
    If Abs(Produkt) > Rechnernull Then
        RA_Bestimmtheitsmaß = s ^ 2 / Produkt
    Else
        RA_Bestimmtheitsmaß = 0
    End If
End Function

Sub RA_Ermittle_alle_Zuwächse_im_Bestimmtheitsmaß( _
                    x() As Double, q As Integer, n As Integer, a() As Double)
    ' vgl. Abschnitt 13.5.2
    Dim i As Integer, k As Integer, l As Integer
    Dim p0 As Long, p As Long, r() As Integer

    p0 = Fakultät(q)
    ReDim r(q) As Integer
    Call PM_Initialisiere(r, q)

    For p = 1 To p0
        For k = 1 To q
            For i = 1 To n
                x(i, k) = x(i, q + r(k))
            Next i
        Next k

        For k = 2 To q
            For l = 1 To k - 1
                Call RA_Bereinige(x, q, n, k, l)
            Next l
        Next k

        For k = 1 To q
```

```
            a(p - 1, k - 1) = RA_Bestimmtheitsmaß(x, q, n, 0, k)
        Next k
        Call PM_Nächste(r, q)
    Next p
End Sub
```

3.2 Sequenzielle Koeffizienten

(Vgl. Kapitel 7 und 14)

3.2.1 Tabellenfunktionen

```
Function SupplementärKoeff(Y_Werte As Variant, X_Werte As Variant) _
                                                                    As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    ReDim a(q - 1, 1) As Double
    Call RA_Ermittle_die_supplementären_Koeffizienten(x, q, n, a)
    SupplementärKoeff = a
End Function

Function SupplementärTeile(Y_Werte As Variant, X_Werte As Variant) _
                                                                    As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    ReDim a(q - 1, 1) As Double
    Call RA_Zerlege_den_totalen_Einfluss(x, q, n, a)
    SupplementärTeile = a
End Function
```

3.2.2 Hilfsfunktionen

```
Option Private Module

Function RA_Einflussmaß(x() As Double, q As Integer, n As Integer, _
                                    k As Integer, l As Integer) As Double
    ' vgl. Abschnitt 11.2.7
    Dim sk As Double, sl As Double, i As Integer

    sk = 0
    sl = 0
    For i = 1 To n
        sk = sk + x(i, k) ^ 2
        sl = sl + x(i, k) * x(i, l)
    Next i

    If Abs(sk) > Rechnernull Then
        RA_Einflussmaß = sl / sk
    Else
        RA_Einflussmaß = 0
    End If
End Function

Function RA_Varianz(x() As Double, q As Integer, n As Integer, _
                                                    k As Integer) As Double
    ' vgl. Abschnitt 11.2.5
    Dim s As Double, i As Integer

    s = 0
```

```
    For i = 1 To n
        s = s + x(i, k) ^ 2
    Next i
    RA_Varianz = s / n
End Function

Sub RA_Bereinige(x() As Double, q As Integer, n As Integer, _
                                                k As Integer, l As Integer)
    Dim Faktor As Double, i As Integer

    Faktor = RA_Einflussmaß(x, q, n, l, k)
    For i = 1 To n
        x(i, k) = x(i, k) - Faktor * x(i, l)
    Next i
End Sub

Sub RA_Multipliziere(x() As Double, q As Integer, n As Integer, _
                                          k As Integer, Faktor As Double)
    Dim i As Integer

    For i = 1 To n
        x(i, k) = Faktor * x(i, k)
    Next i
End Sub

Sub RA_Synchronisiere(x() As Double, q As Integer, n As Integer, _
                                                k As Integer, l As Integer)
    ' vgl. Abschnitte 14.1.3
    Dim vk As Double, vl As Double

    vk = RA_Varianz(x, q, n, k)
    If Abs(vk) > Rechnernull Then
        vl = RA_Varianz(x, q, n, l)
        Call RA_Multipliziere(x, q, n, k, vl / vk)
    End If
End Sub

Sub RA_Bereinige_und_synchronisiere(x() As Double, q As Integer, _
                                                            n As Integer)
    Dim k As Integer, l As Integer

    For k = 2 To q
        For l = 1 To k - 1
            Call RA_Bereinige(x, q, n, k, l)
        Next l
        Call RA_Synchronisiere(x, q, n, k, q + k)
    Next k
End Sub

Sub RA_Ermittle_die_supplementären_Koeffizienten(x() As Double, _
                                  q As Integer, n As Integer, b() As Double)
    ' vgl. Abschnitte 14.2.1 und 14.3.1
    Dim k As Integer

