Apl. Prof. Dr. Reinhold Kosfeld April 2025

Zeitreihenökonometrie mit R

Inhalt

1. Einführung

1.1 Vorbemerkungen
1.2 Rechnen mit Zahlen und Funktionen
1.3 Variablen, Vektoren und Matrizen
1.4 Zeitreihenobjekt und -diagramm
1.5 Einlesen von Zeitreihen aus Datendateien

2. Elementare Zeitreihenanalyse

2.1 Trend und glatte Komponente
2.1.1 Lineares und exponentielles Trendmodell
2.1.2 Methode der gleitenden Durchschnitte
2.2 Saisonkomponente und Saisonbereinigung
2.2.1 Phasendurchschnittsverfahren
2.2.2 Regressionsverfahren
2.3 Exponentielle Glättung
2.4. Autokorrelation
2.4.1 Autokorrelationskoeffizient und Korrelogramm
2.4.2 Portmanteau-Tests

3. Stochastische Prozesse und ARIMA-Modelle

3.1 White-Noise-Prozess und Random-Walk-Prozess
3.2 AR-Prozesse und MA-Prozesse
3.3 ARIMA-Modellierung ökonomischer Zeitreihen
3.4 ARIMA-Prognose ökonomischer Zeitreihen

4. Nichtstationarität und Kointegration

4.1 Einheitswuzeltests (Unit-Root-Tests)

4.2 Kointegration

5. ARCH- und GARCH-Modelle

5.1 Volatilitätscluster und Nichtlinearitäten

5.2 ARCH-Modelle und ARCH-LM-Test

5.3 GARCH-Modelle

1. Einführung
1.1 Vorbemerkungen

Die Statistiksoftware und Programmierumgebung R wird von einer internationalen Entwicklergemeinschaft als Open Source Software entwickelt und kann von jedermann entsprechend der GNU General Public License Version 2 (Juni 1991) frei genutzt und auch weiterverbreitet werden. Die Software wird auf der Homepage des R - Projektes (http://www.r-project.org) als Download bereitgestellt. Quellcode und Binärdateien für diverse Betriebssysteme werden durch CRAN bereitgestellt (http://CRAN.R-project.org).

R ist eine hochflexible, interpretierte Programmiersprache und –umgebung zur statistischen und grafischen Datenanalyse. Es handelt sich hierbei um keine vollständige, grafische Benutzeroberfläche (GUI), jedoch sind Werkzeuge zu ihrer Entwicklung vorhanden. In dem R-Commander (library(Rcmdr)) lassen sich aktuell bereits einige Methoden der Datenanalyse menügesteuert ausführen. Bei der Ausführung von R-Funktionen erfolgt in der Regel keine Ausgabe aller errechneten Werte. Vielmehr werden einige Ergebnisse erst einmal in Objekten zwischengespeichert. Sie können später jederzeit abgerufen werden. Andererseits speichert R keine Ergebnisse ab, wenn kein Objekt hierzu angegeben wird.

R kann sowohl interaktiv oder im Batch-Modus genutzt werden. Der Batch-Modus sollte bei größeren Befehlsfolgen verwendet werden, in dem eine Skript-Datei ausgeführt wird. Das Programm ist bereit, wenn das Zeichen > erscheint. Befehle können durch ein Semikolon (;) oder durch den Beginn einer neuen Zeile voneinander getrennt werden. Ist ein Befehl nicht vollständig, erscheint das Pluszeichen (+). Bereits verwendete Befehle können mit den Pfeiltasten (↑ und ↓) abgerufen werden. Mit dem Zeichen # wird ein Kommentar eingeleitet. Als Dezimalzeichen wird der Punkt (.), nicht das Komma (,) verwendet. Die ESC-Taste unterbricht die gerade ausgeführte Rechenprozedur.

Alle in einer R-Sitzung vorhandenen Objekte können mit dem Befehl ls() angezeigt. Ein Objekt obj lässt sich mit dem remove-Befehl rm(obj) löschen. Texte wie z.B. „Zeitreihenökonometrie macht Spaß“ können unter Verwendung von Anführungszeichen mit einem Gleichheitszeichen (=) oder Pfeil (<-) einem Objekt obj zugewiesen werden:

```
>obj = "Zeitreihenökonometrie macht Spaß"   oder   > obj <- "Zeitreihenökonometrie macht Spaß"
>obj
[1] " Zeitreihenökonometrie macht Spaß ".
```

Hier wird bei Zuweisungen allein von dem Gleichheitszeichen (=) Gebrauch gemacht. In diesem Fall ist das Objekt obj vom Typ character (String-Objekt (Textzeichenfolge)).

Die verfügbaren Objekte in der aktuellen R-Umgebung werden mit dem Befehl ls() in der R-Konsole angezeigt:

```
> ls()
 [1] "obj"
```

Das aktuelle Arbeitsverzeichnis (working directory) kann mit dem getwd()-Befehl überprüft werden:

```
> getwd()
[1] "C:/Users/Kosfeld/Documents/Zeitreihenoekonometrie/Zeitreihenoekonometrie mit R/Daten"
```

Es kann über das R-Menü File – Change Dir… oder mit dem setwd()-Befehl geändert werden:

```
> setwd("C:/Users/Kosfeld/Documents/Zeitreihenoekonometrie/Zeitreihenoekonometrie mit R/Daten").
```

Verlassen werden kann das Programm R durch den quit-Befehl q() oder durch Anklicken des Symbols x des Programmfensters R Console.

1.2 Rechnen mit Zahlen und Funktionen

Das Programm R kann als Taschenrechner benutzt werden, indem Rechenoperationen wie Addition (+), Subtraktionen (-), Multiplikationen (*), Divisionen (/) und Potenzierung (^) direkt ausgeführt werden:

```
> 3+4
[1] 7

> 7-12
[1] -5

> 5*6
[1] 30

> 9/4
[1] 2.25
```

Die Exponentiation mit einer beliebigen Basis erfolgt mit dem Zeichen ^,

```
> 2^4
[1] 16,
```

während sie zur Basis e (Eulersche Zahl: e = 2.718282) mit der exp-Funktion ausgeführt wird:

```
>exp(3)
[1] 20.08554
```

Die Quadratwurzel wird mit der sqrt-Funktion gezogen:

```
>sqrt(64)
[1] 8
```

Der Logarithmus zur Basis e wird mit der log-Funktion gebildet,

```
> log(8)
[1] 2.079442
```

und der Logarithmus zur Basis 10 mit der log10-Funktion:

```
> log10(1000)
[1] 3
```

Neben den arithmetischen Funktionen exp, sqrt und log führen wir an dieser Stelle die trigonometrischen Funktionen sin, cos und tan sowie ihre Umkehrfunktionen asin, acos und atan ein. Als Argument der Sinusfunktion muss der Winkel in Bogenmaß z.B. 2 (= 2*180/pi = 114.5916°) angegeben werden:

```
> sin(2)
[1] 0.9092974
```

Die Kreiskonstante π ist in R unter pi gespeichert:

```
>pi
[1] 3.141593
```

Mit der Konstanten pi lässt sich der mit der Arcussinus-Funktion asin in Bogenmaß ermittelte Winkel,

```
>asin(0.9)
[1] 1.119770,
```

in einen Winkel in Grad transformieren:

```
>asin(0.9)*180/pi
[1] 64.15807
```

Mit der Funktion round werden Zahlen auf ganze Stellen,

```
>round(1.8786)
[1] 2
```

oder auf die durch das Argument digit=2 angegebenen Dezimalstellen gerundet:

```
>round(1.8786, digit=2)
[1] 1.88
```

Übungsaufgaben

1.2.1 Berechnen Sie den Ausdruck $4 \cdot (6\frac{1}{2} - \frac{1}{2}) / 2$!

1.2.2 Berechnen Sie den Ausdruck $(8\frac{1}{3} + 2\frac{1}{2})^2 / \sqrt{2}$ und runden Sie ihn auf drei Dezimalstellen!

1.2.3 Berechnen Sie den Ausdruck $\log_{10} 1000 + \sqrt[3]{27}$!

1.2.4 Berechnen Sie den Ausdruck $x^2 = \log_2 16$!

1.2.5 Berechnen Sie den Ausdruck $\ln x = \sqrt{8}$!

1.3 Variablen, Vektoren und Matrizen

In den vorherigen Beispielen geht nach der Auswertung das Ergebnis nach seiner Ausgabe am Bildschirm verloren. Um die Ergebnisse der Rechenoperationen in einer Sitzung verfügbar zu haben, müssen sie unter einem Variablennamen gespeichert werden.

Das Ergebnis wird nicht mit der Zuweisung zu einer Variablen ausgegeben, sondern erst mit einem weiteren Befehl durch Nennung des Variablennamens:

```
> a = log10(1000)
> a
[1] 3
```

Das Objekt a ist hier der Spezialfall eines numerischen Vektors mit nur einem Element. Ein Vektor ist in R die einfachste Datenstruktur oder der einfachste Objekttyp.

Allgemein lässt sich ein Vektor durch Zuweisung von Zahlen mit Hilfe der Funktion c() definieren:

```
> x = c(3, 6, 7, 1, 9, 4)
> x
[1] 3 6 7 1 9 4
```

Ein Vektor mit einer Folge aufeinanderfolgender natürlicher Zahlen wird durch

```
> y = 1:6
> y
[1] 1 2 3 4 5 6
```

kreiert. Eine allgemeinere Sequenz von Zahlen kann mit dem seq()-Befehl durch Angabe eines Inkrements z.B. by = 0.5

```
> a = seq(1, 4, by = 0.5)
```

```
>a
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

oder der Länge z.B. length = 7

```
> b = seq(1, 4, length=7)
> b
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

einem Vektor zugewiesen werden. Das erste Argument gibt dabei jeweils den Startwert (from = 1), das zweite den Endwert (to = 4) an.

Unter Verwendung der einfachen Operatoren +, -, *, / und ^ werden Operationen mit Vektoren elementweise ausgeführt. In den folgenden Beispielen werden die Vektoren x,

```
> x
[1] 3 6 7 1 9 4,
```

und y,

```
> y
[1] 1 2 3 4 5 6,
```

verwendet:

```
> d = x + 3
> d
[1]  6  9 10  4 12  7

> e = 2*x
> e
[1]  6 12 14  2 18  8

> f = x + y
> f
[1]  4  8 10  5 14 10

> g = x * y
> g
[1]  3 12 21  4 45 24
```

Um das Skalarprodukt der beiden Vektoren x und y zu bilden, muss der Operator %*% verwendet werden:

```
> t(x)%*%y
    [,1]
[1,]  109
```

Hierbei wird der Zeilenvektor t(x) mit dem Spaltenvektor y multipliziert. Die Transponation eines Spaltenvektors in einen Zeilenvektor oder eines Zeilenvektors in einen Spaltenvektor erfolgt mit der Funktion t().

Das äußere Produkt (dyadische Produkt) der beiden Vektoren x und y ergibt sich durch Multiplikation des Spaltenvektors x mit dem Zeilenvektor t(y) unter Verwendung des Multiplikationsoperators %*%:

```
> x%*%t(y)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    3    6    9   12   15   18
[2,]    6   12   18   24   30   36
[3,]    7   14   21   28   35   42
[4,]    1    2    3    4    5    6
[5,]    9   18   27   36   45   54
[6,]    4    8   12   16   20   24
```

Das Ergebnis ist in diesem Fall eine 6x6-Matrix.

Aus der Sicht der Statistik lassen sich in einem Vektor die Werte eines Merkmals (Variablen) darstellen, die an n Untersuchungseinheiten erhoben worden sind. Allgemein werden bei einer Erhebung jedoch die Daten von m Merkmalen erfasst. Diese Daten lassen sich zweckmäßig in einer nxm-Matrix anordnen.

Eine Matrix lässt sich z.B. aus den Werten eines Vektors durch Angabe der Zeilen- und Spaltenzahl (nrow und ncol) mit der Funktion matrix erzeugen:

```
> A = matrix(c(3,5,6,2,8,4), nrow=3, ncol=2)
> A
     [,1] [,2]
[1,]    3    2
[2,]    5    8
[3,]    6    4
```

Das erste Argument data ist hierbei mit der Funktion c() spezifiziert. Standardmäßig wird die Matrix spaltenweise ausgefüllt. Sollen die Werte zeilenweise eingetragen werden, muss das Argument byrow gleich TRUE gesetzt werden:

```
> B = matrix(1:6, nrow=3, ncol=2, byrow=TRUE)
> B
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
```

Mit colnames() können den Spalten einer Matrix Namen zugeordnet werden:

```
>colnames(B) = c("b1", "b2")
```

```
> B
     b1 b2
[1,]  1  2
[2,]  3  4
[3,]  5  6
```

Entsprechend lassen sich mit rownames() den Zeilen einer Matrix Namen zuordnen. Beides gleichzeitig wird mit dem Befehl dimnames () bewerkstelligen:

```
>dimnames(B) = list(c("B1", "B2", "B3"),c("b1", "b2"))
> B
    b1 b2
B1  1  2
B2  3  4
B3  5  6
```

Die Dimensionen von Matrizen können mit der dim-Funktion abgefragt werden:

```
>dim(B)
[1] 3 2
```

Der erste Wert gibt die Zeilenzahl, der zweite die Spaltenzahl an.

Für manche Berechnungen ist es zweckmäßig, die Zeilen- und Spalten einer Matrix als Vektoren zu definieren. Enthalten die beiden Spalten der Matrix B die Werte zweier Variablen, können diese extrahiert und z.B. den Variablenvektoren b1 und b2 zugeordnet werden:

```
> b1 = B[,1]
> b1
B1 B2 B3
 1  3  5
> b2 = B[,2]
> b2
B1 B2 B3
 2  4  6.
```

Mit den Funktionen rbind und cbind lassen sich umgekehrt Vektoren durch zeilen- bzw. spaltenweises Zusammenfügen zu einer Matrix verbinden. So erhält man durch eine zeilenweise Verknüpfung der Vektoren b1 und b2 die Matrix B1,

```
> B1 = rbind(b1,b2)
> B1
   B1 B2 B3
b1  1  3  5
b2  2  4  6,
```

und durch spaltenweise Verknüpfung die Matrix B2:

```
> B2 = cbind(b1,b2)
> B2
 b1 b2
B1  1  2
B2  3  4
B3  5  6.
```

Übungsaufgaben

1.3.1Weisen Sie der Variablen x den Wert $4^3 - 6 \cdot 3$ und der Variablen y den Aus-druck $3 \cdot x + 5$ zu. Berechnen Sie den Ausdruck $z = \left(\frac{1}{3}x^2 - x \cdot \sqrt{y}\right)$!

1.3.2 Berechnen Sie $s = \frac{\sqrt[3]{(y^3 - 3 \cdot x + 4)}}{y - 2}$ unter Verwendung der in Aufgabe 1.3.1 angegebenen Zuweisungen für x und y!