    Call RA_Bereinige_und_synchronisiere(x, q, n)
    For k = 1 To q
        b(k - 1, 0) = RA_Einflussmaß(x, q, n, k, 0)
        b(k - 1, 1) = RA_Bestimmtheitsmaß(x, q, n, k, 0)
    Next k
End Sub

Sub RA_Ermittle_Teilkoeffizienten_und_Bestimmtheitsmaße(x() As Double, _
                                  q As Integer, n As Integer, b() As Double)
    ' vgl. Abschnitte 14.4.3 und 14.6.2
    Dim k As Integer
```

```
    For k = 1 To q
        b(k - 1, 0) = RA_Einflussmaß(x, q, n, q + q, k) * _
                      RA_Einflussmaß(x, q, n, k, 0)
b(k - 1, 1) = RA_Bestimmtheitsmaß(x, q, n, q + q, k) * _
                      RA_Bestimmtheitsmaß(x, q, n, q + q, 0)
Next k
End Sub

Sub RA_Zerlege_den_totalen_Einfluss(x() As Double, q As Integer, _
                                                    n As Integer, b() As Double)

    Call RA_Bereinige_und_synchronisiere(x, q, n)
    Call RA_Ermittle_Teilkoeffizienten_und_Bestimmtheitsmaße(x, q, n, b)
End Sub
```

3.3 Partikulare Koeffizienten

(Vgl. Kapitel 8 und 15)

3.3.1 Tabellenfunktionen

```
Function DurchschnittsKoeff(Y_Werte As Variant, X_Werte As Variant) _
                                                                    As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    ReDim a(q - 1, 1) As Double
    Call RA_Ermittle_die_gemittelten_Koeffizienten(x, q, n, a)
    DurchschnittsKoeff = a
End Function

Function DurchschnittsTeile(Y_Werte As Variant, X_Werte As Variant) _
                                                                    As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    ReDim a(q - 1, 1) As Double
    Call RA_Zerlege_gemittelt_den_totalen_Einfluss(x, q, n, a)
    DurchschnittsTeile = a
End Function
```

3.3.2 Hilfsfunktionen

```
Option Private Module

Sub RA_Setze_Bereinigung_zurück(x() As Double, q As Integer, n As Integer)
                                                Dim k As Integer, i As Integer

    For k = 1 To q
        For i = 1 To n
            x(i, k) = x(i, q + k)
        Next i
    Next k
End Sub

Sub RA_Bereinige_und_synchronisiere_reihenfolgenabhängig( _
                  x() As Double, q As Integer, n As Integer, r() As Integer)
    Dim k As Integer, l As Integer

    For k = 2 To q
        For l = 1 To k - 1
```

```
            Call RA_Bereinige(x, q, n, r(k), r(l))
        Next l
        Call RA_Synchronisiere(x, q, n, r(k), q + r(k))
    Next k
End Sub

Sub RA_Ermittle_die_gemittelten_Koeffizienten( _
                x() As Double, q As Integer, n As Integer, b() As Double)
    ' vgl. Abschnitt 15.1.2
    Dim i As Integer, k As Integer, l As Integer
    Dim Faktor As Double, r() As Integer
    Dim p0 As Long, p As Long, Varianz As Double

    For k = 1 To q
        b(k - 1, 0) = 0
        b(k - 1, 1) = 0
    Next k

    ReDim r(q) As Integer
    Call PM_Initialisiere(r, q)
    p0 = Fakultät(q)
    For p = 1 To p0
        Call RA_Setze_Bereinigung_zurück(x, q, n)
        Call RA_Bereinige_und_synchronisiere_reihenfolgenabhängig _
            (x, q, n, r)

        For k = 1 To q
            b(k - 1, 0) = b(k - 1, 0) + RA_Einflussmaß(x, q, n, k, 0)
            b(k - 1, 1) = b(k - 1, 1) + RA_Bestimmtheitsmaß(x, q, n, 0, k)
        Next k

        Call PM_Nächste(r, q)
    Next p

    For k = 1 To q
        b(k - 1, 0) = b(k - 1, 0) / p0
        b(k - 1, 1) = b(k - 1, 1) / p0
    Next k
End Sub

Sub RA_Zerlege_gemittelt_den_totalen_Einfluss(x() As Double, _
                                q As Integer, n As Integer, b() As Double)
    ' vgl. Abschnitt 15.4.1
    Dim a() As Double, c() As Double, k As Integer, r() As Integer
    Dim p0 As Long, p As Long

    For k = 1 To q
         b(k - 1, 0) = 0
         b(k - 1, 1) = 0
    Next k

    ReDim r(q) As Integer
    Call PM_Initialisiere(r, q)
    p0 = Fakultät(q - 1)
    ReDim c(q - 1, 1) As Double

    For p = 1 To p0
        Call RA_Setze_Bereinigung_zurück(x, q, n)
        Call RA_Bereinige_und_synchronisiere_reihenfolgenabhängig _
            (x, q, n, r)
        Call RA_Ermittle_Teilkoeffizienten_und_Bestimmtheitsmaße _
            (x, q, n, c)

        For k = 1 To q
             b(k - 1, 0) = b(k - 1, 0) + c(k - 1, 0)
             b(k - 1, 1) = b(k - 1, 1) + c(k - 1, 1)
        Next k
```

```
        Call PM_Nächste(r, q - 1)
    Next p

    For k = 1 To q
         b(k - 1, 0) = b(k - 1, 0) / p0
         b(k - 1, 1) = b(k - 1, 1) / p0
    Next k
End Sub
```

3.4 Hierarchische Zerlegungen

(Vgl. Kapitel 8 und 15)

3.4.1 Tabellenfunktionen

```
Function AtomGewichte(Y_Werte As Variant, X_Werte As Variant) As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    Call RA_Ermittle_alle_Atomgewichte(x, q, n, a)
    AtomGewichte = a
End Function

Function KombiLexikon(Anzahl As Integer) As Variant
    Dim a() As Integer

    Call Erstelle_ein_Lexikon_aller_Kombinationen(Anzahl, a)
    KombiLexikon = a
End Function

Function KombiGewichte(Y_Werte As Variant, X_Werte As Variant) As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    Call RA_Zerlege_nach_Kombinationen(x, q, n, a)
    KombiGewichte = a
End Function

Function StufenGewichte(Y_Werte As Variant, X_Werte As Variant) As Variant
    Dim x() As Double, q As Integer, n As Integer, a() As Double

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)
    Call RA_Zerlege_nach_Stufen_und_Regressoren(x, q, n, a)
    StufenGewichte = a
End Function
```

3.4.2 Hilfsfunktionen

```
Option Private Module

Sub RA_Ermittle_alle_Atomgewichte(x() As Double, q As Integer, _
                                  n As Integer, a() As Double)
    ' vgl. Abschnitt 15.7.7
    Dim i As Integer, k As Integer, l As Integer
    Dim p0 As Long, p As Long, r() As Integer

    p0 = Fakultät(q)
    p = 1
    ReDim r(q) As Integer
    Call PM_Initialisiere(r, q)
    ReDim a(p0 - 1, q - 1) As Double
```

```
    For p = 1 To p0
        Call RA_Setze_Bereinigung_zurück(x, q, n)
        For k = 2 To q
                For l = 1 To k - 1
                    Call RA_Bereinige(x, q, n, r(k), r(l))
                Next l
        Next k

        For k = 1 To q
            a(p - 1, k - 1) = RA_Bestimmtheitsmaß(x, q, n, 0, k) / p0
        Next k
        Call PM_Nächste(r, q)
    Next p
End Sub

Sub RA_Zerlege_nach_Stufen_und_Regressoren(x() As Double, q As Integer, _
                                                    n As Integer, a() As Double)
    ' vgl. Abschnitt 15.7.9, Teil d)
    Dim t() As Double, r() As Integer, l As Integer, k As Integer
    Dim p0 As Long, p As Long

    Call RA_Ermittle_alle_Atomgewichte(x, q, n, t)
    Call Erstelle_ein_Lexikon_aller_Permutationen(q, r)
    p0 = Fakultät(q)
    ReDim a(q - 1, q - 1)

    For l = 1 To q
        For k = 1 To q
            a(l - 1, k - 1) = 0
        Next k
    Next l

    For p = 1 To p0
        For k = 1 To q
            For l = 1 To q
                If r(p - 1, l - 1) = k Then
                    a(l - 1, k - 1) = _
                    a(l - 1, k - 1) + t(p - 1, k - 1)
                End If
            Next l
        Next k
    Next p
End Sub

Function RA_Ermittle_Gewicht_der_Kombination(x() As Double, q As Integer, _
                                        n As Integer, M() As Boolean) As Double
    ' vgl. Abschnitt 15.7.9, Teil b)
    Dim t() As Double, r() As Integer, l As Integer, k As Integer
    Dim h As Integer, p0 As Long, p As Long, a As Double, w As Boolean