1.3.3 Definieren Sie einen Vektor x aus den Werten 3, 5, 8, 2, 4, 1!

1.3.4 Erzeugen Sie einen Vektor y mit den Komponenten 4, 5, 6, 7, 8, 9 auf drei unterschiedliche Arten!

1.3.5 Addieren und multiplizieren Sie die in den Aufgaben 1.3.3 und 1.3.4 gebildeten Vektoren x und y elementweise und speichern Sie die Ergebnisse unter den Objektnamen s bzw. m!

1.3.6 Bilden Sie das Skalarprodukt der beiden Vektoren x und y (s. Aufg. 1.3.3 und 1.3.4)!

1.3.7 Bilden Sie aus den beiden Vektoren x und y (s. Aufg. 1.3.3 und 1.3.4) 2x3-Matrizen A und B!

1.3.8 Multiplizieren Sie die Matrix A mit der transponierten Matrix B (s. Aufg. 1.3.7) und speichern Sie das Matrixprodukt unter dem Objektnamen AB!

1.3.9 Füllen Sie eine 3x3-Matrix C zeilenweise mit den Werten 1 bis 9!

1.3.10 Weisen Sie den Spalten der in Aufg. 1.3.9 gebildeten Matrix C die Namen c1, c2 und c3 zu!

1.3.11 Verbinden Sie die in den Aufgaben 1.3.3 und 1.3.4 gebildeten Vektoren x und y zu einer 6x2-Matrix D!

1.4 Zeitreihenobjekte und -diagramm

Zeitreihen können mit der Funktion ts durch Angabe eines Vektors oder einer Matrix mit quantitativen Daten, der Start- und Endperiode sowie der Periodizität (frequency) als univariate – oder allgemeiner multivariate – Zeitreihen definiert werden. Beispielhaft seien die Jahresdaten des Kfz-Bestands für den Zeitraum 2001 bis 2009 als Werte der Zeitreihe kfz definiert:

```
> kfz = ts(c(27116,27858,28452,29122,29905,30618,31748,32762,33764), start=2001,
end=2009,frequency=1)
```

Die Anwendung der Funktion str auf das Objekt kfz weist seine interne Struktur aus,

```
> str(kfz)
 Time-Series [1:9] from 2001 to 2009: 27116 27858 28452 29122 29905 ...,
```

während mit der Funktion class die Objektklasse angegeben wird:

```
> class(kfz)
[1] "ts".
```

Definition und Inhalt des ts-Objekts kfz werden durch Eingabe des Objektnamens nach Bestätigung mit ENTER aufgelistet:

```
> kfz
Time Series:
Start = 2001
End = 2009
Frequency = 1
[1] 27116 27858 28452 29122 29905 30618 31748 32762 33764
```

Eines der beiden Argumente end oder frequency ist entbehrlich. Letzteres gibt die Anzahl der Zeitreihenwerte pro Jahr an:

- Jahresdaten: frequency=1
- Halbjahresdaten: frequency=2
- Quartalsdaten: frequency=4
- Monatsdaten: frequency=12

Anstelle der Periodizität (frequency) kann auch die äquidistante Differenz (deltat) Δx_t zwischen zwei aufeinanderfolgenden Zeitreihenwerten x_t und x_{t-1}als Argument der ts-Funktion angegeben werden.

Eine deskriptive Beschreibung einer Zeitreihe erhält man mit der summary-Funktion:

```
>summary(kfz)
  Min.  1st Qu.  Median   Mean   3rd Qu.   Max.
27116   28452   29905   30149   31748   33764
```

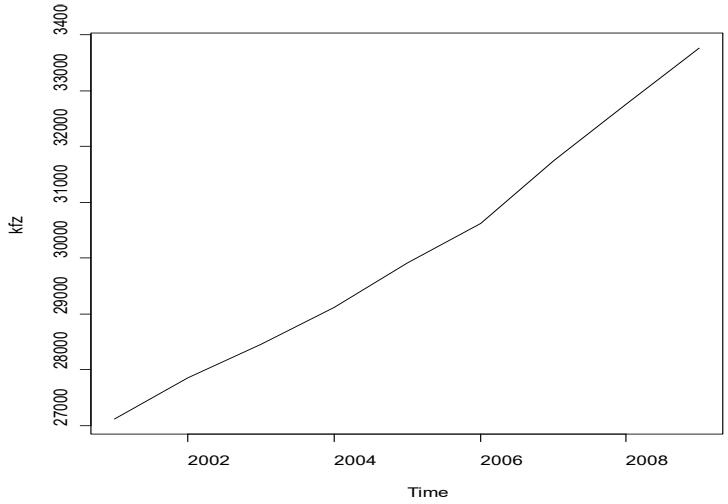
Sie besteht aus der Angabe des kleinsten und größten Zeitreihenwerts, der Quartile (2. Quartil = Median) und des arithmetischen Mittels.

Ein Zeitreihendiagramm lässt sich mit der plot-Funktion erstellen. Hierbei wird die Zeit t auf der Abzisse abgetragen, während die Ordinate die Werte der Zeitreihe (x_t) wiedergibt. Wird die Funktion plot auf ein Zeitreihenobjekt angewandt, hat R Kenntnis über die Zeit, so dass nur noch die Zeitreihe selbst genannt werden muss. Im einfachsten

Fall erhält man ein Zeitreihendiagramm z.B. für den Kfz-Bestand, indem man allein den Namen des Zeitreihenobjekts (kfz) in der plot-Funktion angibt:

```
>plot(kfz)
```

Abbildung 1.1: Elementares Zeitreihendiagramm des Kfz-Bestands



Das in Abbildung 1.1 wiedergegebene elementare Zeitreihendiagramm lässt sich durch die Aufnahme optionaler Argumente und Verwendung von Low-level-Befehlen verbessern, die die aktive Grafik verändern:

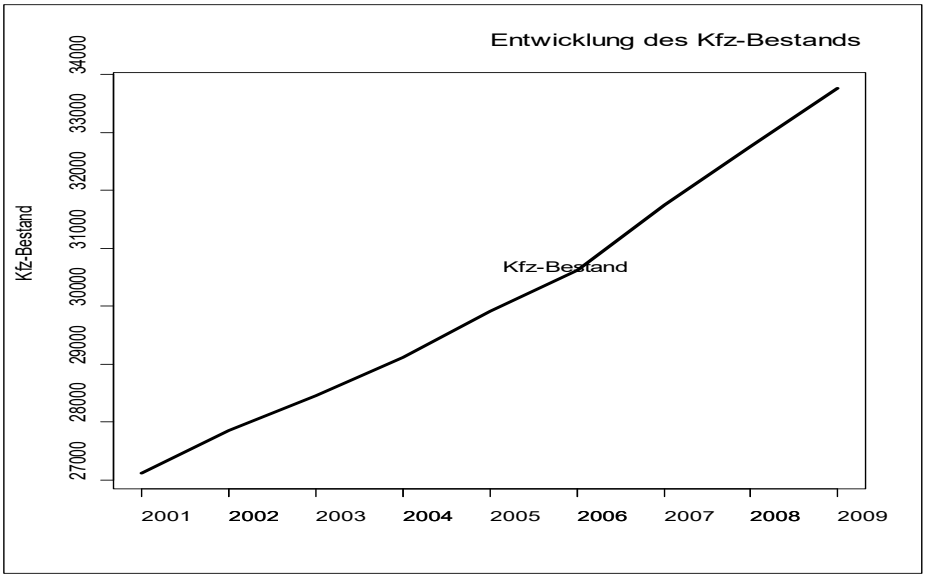
```
>plot(kfz, main="Entwicklung des Kfz-Bestands", xlab="", ylab="Kfz-Bestand", lwd=3, cex.main=1.25,
font.main=1)
>axis(side=1, at=c(2001,2002,2003,2004,2005,2006,2007,2008,2009))
>text(2005, 30700, "Kfz-Bestand", adj=0.4, cex=1.1)
>box(which="figure")
```

In der plot-Funktion (High-level-Befehl) legt das Argument main die Überschrift des Zeitreihendiagramms fest. Mit ylab wird die Beschriftung der Ordinate spezifiziert; xlab="" unterdrückt die Beschriftung der Abzisse mit „Time“. Während der Parameter lwd die Linienstärke des Zeitreihenpolygons (Kurvenzugs) kontrolliert, bestimmt der Parameter cex.main die Größe der Überschrift (Standardeinstellungen: lwd=1 und cex.main=1).Der Parameter font gibt den Schrifttyp (1 = einfach, 2 = fett, 3 = kursiv, 4 = fett kursiv) an.

Wie aus Abbildung 1.1 hervorgeht, erstellt das Programm R für die Zeitreihe kfz automatisch die Abzisse mit Angabe der geraden Jahreszahlen. Die Abzisse mit allen Jahreszahlen werden der aktiven Grafik mit dem axis-Befehl durch das Argument at hinzugefügt. Das Argument side=1 gibt dabei an, dass die Achse unten angefügt werden soll.

Mit dem text-Befehl erfolgt eine Beschriftung des Zeitreihenpolygons als Kfz-Bestand. Die Adjustierung des Textes wird über den Parameter adj (0: Ausrichtung links, 0.5: Ausrichtung mittig, 1: Ausrichtung rechts) gesteuert. Die Schriftgröße wird hierbei mit dem Parameter cex auf 1.1 festgelegt. Der box-Befehl mit dem Argument which=“figure“ oder which=“outer“ zieht einen äußeren Rahmen um die Grafik.

Abbildung 1.2: Zeitreihendiagramm des Kfz-Bestands



Wenn Vektoren und Matrizen zeitbezogene quantitative Daten enthalten, können sie mit dem Befehl as.ts oder ts in univariate bzw. multivariate Zeitreihen umgewandelt werden. So wird der Vektor kfz, der Beobachtungswerte des Kfz-Bestands enthält,

```
>kfz = c(27116, 27858, 28452,29122, 29905, 30618, 31748, 32762, 33764)
```

durch den Befehl

```
>kfz = as.ts(kfz)  oder  >kfz = ts(kfz)
```

in ein Zeitreihenobjekt umgewandelt, dessen Werte auf die Zeiteinheiten 1, 2, …, 9 bezogen werden:

```
>kfz
Time Series:
Start = 1
End = 9
Frequency = 1
[1] 27116 27858 28452 29122 29905 30618 31748 32762 33764
```

Bei Spezifizierung der Argumente start, end und frequency in dem as.ts-Befehl wie in der ts-Funktion lässt sich die Zeitreihe auf den Beobachtungszeitraum 2001 – 2009 datieren. Diese Datierung kann auch im Nachhinein mit dem ts-Befehl erreicht werden:

```
>kfz=ts(kfz, start=2001, frequency=1)
>kfz
Time Series:
Start = 2001
End = 2009
Frequency = 1
[1] 27116 27858 28452 29122 29905 30618 31748 32762 33764
```

Ein Ausschnitt einer Zeitreihe wird mit der Funktion window ausgewählt:

```
>window(kfz, start=2003, end=2007)

Time Series:
Start = 2003
End = 2007
Frequency = 1
[1] 28452 29122 29905 30618 31748
```

Hier wird der Ausschnitt auf den Beobachtungszeitraum 2003 – 2007 festgelegt.

Übungsaufgaben

1.4.1 Erzeugen Sie einen Vektor x aus den Werten 4, 6, 9, 7, 5, 8, 12, 11, 7, 9, 12, 10!

1.4.2 Definieren Sie den in Aufgabe 1.4.1 erstellten Vektor x als Zeitreihe zeitx und erstellen Sie ein elementares Zeitreihendiagramm!

1.4.3 Datieren Sie die in Aufgabe 1.4.2 definierte Zeitreihe zeitx auf den Zeitraum 2000 bis 2011!

1.4.4 Plotten Sie die in Aufgabe 1.4.2 datierte Zeitreihe zeitx mit einer Linienstärke von 3.5. Beschriften Sie die Ordinate und den Kurvenzug mit zeitx. Überschreiben Sie das Zeitreihendiagramm mit „Zeitreihe zeitx“ bei einer Schriftgröße von 1.15.

1.4.5 Zeichnen Sie ein Diagramm der Zeitreihe zeitx (s. Aufg. 1.4.3) für den Zeitraum von 2003 – 2009!

1.5 Einlesen von Zeitreihen aus Datendateien

Kurze Zeitreihen, die hier überwiegend zur Illustration der Methoden der Zeitreihenanalyse verwendet werden, lassen sich unaufwändig über die Tastatur in R eingeben. Lange oder multiple Zeitreihen werden dagegen üblicherweise von einer Datendatei eingelesen. Sind die Daten in einer .xls- oder .xlsx-Datei gespeichert, können sie mit einem Tabellenkalkulationsprogramm wie z.B. Excel in eine CSV-Textdatei (Comma Separated Values) umgewandelt werden. Sie lassen sich dann mit dem read.table- oder read.csv-Befehl in R einlesen.

Der Index der Auftragseingänge des Verarbeitenden Gewerbes (AE_VG) ist vom 2. Quartal 2008 bis zum 1. Quartal 2011 zusammen mit den Zeitindizes 2008.2, 2009.1, …. 2011.1 in der CSV-Datei Auftragseingaenge verfügbar. Die Daten können gleichwertig mit dem read.table- oder read.csv-Befehl in R eingelesen und als R-Objekt gespeichert werden:

```
> Auftrag = read.csv(file="Auftragseingaenge.csv", dec=".", sep=";", header=TRUE)
```

```
> Auftrag
    Zeit    AE_VG
1  2008.2   106.6
2  2008.3   108.6
3  2008.4   115.9
4  2009.1   122.1
5  2009.2   123.8
6  2009.3   117.8
7  2009.4   125.4
8  2010.1   130.7
9  2010.2   124.9
10 2010.3   128.5
11 2010.4   133.7
12 2011.1   137.7
```

Neben dem Dateinamen (file) sind in dem read.csv-Befehl einige optionale Parameter angegeben. Das Argument dec gibt an, dass der Punkt in der Datendatei Auftragseingaenge.csv als Dezimalzeichen verwendet worden ist. Die Werte selbst sind – nicht sichtbar – in dieser Datei durch Semikola voneinander getrennt (sep=“;“). Der logische Wert TRUE des Arguments header zeigt an, dass die erste Zeile der Datei Auftragseingaenge.csv die Variablennamen enthält.

Der Inhalt der Datei Auftragseingaenge.csv wird dem R-Objekt Auftrag zugewiesen. Die erste Spalte enthält als Zeitangaben die Jahre und Quartale der Beobachtungswerte unter dem Variablennamen Zeit. Die Indexwerte der Auftragseingänge sind unter dem Variablennamen AE_VG gespeichert.

Mit dem Befehl

```
>class(Auftrag)
[1] "data.frame"
```

wird der Typ des Objekts Auftrag ausgegeben. Auftrag ist noch kein Zeitreihenobjekt, sondern ein Objekt vom Typ data.frame, d.h. ein Datensatz, der wie ein Matrixobjekt aus Zeilen und Spalten besteht. In den Zeilen stehen die Beobachtungswerte der statistischen Einheiten, in den Spalten die Variablen. Im Unterschied zu einem matrix-Objekt kann ein data.frame-Objekt jedoch auch nicht-numerische Werte z.B. für qualitative Variablen enthalten.

Der str-Befehl

```
>str(Auftrag)
'data.frame':   12 obs. of  2 variables:
$ Zeit : num  2008 2008 2008 2009 2009 ...
 $ AE_VG: num  107 109 116 122 124 ...
```

gibt Aufschluss über die Struktur des data.frame-Objekts Auftrag.

Da die Auftragseingänge in R mit Methoden der Zeitreihenanalyse ausgewertet werden sollen, definieren wir die zweite Spalte des data.frame-Objekts Auftrag als Zeitreihe AEVG:

```
> AEVG = ts(Auftrag$AE_VG, start=c(2008,2), frequency=4)
> AEVG
      Qtr1  Qtr2  Qtr3  Qtr4
2008        106.6 108.6 115.9
2009 122.1 123.8 117.8 125.4
2010 130.7 124.9 128.5 133.7
2011 137.7
```

Das Argument start der Funktion ts ist jetzt ein Vektor bestehend aus dem Jahr (2008) und Quartal (2) des Anfangswerts der Zeitreihe AEVG. Der Wert 4 des Arguments frequency definiert Quartalsdaten, bei denen für ein komplettes Jahr vier Zeitreihenwerte vorliegen.

Mit dem Befehl

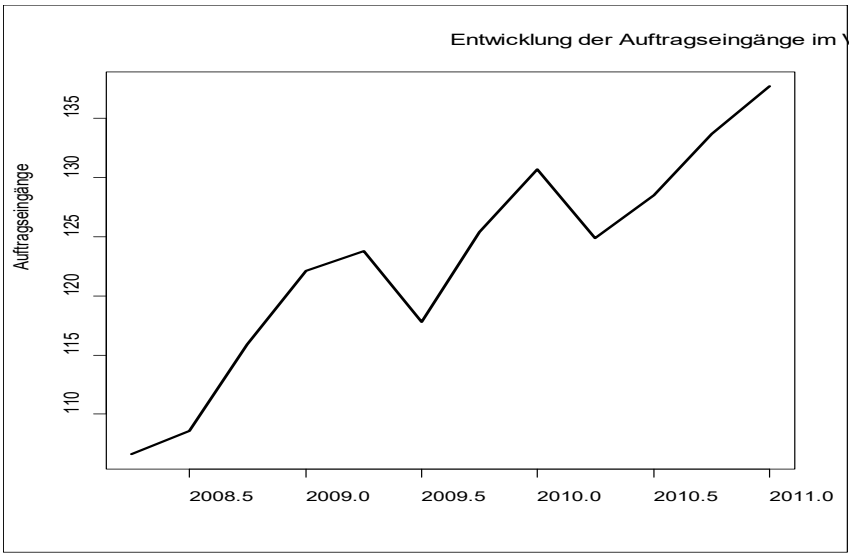
```
>class(AEVG)
[1] "ts"
```

erhalten wir die Bestätigung, dass das Objekt AEVG vom Typ ts, d.h. ein Zeitreihen-objekt, ist.

Das Zeitreihendiagramm von AEVG weist in dem Zeitraum vom 2. Quartal 2008 bis zum 1. Quartal 2011 einen zweimal unterbrochenen Aufwärtstrend auf:

```
>plot(AEVG, main="Entwicklung der Auftragseingänge im Verarbeitenden Gewerbe", xlab="",
ylab="Auftragseingänge",lwd=3, cex.main = 1.2, font.main=1)
> lines(AEVG.gl3, col="red", lwd=3)
>box(which="figure").
```

Abbildung 1.3: Zeitreihendiagramm der Auftragseingänge im Verarbeitenden Gewerbe



Die Werte auf der Zeitachse geben nach den Jahreszahlen die in R standardmäßig verwendete Bezeichnung der Quartale wieder: Quartal 1: 0, Quartal 2: 0.25, Quartal 3: 0.5, Quartal 4: 0.75. Aus Platzgründen stehen auf der Zeitachse pro Jahr jeweils nur die Labels für das erste und dritte Quartal.

Zur Illustration von Methoden der Saisonbereinigung ziehen wir später die Zeitreihe der Löhne und Gehälter je Beschäftigten heran, die für den Zeitraum vom 1. Quartal 2007 bis zum 4. Quartal 2011 zur Verfügung steht. Die Daten werden aus der CSV-Datei LuGBesch in R eingelesen und dem data.frame-Objekt LuGBesch zugewiesen:

```
> LuGBesch = read.csv(file="LuGBesch.csv", dec=".", sep=";", header=TRUE)
> head(LuGBesch)
   Zeit  Lohn
1 2007.1 113.6
2 2007.2 121.3
3 2007.3 122.0
4 2007.4 138.8
5 2008.1 116.3
6 2008.2 125.7
> str(LuGBesch)
'data.frame':   20 obs. of  2 variables:
$ Zeit: num  2007 2007 2007 2007 2008 ...
 $ Lohn: num  114 121 122 138 116 ...
>class(LuGBesch)
[1] "data.frame"
```

Anstatt mit LuGBesch den gesamten Datensatz in R anzufordern, wird mit dem head-Befehl nur der erste Teil ausgegeben.

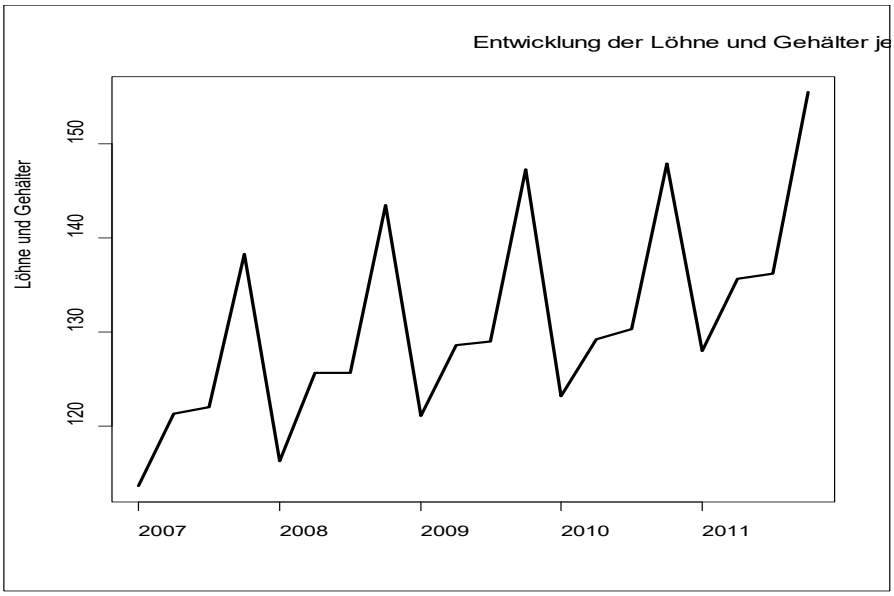
Eine Transformation der Lohn- und Gehaltsdaten des data.frame-Objekts LuGBesch in eine Zeitreihe Lohn erfolgt mit der ts-Funktion:

```
> Lohn = ts(LuGBesch[,2], start=c(2007,1), frequency=4)
> Lohn
  Qtr1  Qtr2  Qtr3  Qtr4
2007 113.6 121.3 122.0 138.8
2008 116.3 125.7 125.7 143.5
2009 121.1 128.6 129.0 147.3
2010 123.2 129.2 130.3 147.9
2011 128.0 135.7 136.2 155.5
> str(Lohn)
 Time-Series [1:20] from 2007 to 2012: 114 121 122 138 116 ...
> class(Lohn)
[1] "ts"
```

Aus dem Plot der Zeitreihe Lohn (Abbildung 1.4) geht ein aufsteigender Trend hervor, um den ein ausgeprägtes Saisonmuster erkennbar ist.

```
> plot(Lohn, main="Entwicklung der Löhne und Gehälter je Beschäftigten", xlab="",ylab="Löhne und Gehälter", lwd=3, cex.main = 1.2, font.main=1)
> box(which="figure")
```

Abbildung 1.4: Zeitreihendiagramm der Löhne und Gehälter je Beschäftigten



Übungsaufgaben

1.5.1 Erstellen Sie eine CSV-Datei Zeitx mit den Variablen Jahr (2000 – 2011) und zeitx (Daten s. Aufg. 1.4.1)!

1.5.2 Lesen Sie die CSV-Datei Zeitx als data.frame-Objekt Zeitx in R ein!

1.5.3 Definieren Sie die Variable zeitx des data.frame-Objekt Zeitx als Zeitreihe zeitx!

1.5.4 Plotten Sie die in Aufgabe 1.5.3 erstellte Zeitreihe zeitx unter Verwendung der kompletten Jahresangaben auf der Abzisse!

2. Elementare Zeitreihenanalyse

2.1 Trend und glatte Komponente

2.1.1 Lineares und exponentielles Trendmodell

Als erstes sollen globale Trends in Zeitreihen ohne Saison- und Konjunkturkomponente modelliert werden. Im einfachsten Fall ist eine Zeitreihe (x_t) durch einen linearen Trend geprägt.

(2.1) $x_t = \alpha + \beta \cdot t + u_t$, t=1.2….n.

Die Zeitvariable t nimmt hierbei Werte aus der Menge der natürlichen Zahlen von 1 bis n an, wobei n die Länge der Zeitreihe bezeichnet. Während die Zeitreihenwerte x_1, x_2, …, x_n in einem Zeitreihenobjekt verfügbar sind, muss die Zeitvariable t in R erst noch definiert werden. Das Trendmodell (2.1) lässt sich bei Gültigkeit der Standardannahmen mit der Methode der kleinsten Quadrate schätzen.

Da das in Abbildung 1.2 wiedergegebene Zeitreihendiagramm im Wesentlichen auf konstante jährliche Zuwächse des Kfz-Bestands in der Beobachtungsperiode 2001 – 2009 hinweist, soll der trendmäßigen Verlauf der Zeitreihe kfz zunächst durch ein lineares Zeitreihenmodell abgebildet werden.

Die mit der Funktion length bestimmte Länge der Zeitreihe kfz.

```
> n = length(kfz)
> n
[1] 9,
```

wird verwendet, um den Vektor tder Zeitindizes zu bestimmen:

```
> t = seq(from=1, to=n) oder  t = 1:n
> t
[1] 1 2 3 4 5 6 7 8 9.
```

Die lineare Trendfunktion der Zeitreihe kfz lässt sich unter Verwendung der lm-Funktion bestimmen, die allgemein zur Schätzung von Regressionsmodellen herangezogen werden kann, die linear in den Koeffizienten sind bzw. linearisiert werden können.

```
>kfz.lt = lm(kfz~t)
>kfz.lt

Call:
lm(formula = kfz ~ t)

Coefficients:
(Intercept)        t
   26033.4      823.2
```

Im einfachsten Fall wird die Funktion lm dadurch aufgerufen, dass in Klammern nur die Formel des zu schätzenden Modells angegeben wird. Abhängige und unabhängige Variablen werden durch eine Tilde (~) voneinander getrennt. Auf der linken Seite steht

die abhängige Variable, d.h. die Zeitreihe kfz. Auf der rechten Seite steht der Zeitvektor t als unabhängige Variable.

Die Kleinst-Quadrate-Schätzung der linearen Trendfunktion weist für die Zeitreihe kfz ein absolutes Glied (intercept) von 26033.4 und ein Steigungsmaß von 823.2 aus. Das absolute Glied gibt den Trendwert für das Jahr vor Beginn des Beobachtungszeitraum, d.h. das Jahr 2000 (Zeitindex 0), an. In diesem Jahr ist trendmäßig ein Bestand von 26033.4 Tsd. Kraftfahrzeugen erreicht worden. Das Steigungsmaß gibt an, dass der Bestand an Kraftfahrzeugen im Trend jährlich im Beobachtungszeitraum um 823.2 Tsd. Kraftfahrzeuge zugenommen hat.

Ein umfassenderer Output der Kleinst-Quadrate-Schätzung des linearen Trendmodells (2.1) wird durch die R-Funktion summary angezeigt:

```
>summary(kfz.lt)

Call:
lm(formula = kfz ~ t)

Residuals:
   Min     1Q     Median    3Q    Max
-354.64 -204.24  -47.84  178.16  321.76
Coefficients:
              Estimate   Std. Error  t value   Pr(>|t|)
(Intercept)  26033.44    185.24   140.54   2.44e-13 ***
t                823.20     32.92     25.01   4.17e-08 ***
---
Signif.codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 255 on 7 degrees of freedom
Multiple R-squared: 0.9889,    Adjusted R-squared: 0.9873
F-statistic: 625.4 on 1 and 7 DF,  p-value: 4.171e-08.
```

Aus ihm geht hervor, dass die geschätzten Regressionskoeffizienten hochsignifikant sind. Der lineare Trend „erklärt“ 98,9% der Varianz des Kfz-Bestands.