    Call RA_Ermittle_alle_Atomgewichte(x, q, n, t)
    Call Erstelle_ein_Lexikon_aller_Permutationen(q, r)
    p0 = Fakultät(q)

    l = 0
    For k = 1 To q
        If M(k) Then
            l = l + 1
        End If
    Next k

    a = 0
    For p = 1 To p0
        For k = 1 To q

            If M(k) And (r(p - 1, l - 1) = k) Then
```

```
                w = True
                For h = 1 To l
                    w = w And M(r(p - 1, h - 1))
                Next h
            Else
                w = False
            End If

            If w Then
                a = a + t(p - 1, k - 1)
            End If
        Next k
    Next p

    RA_Ermittle_Gewicht_der_Kombination = a
End Function

Sub KB_Initialisiere(M() As Boolean, q As Integer)
    Dim k As Integer

    For k = 1 To q
        M(k) = False
    Next k
End Sub

Sub KB_Nächste(M() As Boolean, q As Integer)
    Dim k As Integer

    For k = q To 1 Step -1
        If M(k) Then
            M(k) = False
        Else
            M(k) = True
            Exit For
        End If
    Next k
End Sub

Sub Erstelle_ein_Lexikon_aller_Kombinationen(q As Integer, a() As Integer)
    Dim p0 As Long, p As Long, M() As Boolean, k As Integer

    p0 = 2 ^ q - 1
    ReDim M(q)
    Call KB_Initialisiere(M, q)
    Call KB_Nächste(M, q)
    ReDim a(p0 - 1, q - 1)

    For p = 1 To p0
        For k = 1 To q
            a(p - 1, k - 1) = M(k)
        Next k
        Call KB_Nächste(M, q)
    Next p
End Sub

Sub RA_Zerlege_nach_Kombinationen(x() As Double, q As Integer, _
                                                    n As Integer, a() As Double)
    ' vgl. Abschnitt 15.7.9, Teil b)
    Dim M() As Boolean, p0 As Long, p As Long

    p0 = 2 ^ q - 1
    ReDim a(p0 - 1, 0)
    ReDim M(q)
    Call KB_Initialisiere(M, q)

    For p = 1 To p0
        Call KB_Nächste(M, q)
```

```
        a(p - 1, 0) = RA_Ermittle_Gewicht_der_Kombination(x, q, n, M)
    Next p
End Sub
```

3.5 Datenrauschen

(Vgl. Kapitel 7 und 14)

3.5.1 Tabellenfunktionen

```
Function SupplementärKoeffRauschen(Y_Werte As Variant, _
                                   X_Werte As Variant, Delta_Y_Wert As Double, _
                         Delta_Werte As Variant, Iterationen As Long) As Variant

    SupplementärKoeffRauschen = VariantenRauschen(Y_Werte, X_Werte, _
        Delta_Y_Wert, Delta_Werte, Iterationen, 1)
End Function

Function DurchschnittsKoeffRauschen(Y_Werte As Variant, _
                                   X_Werte As Variant, Delta_Y_Wert As Double, _
                         Delta_Werte As Variant, Iterationen As Long) As Variant

    DurchschnittsKoeffRauschen = VariantenRauschen(Y_Werte, X_Werte, _
        Delta_Y_Wert, Delta_Werte, Iterationen, 2)
End Function
```

3.5.2 Hilfsfunktionen

```
Option Private Module

Sub RA_Ermittle_die_Koeffizienten_einer_Variante(x() As Double, _
             q As Integer, n As Integer, Variante As Integer, b() As Double)

    Select Case Variante
        Case 1
            Call RA_Ermittle_die_supplementären_Koeffizienten(x, q, n, b)
        Case 2
            Call RA_Ermittle_die_gemittelten_Koeffizienten(x, q, n, b)
    End Select
End Sub

Sub DR_Initialisiere(x() As Double, q As Integer, n As Integer, _
           delta() As Double, Erg() As Double, Erg_verrauscht() As Double, _
             Abw() As Double, x_Ursprung() As Double, Variante As Integer, _
                                 Y_Werte As Variant, X_Werte As Variant, _
                              Delta_Y_Wert As Double, Delta_Werte As Variant)
    Dim k As Integer, l As Integer, i As Integer

    Call RA_Initialisiere(x, q, n, Y_Werte, X_Werte)

    ReDim delta(q)
    delta(0) = Delta_Y_Wert
    For k = 1 To q
        delta(k) = Delta_Werte(k)
    Next k

    ReDim Erg(q - 1, 1)
    Call RA_Ermittle_die_Koeffizienten_einer_Variante(x, q, n, _
        Variante, Erg)
    ReDim Erg_verrauscht(q - 1, 1)
```