Einen visuellen Eindruck über die Güte der Anpassung der Trendgeraden an die Werte der Zeitreihe kfz erhält man aus einem Plot der beiden Zeitreihen in einem Diagramm. Das lm-Objekt kfz.lt besitzt einige Attribute, unter denen eine Reihe von Ergebnissen der Modellschätzung unter verschiedenen Namen gespeichert sind. Der Befehl attributes vermittelt eine Übersicht über die Attribute:

```
>attributes(kfz.lt)
$names
 [1] "coefficients"  "residuals"  "effects"  "rank"  "fitted.values" "assign"  "qr"  "df.residual"  "xlevels"
"call"
[11] "terms"   "model"
```

```
$class
[1] "lm"
```

Die Trendwerte sind in dem Attribut fitted.values enthalten, das mit fit abgekürzt werden kann:

```
>kfzlth = kfz.lt$fit
>kfzlth
     1        2        3        4        5        6        7        8        9
26856.64 27679.84 28503.04 29326.24 30149.44 30972.64 31795.84 32619.04 33442.24.
```

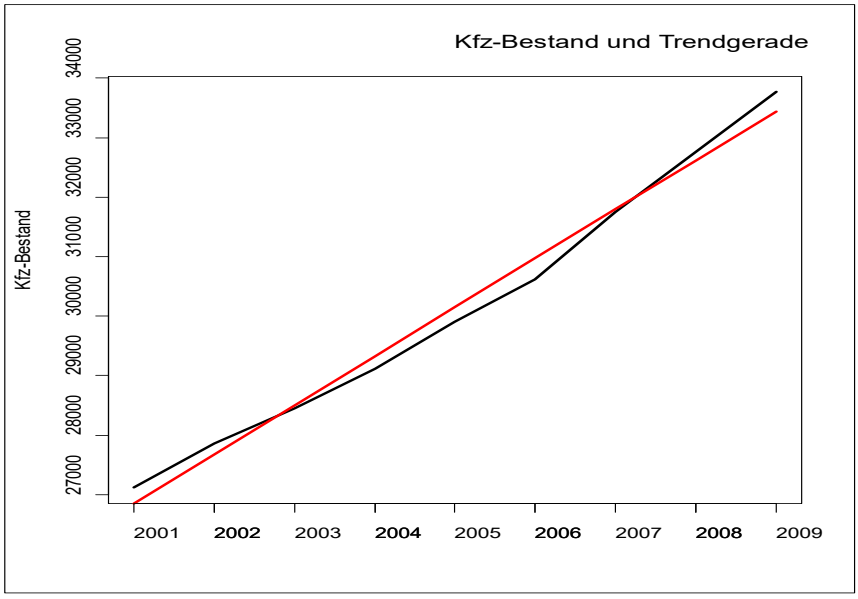
Die Transformation des Vektors kfzlth in eine Zeitreihe erfolgt mit dem bekannten ts- oder as.ts-Befehl:

```
>kfzlth = ts(kfzlth, start=2001, frequency=1)
>kfzlth
Time Series:
Start = 2001
End = 2009
Frequency = 1
     1        2        3        4        5        6        7        8        9
26856.64 27679.84 28503.04 29326.24 30149.44 30972.64 31795.84 32619.04 33442.24
```

Damit kann die Trendgerade in das Zeitreihendiagramm des Kfz-Bestands eingezeichnet werden (Abb. 2.1):

```
>plot(kfz, main="Kfz-Bestand und Trendgerade", xlab="", ylab="Kfz-Bestand", lwd=3, cex.main =
1.25, font.main=1)
>lines(kfzlth, col="red", lwd=3)
>axis(side=1, at=2001:2009)
>box(which="figure")
```

Abbildung 2.1: Zeitreihendiagramm des Kfz-Bestands mit Trendgerade



Unterstellt man die Gültigkeit des Trends innerhalb der nächsten 3 Jahre, lässt sich der zukünftige Kfz-Bestand mittels der linearen Trendfunktion für den Zeitraum 2010 – 2012 prognostizieren. Hierzu werden die Zeitindizes der nächsten drei Jahre (t=10,11,12) zusammen mit der Scheinvariablen (Wert 1 für alle Perioden) für das absolute Glied in einer Matrix X gespeichert:

```
>t.ext=c(1,1,1,10,11,12)
>t.ext
[1]  1  1  1 10 11 12
> x = matrix(t.ext,3,2)
>x
[,1] [,2]
[1,]    1   10
[2,]    1   11
[3,]    1   12
```

Die Trendkoeffizienten sind unter dem Namen coeffcients oder abgekürzt coeff als Attribut des lm-Objekts kfz.lt verfügbar:

```
>kfz.lt$coeff
 (Intercept)        t
  26033.44     823.20
```

Da das Attribut coeff ein Koeffizientenvektor ist, kann es in einer Matrixmultiplikation zur Berechnung der Prognosewerte kfzltf des Kfz-Bestands verwendet werden:

```
>kfzltf = x%*%kfz.lt$coeff
>kfzltf
[,1]
[1,] 34265.44
[2,] 35088.64
[3,] 35911.84
```

Um neben der Trendprognose die im Stützbereich angepassten Werte des Kfz-Bestands in einem Zeitreihendiagramm darzustellen, werden die Regressions- und Prognosewerte zu einem Objekt verbunden und als Zeitreihe definiert:

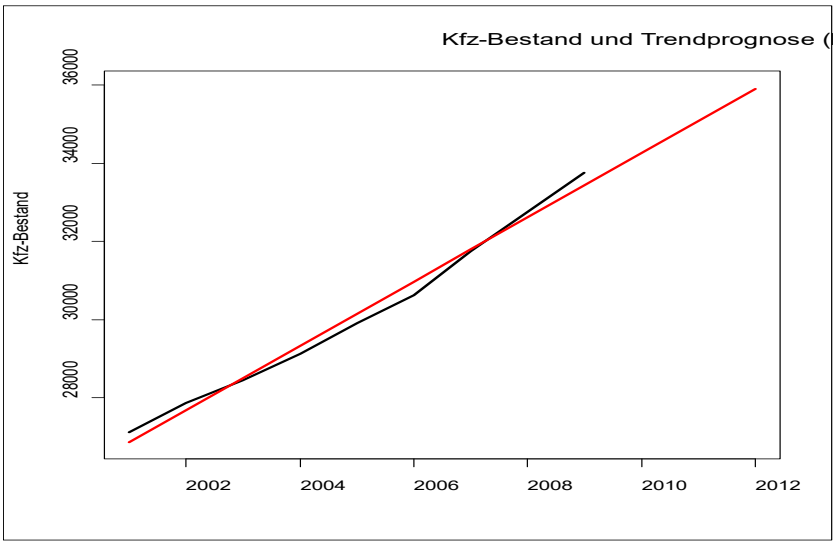
```
>kfzltp = ts(c(kfzlth,kfzltf), start=2001, frequency=1)
>kfzltp
Time Series:
Start = 2001
End = 2012
Frequency = 1
     1        2        3        4        5        6        7        8        9       10       11
    12
26856.64 27679.84 28503.04 29326.24 30149.44 30972.64 31795.84 32619.04 33442.24 34265.44 35088.64
35911.84
```

Das Zeitreihendiagramm des Kfz-Bestands wird mit den angepassten und prognostizierten Werten unter Verwendung der R-Befehle

```
> plot(kfz, main="Kfz-Bestand und Trendprognose (linearer Trend)", xlab="", ylab="Kfz-Bestand",xlim
= c(2001, 2012), ylim = c(26800, 36000), lwd=3, cex.main = 1.25, font.main=1)
> lines(kfzltp, col="red", lwd=3)
> box(which="figure")
```

erstellt. Aus Abbildung 2.2 lässt sich die zukünftige trendmäßige Entwicklung des Kfz-Bestands bei einem Prognosehorizont von drei Jahren erkennen.

Abbildung 2.2: Zeitreihendiagramm des Kfz-Bestands mit Trendprognose (linearer Trend)



Die Unterschätzung des Kfz-Bestands an den Rändern und seine Überschätzung in der Mitte des Stützbereichs durch die lineare Trendfunktion könnte auf einen - schwach - nichtlinearen Verlauf des Trends hinweisen, der z.B. durch die Konjunktur hervorgerufen sein könnte. In diesem Fall könnte die Anpassung durch eine exponentielle oder parabolische Trendfunktion verbessert werden. Hier soll der Exponentialtrend

(2.2) $m_t = \alpha \cdot \beta^t$

betrachtet werden, der angewandt auf eine Zeitreihe (x_t) in Form des linearisierten Regressionsmodells

(2.3) $\log(x_t) = \log(\alpha) + \log(\cdot\beta) \cdot t + u_t$, t=1.2….,n,

mit der Methode der kleinsten Quadrate geschätzt werden kann.

Wählt man den Logarithmus zur Basis 10 (dekadischer Logarithmus), erfolgt die Kleinst-Quadrate-Schätzung des linearisierten Trendmodells (2.3) mit dem R-Befehl

```
> kfz.et = lm(log10(kfz)~t)
> kfz.et

Call:
lm(formula = log10(kfz) ~ t)
```

```
Coefficients:
(Intercept)         t
   4.41910     0.01182.
```

Nach Bildung des Antilogarithmus erhält man die Schätzer für die Trendkoeffizienten α und β der exponentiellen Trendfunktion (2.2):

```
> kfzet.coeff10 = 10^kfz.et$coeff
> kfzet.coeff10
   (Intercept)          t
26248.348939    1.027588
```

Danach liegt der Trendwert im Jahr 2000 bei 26248 Kraftfahrzeugen, die trendmäßig pro Jahr um 2,8% zulegen.

Auch hier soll wiederum der mit der summary-Funktion erzeugte umfassendere Output der Kleinst-Quadrate-Schätzung wiedergegeben werden:

```
> summary(kfz.et)

Call:
lm(formula = log10(kfz) ~ t)

Residuals:
 Min        1Q     Median        3Q       Max
-0.0040389 -0.0021566 -0.0001183  0.0022100  0.0029813

Coefficients:
              Estimate    Std. Error    t value    Pr(>|t|)
(Intercept)  4.4191020   0.0019265   2293.85   < 2e-16 ***
t            0.0118190   0.0003423     34.52   4.44e-09 ***
---
Signif.codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.002652 on 7 degrees of freedom
Multiple R-squared: 0.9942,    Adjusted R-squared: 0.9933
F-statistic:  1192 on 1 and 7 DF,  p-value: 4.437e-09.
```

Erneut sind die beiden geschätzten Regressionskoeffizienten hochsignifikant. Das Bestimmtheitsmaß Multiple R-squared zeigt an, dass der Exponentialtrend 99,4% der logarithmierten Bestandswerte „erklärt". Der Erklärungsgehalt des Exponentialtrends in Bezug auf die originären Werte des Kfz-Bestands beträgt dagegen 98,1%:

```
>varkfz = var(kfz)
> kfzeth = kfz.et$fit
> kfzet10h = 10^kfzeth
> varkfzet10h = var(kfzet10h)
> r2.et = varkfzet10h/varkfz
> r2.et
[1] 0.9813308
```

Mit den logarithmierten Prognosewerten

```
>kfzetf = x.ext%*%kfz.et$coeff
>kfzetf
         [,1]
[1,] 4.537292
[2,] 4.549110
[3,] 4.560929
```

erhält man die Zeitreihe der logarithmierten Trendwerte im Zeitraum 2001-2012:

```
>kfzetp = ts(c(kfzeth,kfzetf), start=2001, frequency=1)
>kfzetp
Time Series:
Start = 2001
End = 2012
Frequency = 1
     1        2        3        4        5        6        7        8        9       10       11
    12
4.430921 4.442740 4.454559 4.466378 4.478197 4.490016 4.501835 4.513654 4.525473 4.537292 4.549110
4.560929
```

Die exponentiellen Trendwerte für die Zeitreihe kfz erhält man durch Bildung des Antilogarithmus:

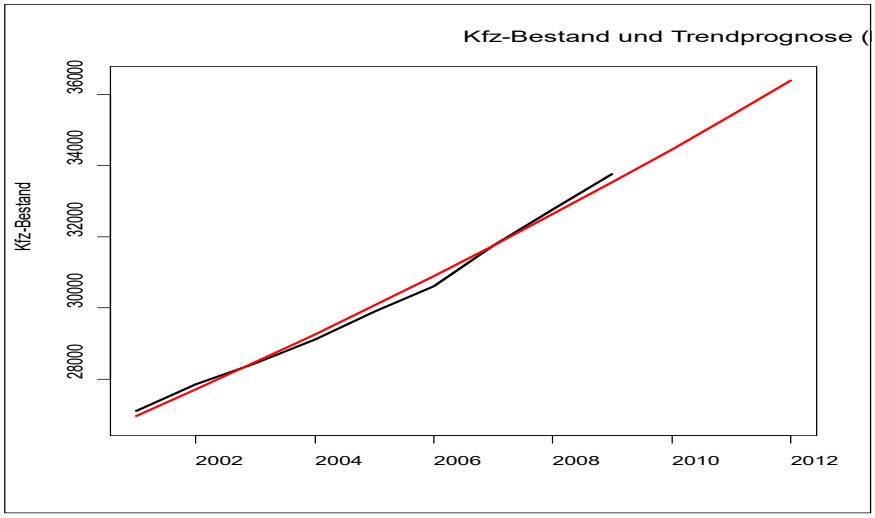
```
> kfzet10p = 10^kfzetp
> kfzet10p
Time Series:
Start = 2001
End = 2012
Frequency = 1
     1        2        3        4        5        6        7        8        9       10       11
    12
26972.48 27716.60 28481.24 29266.97 30074.39 30904.07 31756.65 32632.75 33533.01 34458.12 35408.74
36385.59
```

Damit kann das Zeitreihendiagramm des Kfz-Bestands mit den angepassten und prognostizierten exponentiellen Trendwerte gezeichnet werden (Abb. 2.3):

```
>plot(kfz, main="Kfz-Bestand und Trendprognose (Exponentialtrend)", xlab="", ylab="Kfz-Bestand",xlim
= c(2001, 2012), ylim = c(26800, 36400), lwd=3, cex.main = 1.25, font.main=1)
>lines(kfzet10p, col="red", lwd=3)
> box(which="figure")
```

Auf der Grundlage des Exponentialtrends liegt die Bestandsprognose über derjenigen des linearen Trendmodells.

Abbildung 2.3: Zeitreihendiagramm des Kfz-Bestands
mit Trendprognose (Exponentialtrend)



Visuell lassen kaum Unterschiede in der Güte der Anpassung zwischen dem linearen und exponentiellen Trendmodell erkennen. Da das Bestimmtheitsmaß für den linearen Trend (R^2 = 0,989) geringfügig über demjenigen des Exponentialtrends (R^2 = 0,981) liegt, besteht aufgrund dieses Kriteriums kein Anlass, das ursprüngliche Trendmodell aufzugeben.

Übungsaufgaben

2.1.1 Bestimmen Sie die Länge n der in Aufgabe 1.4.2 definierten Zeitreihe zeitx und erzeugen Sie einen Vektor t aus den Werten 1 bis n!

2.1.2 Schätzen Sie eine lineare Trendfunktion für die Zeitreihe zeitx unter Verwendung der in dem Vektor t (s. Aufg. 2.1.1) enthaltenen Zeitindizierung! Speichern Sie die Standardergebnisse unter dem Objektnamen zeitx.ls!

2.1.3 Rufen Sie die erweiterten Ergebnisse der in Aufgabe 2.1.2 geschätzten linearen Trendfunktion für die Zeitreihe zeitx ab und interpretieren Sie den Standardfehler der Regression, das Bestimmtheitsmaß und die F-Statistik!

2.1.4 Speichern Sie die Trendwerte der Regression von zeitx auf t (s. Aufg. 2.1.2) als Zeitreihenobjekt zeitxlth!

2.1.5 Erstellen Sie ein Zeitreihendiagramm für die in den Aufgaben 1.4.2 und 2.1.4 gebildeten Zeitreihen zeitx und zeitxth unter Angabe der Jahreszahlen 2000 – 2011 auf der Abzisse! Beschriften Sie dabei die Ordinate mit „zeitx“ und spezifizieren Sie die Überschrift „Zeitreihe zeitx und Trendgerade“!

2.1.6 Prognostizieren Sie die Zeitreihe zeitx (s. Aufg. 1.4.2) unter Verwendung der linearen Trendfunktion (s. Aufg. 2.1.2) für den Zeitraum 2012 – 2014 und speichern Sie sie unter dem Objektnamen zeitxltf! Verbinden Sie die Trendwerte im Stützzeitraum, zeitxlth, (s. Aufg. 2.1.4) mit den Prognosewerten zeitltf zu einem einheitlichen Zeitreihenobjekt zeitltp!

2.1.7 Erstellen Sie ein Zeitreihendiagramm der Zeitreihen zeitx und zeitxtp unter Verwendung der in Aufgabe 2.1.5 angegebenen Argumente für die Achsen und der Überschrift „Zeitreihe zeitx und Trendprognose (linearer Trend)“!

2.1.2 Methode der gleitenden Durchschnitte

Mit der Methode der gleitenden Durchschnitte lässt sich in einfacher Form eine Glättung einer Zeitreihe vornehmen. Hierzu werden lokale Mittelwerte gebildet und jeweils dem mittleren Zeitindex zugeordnet. Falls in einer Zeitreihe kein klares Saisonmuster enthalten ist, wählt man zur lokalen Mittelwertbildung eine ungerade Anzahl von Beobachtungswerten, da in diesem Fall die zeitliche Zuordnung eindeutig ist. Man spricht dann von einem zentrierten gleitenden Durchschnitt. Aufgrund seiner Flexibilität bildet der gleitende Durchschnitt die glatte Komponente, d.h. den Trend in Verbindung mit evtl. konjunkturellen Schwankungen, ab.

Ein **zentrierter gleitendender Durchschnitt der Ordnung p** (zentrierter p-gliedriger gleitender Durchschnitt), $\bar{x}_t^p$, ist bei **ungeradem p** ein einfacher Mittelwert aus dem Zeitreihenwert der Periode t sowie den q vor- und nachlaufenden Werten

$$(2.4)\quad \bar{x}_t^p = \frac{1}{p}\sum_{i=-q}^{q} x_{t+i}\,.$$

mit p=2q+1. Die in (2.4) verwendeten Zeitreihenwerten x_{t-q}, x_{t-q+1}, ..., x_t, ..., x_{t+q-1}, x_{t+q} gehen gleichgewichtig in den gleitenden Durchschnitt ein. Je größer q ist, desto größer ist der Glättungseffekt. Bei kleinem q gehen lokale Eigenschaften stärker in den leitenden Durchschnitt ein.

Der Vorteil der Flexibilität der Methode der gleitenden Durchschnitte wird jedoch durch einen Nachteil „erkauft“. Aus Formel (2.4) geht hervor, dass an den beiden Rändern der Zeitreihe jeweils q Durchschnittswerte unbesetzt bleiben.

Obwohl die Quartalsreihe der Auftragseingänge im Verarbeitenden Gewerbe nicht frei von jahreszeitlichen Schwankungen ist, treten diese jedoch nicht in jedem Jahr auf. Aus Abbildung 1.3lässt sich kein klares Saisonmuster erkennen. Hier soll die Zeitreihe AEVG mittels gleitender Durchschnitte der Ordnung 3 geglättet werden. In R können gleitende Durchschnitte mit der Funktion filter gebildet werden:

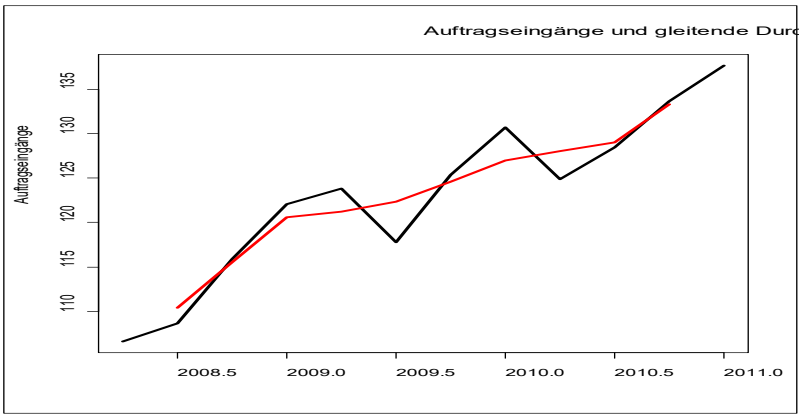
```
> AEVG.gl3 = filter(AEVG, filter=rep(1/3,3), sides=2)
> AEVG.gl3
  Qtr1     Qtr2     Qtr3     Qtr4
2008                NA 110.3667 115.5333
2009 120.6000 121.2333 122.3333 124.6333
2010 127.0000 128.0333 129.0333 133.3000
2011       NA
>class(AEVG.gl3)
[1] "ts"
```

In der Funktion filter muss als Argument die zu glättende Zeitreihe angegeben werden. Außerdem muss der Filter in Form der Gewichte spezifiziert werden, was hier mit der rep-Funktion erfolgt. Bei einem 3-gliedrigen gleitenden Durchschnitt wird das Gewicht 1/3 für die 3 eingehenden Zeitreihenwerte verwendet. Durch das Setzen des Arguments sides auf 2 wird ein zentrierter gleitender Durchschnitt angefordert (Standardeinstellung).

Mit den R-Funktionen plot und lines lässt sich das Zeitreihendiagramm der Auftragseingänge mit dem gleitenden Durchschnitt zeichnen:

```
>plot(AEVG, main="Auftragseingänge und gleitende Durchschnitte", xlab="",ylab="Auftragseingänge",
lwd=3, cex.main = 1.2, font.main=1)
>lines(AEVG.gl3, col="red", lwd=3)
> box(which="figure")
```

Abbildung 2.4: Zeitreihendiagramm der Auftragseingänge mit 3-gliedrigen gleitenden Durchschnitten



Aus Abbildung 2.4 geht der Glättungseffekt der gleitenden Durchschnittsbildung deutlich hervor.

Weist eine Zeitreihe dagegen ein ständig wiederkehrendes Saisonmuster auf, müssten alle Saisonabschnitte (Quartale, Monate) eines Jahres in die gleitende Mittelwertbildung einbezogen werden, um saisonale Effekte aus der Trend-Konjunktur-Komponente (= glatte Komponente) auszuschließen. Bei Quartalsdaten wäre p gleich 4 und bei Monatsdaten p gleich 12 zu setzen. Jedoch würde es bei Verwendung einer geraden Anzahl von Zeitreihenwerten keine mittlere Periode geben, denen der gleitende Durchschnitt zugeordnet werden könnte.

Aus diesem Grund zieht man bei Quartalsdaten einen fünften Wert zur Bestimmung des gleitenden Durchschnitts heran, den man wie den ersten Wert nur halb so stark gewichtet wie die drei mittleren Werte. Da die beiden Randwerte denselben Quartalsindex haben, werden dadurch Verzerrungen der glatten Komponente vermieden.

Bei **geradem p** wird der **zentrierte gleitendender Durchschnitt der Ordnung p** daher im Falle von **Quartalsdaten** durch

$$(2.5)\quad \bar{x}_t^4 = \frac{1}{8}x_{t-2} + \frac{1}{4}\sum_{i=-1}^{1} x_{t+i} + \frac{1}{8}x_{t+2}$$

und im Falle von **Monatsdaten** durch

$$(2.6)\quad \bar{x}_t^{12} = \frac{1}{24}x_{t-6} + \frac{1}{12}\sum_{i=-5}^{5} x_{t+i} + \frac{1}{24}x_{t+6}$$

definiert. Hierbei bleiben an den Rändern des Beobachtungszeitraums jeweils q=p/2 gleitende Durchschnittswerte unbesetzt.

Die Berechnung zentrierter gleitender Durchschnitte lässt sich ebenfalls unter Verwendung der filter-Funktion vornehmen. Bei Quartalsdaten ist hierzu der Filter c(1/8, 1/4, 1/4, 1/4, 1/8) zu verwenden. Für die Zeitreihe Lohn ergibt sich hiermit der zentrierte 4-gliedrige gleitende Duchschnitt Lohn.gl4:

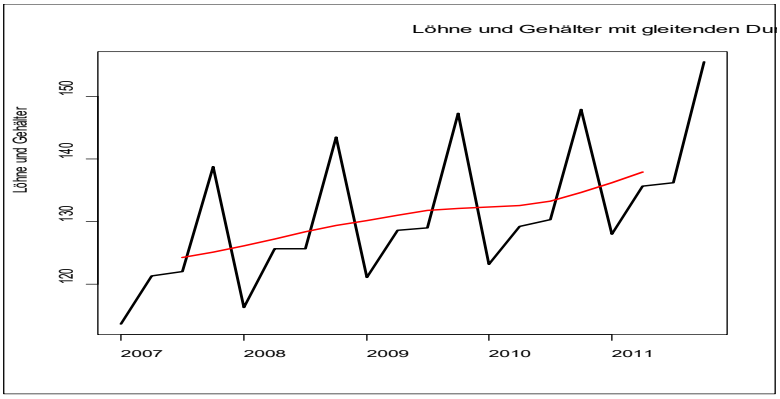
```
Lohn.gl4 = filter(Lohn, filter=c(1/8, rep(1/4, 3), 1/8), sides=2)
Lohn.gl4
        Qtr1       Qtr2       Qtr3       Qtr4
2007      NA         NA   124.2625   125.1500
2008 126.1625   127.2125   128.4000   129.3625
2009 130.1375   131.0250   131.7625   132.1000
2010 132.3375   132.5750   133.2500   134.6625
2011 136.2125   137.9000         NA         NA.
```

Durch Anwendung des Filters c(1/24, rep(1/12, 11), 1/24) lässt sich analog der zentrierte gleitende Durchschnitt $\bar{x}_t^{12}$ bei Monatsdaten berechnen.

Die grafische Darstellung der Indexreihe der Löhne und Gehälter je Beschäftigten mit den gleitenden Durchschnitten (Abb. 2.5) erfolgt wiederum mit den Funktionen plot und lines und die Umrandung mit der box-Funktion:

```
>plot(Lohn, main="Löhne und Gehältermit gleitenden Durchschnitten", xlab="",ylab="Löhne und
Gehälter",lwd=3, cex.main = 1.2, font.main=1)
>lines(Lohn.gl4, col="red", lwd=3)
> box(which="figure")
```

Abbildung 2.5: Zeitreihendiagramm der Löhne und Gehälter mit zentrierten 4-gliedrigem gleitenden Durchschnitten



Die zur Zerlegung von Zeitreihen konzipierte R-Funktion decompose berechnet den hierzu benötigten Trend ebenfalls nach der hier für eine gerade Ordnung vorgestellten Methode der gleitenden Durchschnitte. Die Trendkomponente ist nach Ausführung des decompose-Befehls unter dem Namen trend verfügbar:

```
>Lohn.dec = decompose(Lohn)
```

```
>Lohn.dec$trend
        Qtr1     Qtr2     Qtr3     Qtr4
2007    NA       NA       124.2625 125.1500
2008 126.1625 127.2125 128.4000 129.3625
2009 130.1375 131.0250 131.7625 132.1000
2010 132.3375 132.5750 133.2500 134.6625
2011 136.2125 137.9000    NA       NA
```

Übungsaufgaben

2.1.8 Definieren Sie den Vektor y aus dem Vektor x (s. Aufg. 1.4.1) unter Eliminierung der vierten, achten und zwölften Komponente!

2.1.9 Transformieren Sie den Vektor y in die Zeitreihe zeity mit der Datierung 2003 – 2011!

2.1.10 Glätten Sie die Zeitreihe zeity mit der Methode der gleitenden Durchschnitte der Ordnung 3 und speichern Sie die geglättete Reihe als Zeitreihenobjekt zeity.gl3!

2.1.11 Stellen Sie die originäre Zeitreihe zeity zusammen mit der geglätteten Reihe zeity.gl3 in einem Zeitreihendiagramm dar!

2.1.12 Glätten Sie die Zeitreihe zeity mit der Methode der gleitenden Durchschnitte der Ordnung 4 mit einer geeigneten Zentrierung und speichern Sie die geglättete Reihe als Zeitreihenobjekt zeity.gl4!

2.1.13 Stellen Sie die originäre Zeitreihe zeity zusammen mit der geglätteten Reihe zeity.gl4 in einem Zeitreihendiagramm dar!

2.2 Saisonkomponente und Saisonbereinigung

2.2.1 Phasendurchschnittsverfahren

Sind die Trendwerte m_t mit einer Trendfunktion oder der Methode der gleitenden Durchschnitte berechnet, lassen sich die trendbereinigten Zeitreihenwerte bestimmen. Bei additiver Verknüpfung der Komponenten entsprechen die trendbereinigten Werte als Differenzen der Zeitreihenwerte x_t und der Trendwerte m_t der um die Restkomponente u_t überlagerten Saisonkomponente s_j:[1]

(2.7) $d_t = x_t - m_t = s_j + u_t$.

Nach dem Phasendurchschnittsverfahren werden rohe Saisonziffern s_j^* für die Jahresabschnitte (Phase) durch Mittelung der entsprechenden trendbereinigten Werte gebildet. Bezeichnet man das Jahr mit i und die Phase mit j, ergeben sich die rohen Saisonziffern aus

[1] Bei einer multiplikativen Verknüpfung ergeben sich die trendbereinigten Werte durch Division der Zeitreihen- durch die Trendwerte. Das Phasendurchschnittsverfahren kann dann analog hierauf angewendet. Alternativ kann die Berechnung der Saisonkomponente unter Verwendung des additiven Modells nach Logarithmenbildung vorgenommen werden (s. Kosfeld/Eckey/Türck, 2016, Deskriptive Statistik, S. 284-286).

(2.8) $s_j^* = \frac{1}{k_j} \sum_i d_{ij}$.

In (2.8) gibt k_j die Anzahl der Jahre an, für die die saisonbereinigten Werte verfügbar sind.

Die endgültige Saisonkomponente s_j wird unter Beachtung der Normierungsbedingung $\sum_j s_j = 0$ bestimmt. Hierzu wird das arithmetische Mittel $\bar{d}$ der rohen (unnormierten) Saisonziffern s_j^* von diesen subtrahiert:

(2.9) $s_j = s_j^* - \bar{d}$.

Die Saisonziffern s_j geben die normierte Sasionkomponente wieder. Bei $s_j>0$ ist die Phase j überdurchschnittlich stark ausgeprägt und $s_j<0$ indiziert eine unterdurchschnittliche Saison.

Die trendbereinigten Werte $\tilde{x}_t$ erhält man durch Subtraktion der Saisonkomponente s_j von den Zeitreihenwerten x_t:

(2.10) $\tilde{x}_t = x_t - s_j = x_{ij} - s_j$.

Als Beispiel soll hier die Saisonbereinigung für die Zeitreihe der Löhne und Gehälter je Beschäftigten vorgenommen werden. Aus Abbildung 1.4 geht hervor, dass die Amplitude der Saisonschwankungen keinen steigenden oder fallenden Trend besitzt, so dass das additive Modell der Komponentenzerlegung anwendbar ist.