```
    ReDim Abw(q - 1, 1)
    For k = 1 To q
        For l = 0 To 1
            Abw(k - 1, l) = 0
        Next l
    Next k

    ReDim x_Ursprung(n, 2 * q)
    For k = 0 To 2 * q
        For i = 0 To n
            x_Ursprung(i, k) = x(i, k)
        Next i
    Next k
    Call Randomize   ' Initialisiere den Zufallszahlengenerator
End Sub

Sub RA_Verrausche_Vektor(x() As Double, q As Integer, n As Integer, _
                                                  k As Integer, delta As Double)
    ' vgl. Abschnitt 14.11.3
    Dim i As Integer, d As Double

    If delta > 0 Then
        For i = 0 To n
            d = Excel.Application.NormInv(Rnd, 0, delta)
            x(i, k) = x(i, k) + d
        Next i
    End If
End Sub

Sub RA_Verrausche(x() As Double, q As Integer, n As Integer, _
                                 delta() As Double, x_Ursprung() As Double)
    Dim k As Integer, i As Integer

    Call Randomize   ' Initialisiere den Zufallszahlengenerator
    Call RA_Verrausche_Vektor(x, q, n, 0, delta(0))
    For k = 0 To q
        Call RA_Verrausche_Vektor(x, q, n, q + k, delta(k))
    Next k

    Call RA_Vervollständige_die_Initialisierung(x, q, n)
End Sub

Sub DR_Verrausche(x() As Double, q As Integer, n As Integer, _
        delta() As Double, Erg() As Double, Erg_verrauscht() As Double, _
            Abw() As Double, x_Ursprung() As Double, Variante As Integer)
    ' vgl. Abschnitt 14.10.4
    Dim k As Integer, i As Integer, l As Integer, d As Double

    Call RA_Verrausche(x, q, n, delta, x_Ursprung)
    Call RA_Ermittle_die_Koeffizienten_einer_Variante(x, q, n, Variante, _
        Erg_verrauscht)

    For k = 1 To q
        For l = 0 To 1
            d = Erg_verrauscht(k - 1, l) - Erg(k - 1, l)
            Abw(k - 1, l) = Abw(k - 1, l) + d * d
        Next l
    Next k

    For i = 0 To n
        For k = 0 To 2 * q
            x(i, k) = x_Ursprung(i, k)
        Next k
    Next i
End Sub

Sub DR_Werte_Aus(x() As Double, q As Integer, n As Integer, _
```

```
            delta() As Double, Erg() As Double, Erg_verrauscht() As Double, _
              Abw() As Double, x_Ursprung() As Double, Variante As Integer, _
                                                                Anzahl As Long)
    ' vgl. Abschnitt 14.10.5
    Dim k As Integer, l As Integer

    For k = 1 To q
        For l = 0 To 1
            Abw(k - 1, l) = Sqr(Abw(k - 1, l) / Anzahl)
        Next l
    Next k
End Sub

Function VariantenRauschen(Y_Werte As Variant, X_Werte As Variant, _
                           Delta_Y_Wert As Double, Delta_Werte As Variant, _
                       Iterationen As Long, Variante As Integer) As Variant
    Dim x() As Double, q As Integer, n As Integer
    Dim delta() As Double, Erg() As Double, Erg_verrauscht() As Double
    Dim Abw() As Double, x_Ursprung() As Double, l As Long

    Call DR_Initialisiere(x, q, n, delta, Erg, Erg_verrauscht, _
        Abw, x_Ursprung, Variante, _
        Y_Werte, X_Werte, Delta_Y_Wert, Delta_Werte)

    For l = 1 To Iterationen
        Call DR_Verrausche(x, q, n, delta, Erg, Erg_verrauscht, _
            Abw, x_Ursprung, Variante)
    Next l

    Call DR_Werte_Aus(x, q, n, delta, Erg, Erg_verrauscht, _
        Abw, x_Ursprung, Variante, Iterationen)
    VariantenRauschen = Abw
End Function
```

4 Literatur

Fickel, Norman (1999): *Sequenzialregression: Eine neodeskriptive Lösung des Multikollinearitätsproblems mittels stufenweise bereinigter und synchronisierter Variablen.* Habilitationsschrift. Friedrich-Alexander-Universität, Nürnberg.

Jacobson, Reed (1997): *Microsoft Excel 97 Visual Basic Schritt für Schritt.* Microsoft Press, Unterschleißheim.

Microsoft Corporation (1997): *Microsoft Excel 97.* Computersoftware. Microsoft GmbH, Unterschleißheim.