Die Trendbereinigung der Zeitreihe Lohnerfolgt unter Verwendung der zentrierten 4-gliedrigen gleitenden Durchschnitte Lohn.gl4z:

```
>Lohn.dij = Lohn-Lohn.gl4
>Lohn.dij
        Qtr1    Qtr2    Qtr3    Qtr4
2007    NA      NA      -2.2625 13.6500
2008 -9.8625 -1.5125 -2.7000 14.1375
2009 -9.0375 -2.4250 -2.7625 15.2000
2010 -9.1375 -3.3750 -2.9500 13.2375
2011 -8.2125 -2.2000    NA      NA
```

Die detaillierte Berechnung der Saisonkomponente erweist sich als etwas aufwändig. Zunächst wird mit der Funktion cycle die Phase (Quartal) der einzelnen Werte der Zeitreihe Lohn identifiziert und mit der Funktion factor als Faktor definiert:

```
>cycle(Lohn)
     Qtr1 Qtr2 Qtr3 Qtr4
2007  1    2    3    4
2008  1    2    3    4
2009  1    2    3    4
2010  1    2    3    4
2011  1    2    3    4
```

```
>fcycle = factor(cycle(Lohn))
>fcycle
[1] 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
Levels: 1 2 3 4
```

Anschließend werden hieraus vier Saison-Dummies d1, d2, d3 und d4 gebildet:

```
> d1 = fcycle==1
> d1
 [1]  TRUE FALSE FALSEFALSE  TRUE FALSE FALSEFALSE  TRUE FALSE FALSEFALSE  TRUE FALSE FALSEFALSE  TRUE
FALSE FALSEFALSE
> d1 = as.numeric(d1)
> d1
[1] 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0
> d2 = as.numeric(fcycle==2)
> d2
 [1] 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0
> d3 = as.numeric(fcycle==3)

>d3
 [1] 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0
> d4 = as.numeric(fcycle==4)
> d4
 [1] 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
```

Mit den Saison-Dummies kann für jede Phase die Anzahl der verfügbaren trendbereinigten Werte Lohn.dij ermittelt werden:

```
> k1 = length(na.omit(d1[d1*Lohn.dij!=0]))
> k1
[1] 4
> k2 = length(na.omit(d2[d2*Lohn.dij!=0]))
>k2
[1] 4
> k3 = length(na.omit(d3[d3*Lohn.dij!=0]))
> k3
[1] 4
> k4 = length(na.omit(d4[d4*Lohn.dij!=0]))
> k4
[1] 4
```

Damit können die rohen Saisonziffern s1_, s2_, s3_ und s4_ berechnet werden:

```
> s1_= sum(Lohn.dij*d1, na.rm=TRUE)/k1
> s1_
[1] -9.0625
> s2_= sum(Lohn.dij*d2, na.rm=TRUE)/k2
```

```
> s2_
[1] -2.378125
> s3_= sum(Lohn.dij*d3, na.rm=TRUE)/k3
> s3_
[1] -2.66875
> s4_ = sum(Lohn.dij*d4, na.rm=TRUE)/k4
> s4_
[1] 14.05625
```

Mit dem arithmetischen Mittel dm der rohen Saisonziffern,

```
>dm = (s1_+s2_+s3_+s4_)/4
>dm
[1] -0.01328125,
```

ergibt sich schließlich die normierte Saisonkomponente:

```
> s1 = s1_ - dm
> s1
[1] -9.049219
> s2 = s2_ - dm
> s2
[1] -2.364844
> s3 = s3_ - dm
> s3
[1] -2.655469
> s4 = s4_ - dm
> s4
[1] 14.06953
```

Die additive Saisonkomponente ist aber auch nach Ausführung des Befehls

```
>Lohn.dec = decompose(Lohn)
```

unter Lohn.dec$seasfür alle Perioden der Zeitreihe verfügbar:

```
>Lohn.dec$seas
Qtr1      Qtr2      Qtr3      Qtr4
2007 -9.049219 -2.364844 -2.655469 14.069531
2008 -9.049219 -2.364844 -2.655469 14.069531
2009 -9.049219 -2.364844 -2.655469 14.069531
2010 -9.049219 -2.364844 -2.655469 14.069531
2011 -9.049219 -2.364844 -2.655469 14.069531
```

Standardmäßig wird die decompose-Funktion mit dem Argument type=additive ausgeführt. Im Falle des multiplikativen Komponentenmodells ist type=multiplicative zu spezifizieren.

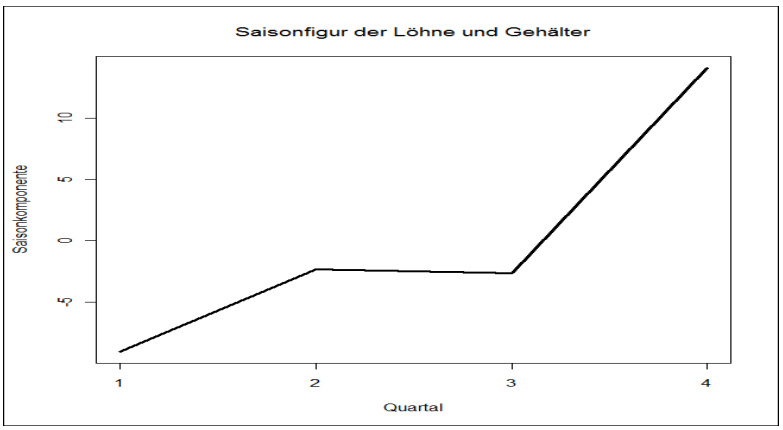
Unter dem Suffix $figure ist die generelle Saisonfigur gespeichert:

```
> Lohn.fig = Lohn.dec$figure
>Lohn.fig
[1] -9.049219 -2.364844 -2.655469 14.069531
```

Die Saisonfigur der Löhne und Gehälter lässt sich mit dem plot- und axis-Befehl grafisch darstellen (Abbildung 2.6):

```
> plot(Lohn.fig, main="Saisonfigur der Löhneund Gehälter", xlab="Quartal", ylab="Saisonkomponente",
lwd=3, cex.main=1.2, font.main=1, type="l", xaxt="n")
>axis(side=1, at=c(1,2,3,4))
>box(which="figure")
```

Abbildung 2.6: Saisonfigur der Löhne und Gehälter



Die Residuen können unter dem Variablennamen Lohn.dec$random abgerufen werden:

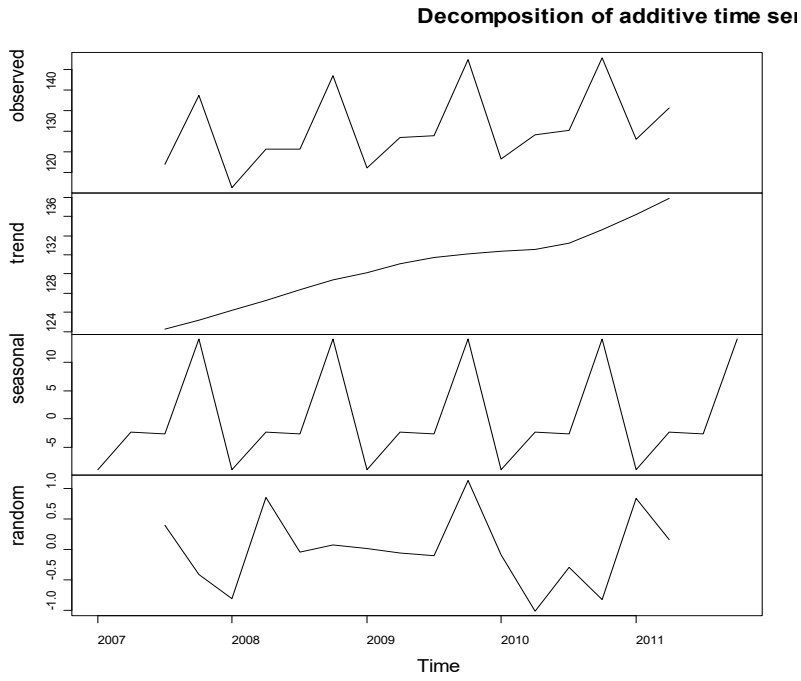
```
>Lohn.dec$random
          Qtr1        Qtr2        Qtr3        Qtr4
2007        NA          NA  0.39296875 -0.41953125
2008 -0.81328125  0.85234375 -0.04453125  0.06796875
2009  0.01171875 -0.06015625 -0.10703125  1.13046875
2010 -0.08828125 -1.01015625 -0.29453125 -0.83203125
2011  0.83671875  0.16484375          NA          NA
```

Mit dem Befehl

```
>plot(Lohn.dec)
```

erstellt R eine kompakte Grafik der Zeitreihe Lohn mit ihren durch die decompose-Funktion bestimmten Komponenten (Abbildung 2.7).

Abbildung 2.7: Zerlegung der Zeitreihe der Löhne und Gehälter



Die Löhne und Gehälter lassen sich bei detaillierter Berechnung unter Verwendung der Dummy-Variablen d_i und der Saisonkomponente s_i von saisonalen Effekten bereinigen:

```
> Lohn.sb = Lohn - d1*s1 - d2*s2 - d3*s3 - d4*s4
> Lohn.sb
         Qtr1     Qtr2     Qtr3     Qtr4
2007 122.6492 123.6648 124.6555 124.7305
2008 125.3492 128.0648 128.3555 129.4305
2009 130.1492 130.9648 131.6555 133.2305
2010 132.2492 131.5648 132.9555 133.8305
2011 137.0492 138.0648 138.8555 141.4305
```

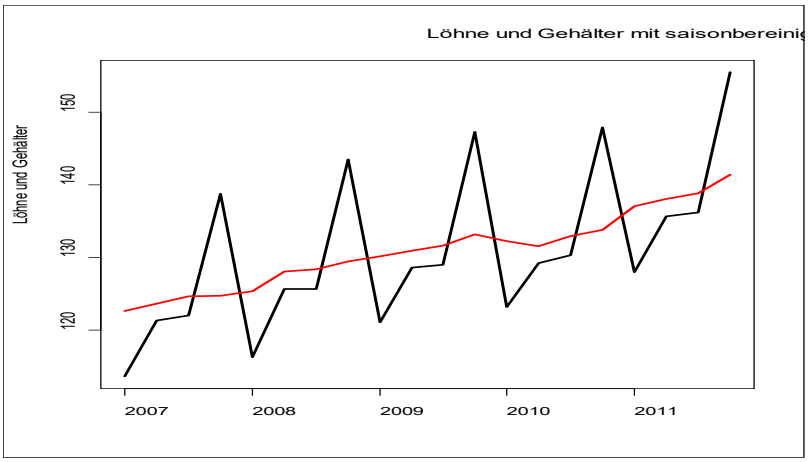
Unter Verwendung des decompose-Befehls erhält man die saisonbereinigte Lohnreihe einfacher aus

```
> Lohn.sb1 = Lohn - Lohn.dec$seas
> Lohn.sb1
         Qtr1     Qtr2     Qtr3     Qtr4
2007 122.6492 123.6648 124.6555 124.7305
2008 125.3492 128.0648 128.3555 129.4305
2009 130.1492 130.9648 131.6555 133.2305
2010 132.2492 131.5648 132.9555 133.8305
2011 137.0492 138.0648 138.8555 141.4305.
```

Schließlich wird die Zeitreihe Lohn zusammen mit der saisonbereinigten Zeitreihe Lohn.bs in einem Zeitreihendiagramm grafisch dargestellt (Abbildung 2.8):

```
>plot(Lohn, main="Löhne und Gehältermit saisonbereinigter Zeitreihe", xlab="",ylab="Löhne und Gehälter",lwd=3, cex.main = 1.2, font.main=1)
>lines(Lohn.sb, col="red", lwd=3)
> box(which="figure")
```

Abbildung 2.8: Zeitreihendiagramm der Löhne und Gehälter mit Saisonbereinigung



Übungsaufgaben

2.2.1 Zerlegen Sie die Zeitreihe zeitx unter Verwendung des additiven Komponentenmodells und plotten Sie sie mit der Trend-, Saison- und Restkomponente unter Verwendung des decompose-Befehls!

2.2.2 Zeichnen Sie die Saisonfigur der Zeitreihe zeitx mit einer geeigneten Beschriftung und interpretieren Sie sie!

2.2.3 Bilden Sie die saisonbereinigte Zeitreihe zeitx.sb und stellen Sie sie zusammen mit der originären Zeitreihe zeitx in einem Zeitreihendiagramm dar!

2.2.2 Regressionsverfahren

Das Phasendurchschnittsverfahren ist ähnlich wie die Methode der gleitenden Durchschnitte eine Rechenvorschrift zur Bestimmung der Saisonkomponente. Wie der Trend kann die Saisonkomponente jedoch auch in einem Regressionsmodell be-stimmt werden. Hierzu werden entsprechend der Anzahl der Phasen Saisondummies d_j, j=1,2,…,p, eingeführt:

$$(2.11)\quad d_j = \begin{cases} 1, \text{ falls Phase } j \\ 0 \text{ sonst} \end{cases}.$$

Die Regressionskoeffizienten der Saisondummies, $ß_1$, $ß_2$, …, $ß_p$, geben die rohen Saisonziffern wieder, die nach einer Normierung vergleichbar mit den Werten des Phasendurchschnittsverfahrens sind. Falls die Zeitreihe x_t trendbehaftet ist, kann der Trend simultan mit der Saisonkomponente in einem Regressionsmodell geschätzt werden.

Der erste Regressionsansatz bezieht ein absolutes Glied in das Modell ein. In diesem Fall dürfen nur p-1 Saisondummies in das Regressionsmodell einbezogen wer-den, da ansonsten die Summe der Saisondummies in jeder Periode gleich 1 wäre und damit dem Wert der Scheinvariablen für das absoluten Glieds entsprechen wür-de. Aufgrund der durch die lineare Abhängigkeit der Spalten der Beobachtungsma-trix definitorisch hervorgerufene Multikollinearität ließen sich die Regressionskoeffi-zienten nicht mehr berechnen. Die ausgeschlossene Saisondummy gibt die Refe-renzkategorie an, d.h. die geschätzten Saisonziffern geben die saisonalen Effekte relativ zu dieser Referenzsaison an.

Unterstellt man einen linearen Trend und wählt die erste Phase des Jahres als Referenz, lautet das Regressionsmodell

(2.12) $x_t = \beta_1 + \beta_2 \cdot d_2 + ... + \beta_p \cdot d_p + \beta_{p+1} \cdot t + u_t$,t=1,2,…,n.

Um das Trend-Saison-Modell (2.12) mit der Kleinst-Quadrate-Methode zu schätzen, ziehen wir die im vorherigen Abschnitt analysierte Zeitreihe Lohn heran. Vorab muss die vormals verwendete Trendvariable t neu definiert werden, da die hier verwendete Zeitreihe länger ist als der bei der Vorstellung der Trendfunktion analysierte Datensatz:

```
> n = length(Lohn)
>n
[1] 20
> t = seq(from=1, to=n)
> t
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

Die Saisondummies können dagegen aus dem Phasendurchschnittsverfahren über-nommen werden. Das erste Quartal dient als Referenzphase. Die übrigen drei Sai-sondummies stehen in der lm-Funktion als erklärende Variablen zusammen mit der Trendvariablen auf der rechten Seite der Gleichung und werden durch das Plus-zeichen (+) miteinander verknüpft:

```
> Lohn.lts1 = lm(Lohn~d2+d3+d4+t)
> Lohn.lts1

Call:
lm(formula = Lohn ~ d2 + d3 + d4 + t)

Coefficients:
(Intercept)       d2       d3       d4        t
  112.6325   6.7925   6.4650  23.5575   0.8675
```

Die Regressionskoeffizienten der Dummyvariablen d2, d3 und d4 lassen sich als rohe Saisonziffern interpretieren. Die unnormierte Saisonkomponente nimmt implizit für das erste Quartal (Referenzquartal) den Wert 0 an. Zum Zwecke einer Normierung ist dieser Wert bei der Berechnung des Durchschnitts der rohen Saisonziffern zu berücksichtigen:

```
>dm = (0+Lohn.lts1$coeff[2]+Lohn.lts1$coeff[3]+Lohn.lts1$coeff[4])/4
```

```
>dm
   d2
9.20375
```

Subtrahiert man den Durchschnittswert dm von 9.21375 von den rohen Saisonziffern, ergibt sich die normierte Saisonkomponente:

```
> s1 = 0 - dm
> s1
    d2
-9.20375
> s2 = Lohn.lts1$coeff[2] - dm
> s2
    d2
-2.41125
> s3 = Lohn.lts1$coeff[3] - dm
> s3
    d3
-2.73875
> s4 = Lohn.lts1$coeff[4] - dm
> s4
    d4
14.35375
```

Ein Vergleich mit der normierten Saisonkomponente des Phasendurchschnittsverfahrens zeigt einen hohen Grad an Übereinstimmung. Völlig identisch ist die Saisonkomponente jedoch nicht, da bei dem Regressionsverfahren ein linearer Trend unterstellt worden ist, während das Phasendurchschnittsverfahren tendenziell eher von der Methode der gleitenden Durchschnitte Gebrauch macht.

Wenn alle Saisondummies in das Regressionsmodell einbezogen werden sollen, muss das absolute Glied zur Vermeidung von Multikollinearität ausgeschlossen wer-den. Der zweite Regressionsansatz ist dann bei linearem Trend durch das Modell

(2.13) $x_t = \beta_1 \cdot d_1 + ... + \beta_p \cdot d_p + \beta_{p+1} \cdot t + u_t$, t=1,2,…,n,

gegeben.

Bei der Kleinst-Quadrate-Schätzung mit der lm-Funktion muss mit 0 oder -1 auf der rechten Seiten der erklärenden Variablen angegeben werden, dass das absolute Glied unterdrückt werden soll:

```
> Lohn.lts2 = lm(Lohn~0+d1+d2+d3+d4+t)
> Lohn.lts2

Call:
lm(formula = Lohn ~ 0 + d1 + d2 + d3 + d4 + t)
```

```
Coefficients:
    d1        d2        d3        d4         t
112.6325  119.4250  119.0975  136.1900   0.8675
```

Erneut geben die Regressionskoeffizienten der Saisondummies die rohen Saisonziffern wieder. Ihr durchschnittlicher Wert lässt sich hier unter Verwendung des Vektor Lohn2.lts2$coeff unter Ausschluss des Trendkoeffizienten (= fünfte Komponente) mit der Funktion mean berechnen:

```
>dm = mean(Lohn.lts2$coeff[-5])
>dm
[1] 121.8363
```

Die Werte der Saisonkomponente ergeben sich wiederum durch Subtraktion ihres Durchschnittswerts von den rohen Saisonziffern:

```
> s1 = Lohn.lts2$coeff[1] – dm

> s1
    d1
-9.20375
> s2 = Lohn.lts2$coeff[2] - dm
> s2
    d2
-2.41125
> s3 = Lohn.lts2$coeff[3] - dm
> s3
    d3
-2.73875
> s4 = Lohn.lts2$coeff[4] - dm
> s4
d4
14.35375
```

Obwohl die rohen Saisonziffern zwischen den beiden Versionen des Regressionsverfahrens voneinander differieren, ist die normierte Saisonkomponente identisch.

Übungsaufgaben

2.2.4 Regressieren Sie die Zeitreihe zeitx auf die Saisondummies d2, d3 und d4 und der Trendvariablen t!

2.2.5 Bestimmen Sie die normierte Saisonkomponente aus den geschätzten Regressionskoeffizienten der Saisondummies d2, d3 und d4!

2.2.6 Interpretieren Sie den Regressionskoeffizienten der Trendvariablen t!

2.2.7 Berechnen Sie die normierte Saisonkomponente aus einer Regression der Zeitreihe zeitx auf die Saisondummies d1, d2, d3 und d4 und der Trendvariablen t!

2.3 Exponentielle Glättung

Die exponentielle Glättung ist ein Verfahren zur kurzfristigen Prognose von Zeit-reihen. Im einfachsten Fall, in dem eine Zeitreihe (x_t) um einen konstanten Wert schwankt, spricht man von einer **exponentiellen Glättung erster Ordnung**. Der Ein-Schritt-Prognosewert $\hat{x}_{t,1}$ lässt sich dann aus der Rekursionsformel

(2.14) $\hat{x}_{t,1} = \alpha \cdot x_t + (1-\alpha) \cdot \hat{x}_{t-1,1}$

bestimmen. Bei der Prognose für die Periode t+1 wird somit der aktuelle Wert x_t mit dem Faktor α und der vorherige Prognosewert $\hat{x}_{t-1,1}$ mit dem Faktor 1-α gewichtet. Je größer α ist, umso stärker geht der aktuelle Zeitreihenwert x_t in die Prognose ein. Innerhalb eines Prognosezeitraums wird die Prognose dann volatiler. Die vergangene Zeitreihenentwicklung wird dagegen bei einem kleinen Wert α stärker berücksichtigt. Dadurch entsteht ein starker Glättungseffekt.

Für den Glättungsparameter α ist ein Wert zwischen 0 und 1 zu wählen. Ein optimaler Wert für α kann aufgrund eines Gütekriteriums ermittelt werden. Häufig wird ein optimaler Wert des Glättungsparameters α aufgrund des mittleren quadratischen Fehlers (mean square error (MSE)) bestimmt.

Falls die Zeitreihe $\{x_t\}$ trendbehaftet ist, kommt die **exponentielle Glättung zweiter Ordnung (Holt-Verfahren)** zur Anwendung. Neben den Parameter α zur Glättung des Niveaus a_t der Zeitreihe tritt ein Parameter β zur Glättung des trendmäßigen Zuwachses b_t:

(2.15) $\hat{x}_{t,h} = a_t + b_t \cdot h$ (h-Schritt-Prognose)

mit

(2.16) $a_t = \alpha \cdot x_t + (1-\alpha) \cdot (a_{t-1} + b_{t-1})$

und

(2.17) $b_t = \beta \cdot (a_t - a_{t-1}) + (1-\beta) \cdot b_{t-1}$.

Ist die Zeitreihe zusätzlich durch saisonale Effekte geprägt, wird ein dritter Parameter γ zur Glättung der Saisonschwankungen $s_t$eingeführt. Die exponentielle Glättung unter Verwendung der drei Glättungsparameter ist als **Holt-Winters-Verfahren** bekannt:

(2.18) $\hat{x}_{t,1} = a_t + b_t \cdot h + s_{t+1-p}$ (h-Schritt-Prognose)

mit

(2.19) $a_t = \alpha \cdot (x_t - s_{t-p}) + (1-\alpha) \cdot (a_{t-1} + b_{t-1})$

(2.20) $b_t = \beta \cdot (a_t - a_{t-1}) + (1-\beta) \cdot b_{t-1}$

(2.21) $s_t = \gamma \cdot (x_t - a_t) + (1-\gamma) \cdot s_{t-p}$

Die Größe p gibt dabei die Anzahl der Jahresabschnitte (Phasen) an z.B. p=4 bei Quartalsdaten und p=12 bei Monatsdaten. Die Verfahren der exponentiellen Glättung erster und zweiter Ordnung ergeben sich als Spezialfälle des Holt-Winters-Verfahrens mit β=0 und γ=0 bzw. γ=0.

Als Beispiel für den Einsatz des Verfahrens der exponentiellen Glättung betrachten wir die Zeitreihe der von den Unternehmen der Schloss- und Beschlagindustrie im Zeitraum 2000-2007 hergestellten Drehkipp-Beschläge Beschlaege:

```
>Beschlaege = c(13317, 12930, 11643, 13098, 12223, 12161, 13230, 14065)
>Beschlaege
[1] 13317 12930 11643 13098 12223 12161 13230 14065
>Beschlaege = ts(Beschlaege, start = 2000, end = 2007, frequency =1)
>Beschlaege
Time Series:
Start = 2000
End = 2007
Frequency = 1
[1] 13317 12930 11643 13098 12223 12161 13230 14065.
```

Da die Werte tendenziell um einen konstanten Wert schwanken, erfolgt die Glättung der Zeitreihe Beschlaege mit der exponentiellen Glättung erster Ordnung. Hierzu sind die beiden Glättungsparameter beta und gamma bei dem Aufruf der Funktion HoltWinters gleich FALSE zu setzen:

```
>Beschlaege.es1 = HoltWinters(Beschlaege, beta=FALSE, gamma=FALSE)
> Beschlaege.es1
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = Beschlaege, beta = FALSE, gamma = FALSE)

Smoothing parameters:
alpha: 0.1656433
beta : FALSE
gamma: FALSE

Coefficients:
[,1]
a 13038.99
```

R ermittelt aufgrund des MSE-Kriteriums als optimalen Gättungsparameter α den Wert 0.166. Der konstante Prognosewert der Zeitreihe Beschlaege beträgt 13038.99 (=a).
Die exponentiell geglätteten Werte sind nach Ausführung der Funktion HoltWinters als Attribut unter dem Namen fitted verfügbar:

```
> Beschlaege.es1f = Beschlaege.es1$fitted
> Beschlaege.es1f
Time Series:
Start = 2001
End = 2007
Frequency = 1
xhatlevel
```

```
2001 13317.00 13317.00
2002 13252.90 13252.90
2003 12986.23 12986.23
2004 13004.74 13004.74
2005 12875.25 12875.25
2006 12756.94 12756.94
2007 12835.30 12835.30
```

Das Attribut fitted enthält somit die Zeitreihen xhat und level, die bei der exponentiellen Glättung erster Ordnung identisch sind. Um die Reihe der geglätteten Werte um Prognosewerte zu erweitern, extrahieren wir die Werte der Reihe xhat und definieren hiermit die Zeitreihe Beschlaege.es1f:

```
>Beschlaege.es1f = Beschlaege.es1$fitted[,1]
> Beschlaege.es1f
Time Series:
Start = 2001
End = 2007
Frequency = 1
[1] 13317.00 13252.90 12986.23 13004.74 12875.25 12756.94 12835.30
```

Die Prognosewerte für den Zeitraum 2008-2011 werden mit dem predict-Kommando erzeugt und unter Beschlaege.es1p gespeichert:

```
> Beschlaege.es1p = predict(Beschlaege.es1, n.ahead=4)
> Beschlaege.es1p
Time Series:
Start = 2008
End = 2011
Frequency = 1
fit
[1,] 13038.99
[2,] 13038.99
[3,] 13038.99
[4,] 13038.99
```

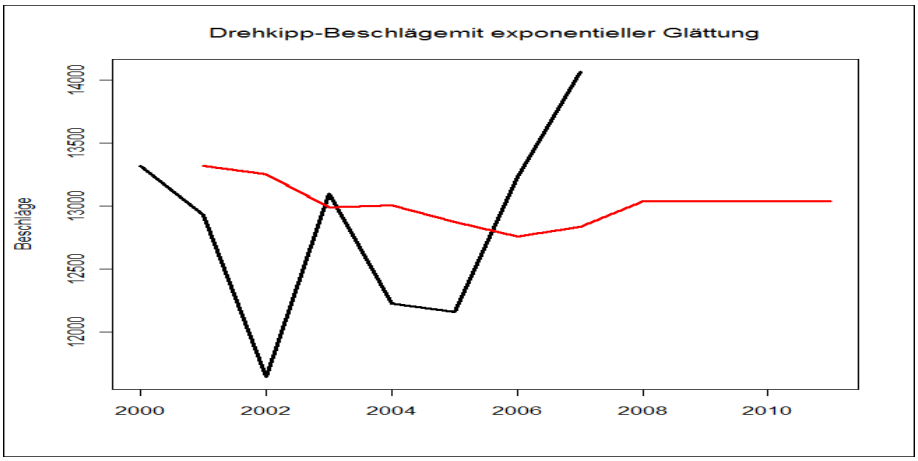
Schließlich werden die beiden Zeitreihen Beschlaege.es1f und Beschlaege.es1p miteinander zu einer einheitlichen Zeitreihe Beschlaege1fp verbunden:

```
> Beschlaege.es1fp = ts(c(Beschlaege.es1f, Beschlaege.es1p), start=2001, end=2011, frequency=1)
> Beschlaege.es1fp
Time Series:
Start = 2001
End = 2011
Frequency = 1
 [1] 13317.00 13252.90 12986.23 13004.74 12875.25 12756.94 12835.30 13038.99 13038.99 13038.99
13038.99
```

Die originäre Zeitreihe Beschlaege und die kombinierte geglättete und prognostizierte Zeitreihe Beschlaege.es1fp werden in einem Zeitreihendiagramm dargestellt (Abb. 2.9):

```
>plot(Beschlaege, main="Drehkipp-Beschlägemit exponentieller Glättung",
xlab="",ylab="Beschläge",xlim = c(2000, 2011),lwd=3, cex.main = 1.2, font.main=1)
>lines(Beschlaege.es1fp, col="red", lwd=3)
> box(which="figure")
```

Abbildung 2.9: Drehkipp-Beschläge mit exponentieller Glättung



Übungsaufgaben

2.3.1 Ermitteln Sie die optimalen Glättungsparameter α und βfür die Zeitreihe AEVG mit der Funktion HoltWinters und interpretieren Sie sie!

2.3.2 Was geben die mit der Funktion HoltWinters für die Zeitreihe AEVG ermittelten Koeffizienten a = 13530.257 und b = 885.846 an?

2.3.3 Für das vierte Quartal des Jahres 2008 werden nach Ausführung der Funktion HoltWinters mit gamma=FALSE für die Zeitreihe AEVG unter dem Attributnamen fitted die Startwerte xhat=110.6, level=108.6 und trend=2.0 ausgewiesen. Geben Sie die Werte xhat, level und trend für die ersten beiden Quartale des Jahres 2009 an!

2.3.4 Welche Prognosewerte der exponentiellen Glättung zweiter Ordnung weist R für die Zeitreihe AEVG im Zeitraum 2011/II-2011/IV aus?

2.3.5 Stellen Sie die originäre Zeitreihe AEVG zusammen mit den geglätteten und prognostizierten Werten in einem Zeitreihendiagramm dar!

2.4 Autokorrelation

2.4.1 Autokorrelationskoeffizient und Korrelogramm

Unter Autokorrelation ist die Korrelation von Werten einer Zeitreihe (x_t) zu verstehen. Folgen die Zeitreihenwerte unmittelbar aufeinander, spricht man von einer Autokorrelation 1. Ordnung. In diesem Fall wird die Korrelation zwischen den n-1 Wertepaaren (x_2, x_1), (x_3, x_2), …, (x_n, x_{n-1}) vergleichbar zum Pearsonschen Korrelationskoeffizienten gemessen:

$$(2.22)\quad r_1 = \frac{\sum_{t=2}^{n}(x_t - \bar{x})(x_{t-1} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}.$$

Die Kovarianz zwischen den aufeinander folgenden Werten wird auf die Varianz der Zeitreihe bezogen. Zur Berechnung des Autokorrelationskoeffizienten 1. Ordnung, r_1, wird somit der Mittelwert $\bar{x}$ der gesamten Zeitreihe (x_t) anstelle der Mittelwerte der beiden Teilreihen verwendet.

Der Autokorrelationskoeffizient 2. Ordnung, r_2, misst die Korrelation zwischen den n-2 Wertepaaren (x_3, x_1), (x_4, x_2), …, (x_n, x_{n-2}):

$$(2.23)\quad r_2 = \frac{\sum_{t=3}^{n}(x_t - \bar{x})(x_{t-2} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}.$$

Allgemein ist der Autokorrelationskoeffizient k-ter Ordnung, r_k, als Korrelation zwischen den n-k Wertepaaren (x_{k+1}, x_1), (x_{k+2}, x_2), …, (x_n, x_{n-k}) durch

$$(2.24)\quad r_k = \frac{\sum_{t=k+1}^{n}(x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2}$$

gegeben.

Um die in einer Zeitreihe vorhandenen Abhängigkeiten aufzudecken, trägt man die Autokorrelationskoeffizienten $r_0=1$, r_1, r_2, …, r_k in einem Koordinatensystem gegen den Lag k ab. Diese Form der grafischen Darstellung heißt **Korrelogramm**. In ein Korrelogramm werden häufig Konfidenzbänder eingezeichnet, die einen Signifikanztest der Autokorrelationskoeffizienten bieten. Mit einem reinen Zufallsprozess (White-Noise-Prozess) als Referenz lässt sich der Test des Autokorrelationskoeffizienten r_k auf Signifikanz auf dem 5%-Niveau approximativ unter Verwendung des 2σ-Intervalls

$$(2.25)\quad [-2/\sqrt{n};\ 2/\sqrt{n}]$$

durchführen. Bei der Interpretation ist jedoch zu berücksichtigen, dass auch im Falle eines reinen Zufallsprozesses bei einem von 20 Autokorrelationskoeffizienten ein signifikantes Ergebnis zu erwarten ist.

Im Fall einer Zeitreihe von Residuen, (u_t), mit dem Mittelwert $\bar{u} = 0$ erhält man den Autokorrelationskoeffizienten k-ter Ordnung aus

$$(2.26)\quad r_k = \frac{\sum_{t=k+1}^{n} u_t \cdot u_{t-k}}{\sum_{t=1}^{n} u_t^2}.$$

Wenn das Zeitreihenmodell, das der Ermittlung der Residuen zugrunde liegt, die Zeitreihe (x_t) adäquat beschreibt, weisen die Residuen keine systematischen Abhängkeiten mehr auf. In dem Korrelogramm werden die Autokorrelationen dann innerhalb der Konfidenzbänder liegen.

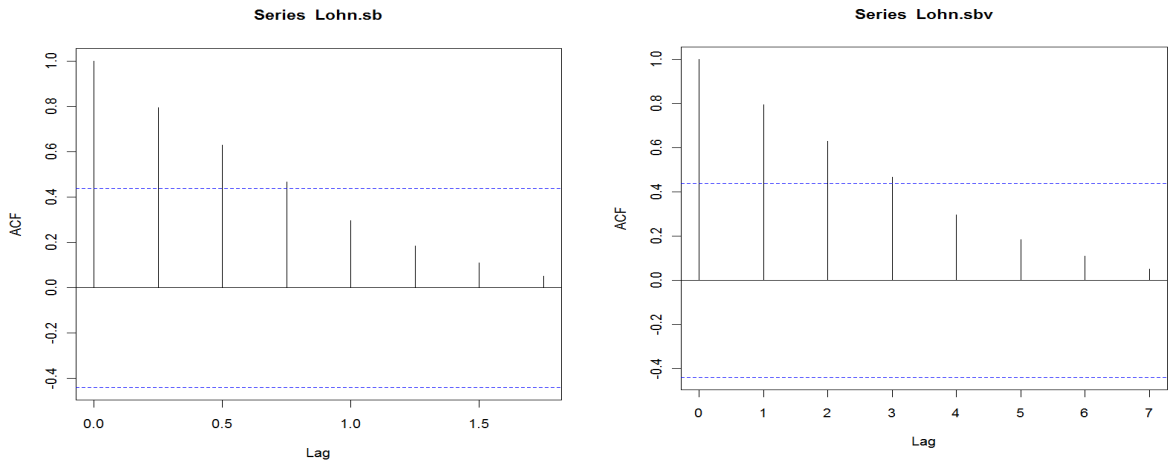
Zunächst sei das Korrelogramm einer trendbehafteten Zeitreihe betrachtet, dass in R durch die Funktion `acf` aufgerufen werden kann. Als Voreinstellung wird hierbei die Anzahl der Lags, für die die Autokorrelationen ausgewiesen werden, durch $10 \cdot \log_{10}(n)$ bestimmt. Bei n=20 Werten der saisonbereinigten Zeitreihe `Lohn.sb` würde das Korrelogramm Autokorrelationen bis zur Ordnung 13 ausweisen. Allerdings gehen in die Berechnung der letzten drei Autokorrelationen weniger als die Hälfte der geordneten Paare ein. Um eine hinreichende Basis zur Berechnung der Autokorrelationen zu haben, wird häufig gefordert, dass der maximale Lag r_{max} höchstens einem Drittel oder Viertel der Anzahl der Beobachtungswerte n entspricht. Setzt man den maximalen Lag `rmax` als ein Drittel der Beobachtungswerte fest,

```
>rmax = ceiling(n/3)
>rmax
[1] 7,
```

dann werden in dem Korrelogramm der Zeitreihe `Lohn.sb` neben der Eigenkorrelation r_0=1 noch 7 Autokorrelationskoeffizienten in Form von Stäben ausgewiesen (Abb. 2.10a):

```
>acf(Lohn.sb, lag.max=rmax).
```

Abbildung 2.10: Korrelogramm der saisonbereinigten Zeitreihe `Lohn.sb`



(a) mit Lags bezogen auf die Phasenzahl

(b) mit gewöhnlichen Lags 0,1,…,7

Abbildung 2.10a zeigt die bei trendbehafteten Zeitreihen zu erwartende langsame Abnahme der Autokorrelationen. Bei unterjährigen Zeitreihen werden in R jedoch auf der Abzisse nicht die gewöhnlichen Lags, sondern die auf die Phasenzahl bezogenen Lags abgetragen. Um auch in diesem Fall die übliche Darstellung des Korrelo-gramms zu erhalten (Abb. 2.10b), wird die Zeitreihe vor dem Aufruf der acf-Funktion in einen Vektor transformiert:

```
>Lohn.sbv = as.vector(Lohn.sb)
>acf(Lohn.sbv, lag.max=rmax).
```

Man erhält die Werte der Autokorrelationskoeffizienten bis zur Ordnung rmax, wenn bei der Ausführung der Funktion acf ein Objektname angegeben wird:

```
>Lohnsbv.kor = acf(Lohn.sbv, lag.max=rmax).
```

In diesem Fall sind die berechneten Autokorrelationskoeffizienten bis zur siebten Ordnung in dem Objekt Lohnsbv.korr gespeichert. Nach Eingabe des Objektnamens Lohnsbv.kor gibt das Programm R die Werte der Autokorrelationskoeffizienten aus:

```
>Lohnsbv.kor
Autocorrelations of series 'Lohn.sbv', by lag
   0     1     2     3     4     5     6     7
1.000 0.795 0.630 0.471 0.297 0.183 0.108 0.049
```

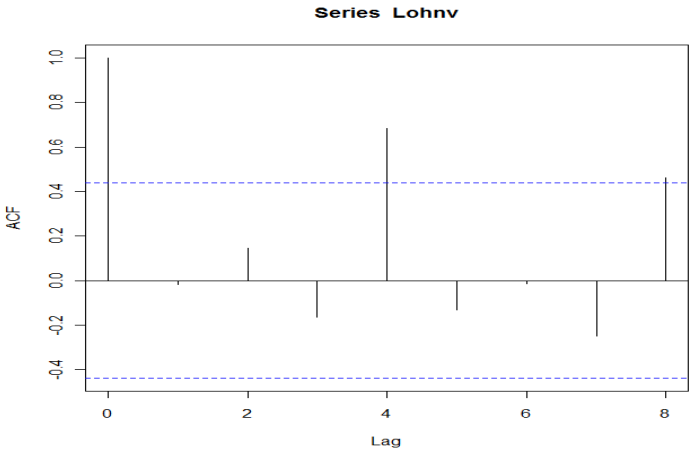
Sie sind aber auch als Attribut des acf-Objekts Lohnsbv.kor unter dem Suffix $acf verfügbar. Man beachte, dass R auch im Falle einer Zuweisung der acf-Funktion zum Objekt Lohnsbv.kor das Korrelogramm (2.10b) unmittelbar erstellt.

Bei einer saisonalen Zeitreihe findet man in der Regel die größten Autokorrelationen bei Lags von p, 2·p, 3·p,. … vor. Erhöht man den maximale Lag für die Zeitreihe Lohn auf 8, dann werden wegen p=4 in einem Korrelogramm Peaks bei den Lags 4 und 8 ersichtlich (Abb. 2.11):

```
>Lohnv = as.vector(Lohn)
>Lohnv.kor = acf(Lohnv, lag.max=rmax+1)
```

Abbildung (2.11) gibt das Korrelogramm der Zeitreihe Lohn nach Transformation in den Vektor Lohnv mit der gängigen Lagspezifikation wieder.

Abbildung 2.11: Korrelogramm der saisonalen Zeitreihe Lohn



Wie zu erwarten nehmen die saisonal bedingten Autokorrelationen mit zunehmender zeitlicher Distanz ab, was auch aus den Werten der Autokorrelationskoeffizienten hervorgeht:

```
>Lohnv.kor
Autocorrelations of series 'Lohnv', by lag
    0      1      2      3      4      5      6      7      8
 1.000 -0.023  0.140 -0.169  0.685 -0.132 -0.014 -0.250  0.463
```

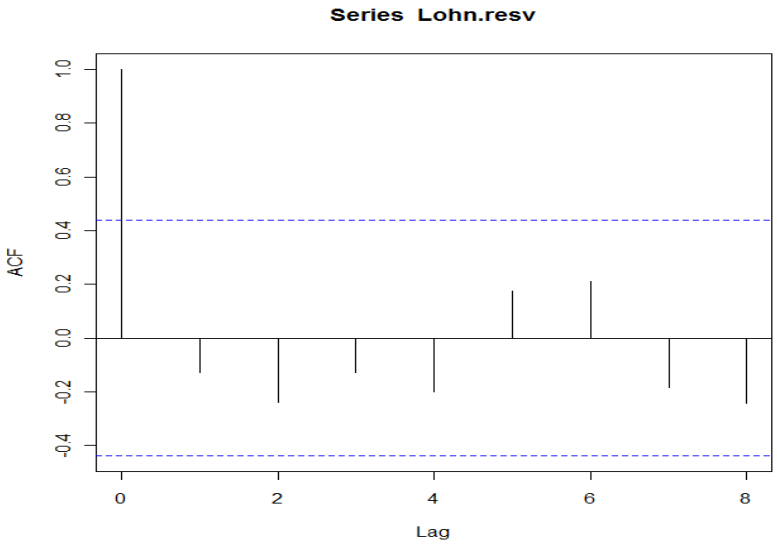
Die Residuen der Zeitreihe ergeben sich im additiven Komponentenmodell nach Subtraktion der Trendkomponente m_t und Saisonkomponente s_t von den originären Zeitreihenwerten x_t. Aufgrund der Anwendung der decompose-Funktion auf die Zeitreihe Lohn sind die Residuen als Attribut des decompose.ts-Objekts Lohn.dec verfügbar. Weist man sie dem Objekt Lohn.res zu und transformiert sie in einen Vektor Lohn.resv, erkennt man, dass aufgrund der gleitenden Durchschnittsbildung für die beiden ersten und letzten Perioden fehlende Werte auftreten:

```
> Lohn.res =Lohn.dec$random
>Lohn.resv = as.vector(Lohn.res)
>Lohn.resv
[1]        NA        NA     0.39296875 -0.41953125 -0.81328125  0.85234375 -0.04453125  0.06796875
0.01171875 -0.06015625 -0.10703125  1.13046875
[13] -0.08828125 -1.01015625 -0.29453125 -0.83203125  0.83671875  0.16484375        NA        NA
```

Dies muss bei der Berechnung der Autokorrelationskoeffizienten in der acf-Funktion durch die Spezifikation des Arguments na.action=na.pass angegeben werden:

```
> Lohnresv.kor = acf(Lohn.resv, na.action=na.pass, lag.max=rmax+1).
```

Abbildung 2.12: Korrelogramm der Residuen der Zeitreihe Lohn



Das mit dem Aufruf der acf-Funktion erstellte Korrelogramm der Residuen der Zeitreihe Lohn ist in Abbildung 2.12 wiedergegeben. Die Werte der Autokorrelationskoeffizienten bis zur achten Ordnung gibt R nach Eingabe des Objektnamens Lohnresv.kor aus:

```
>Lohnresv.kor
Autocorrelations of series 'Lohn.resv', by lag
   0      1      2      3      4      5      6      7      8
 1.000 -0.128 -0.238 -0.129 -0.201  0.173  0.211 -0.186 -0.243.
```

2.4.2 Portmanteau-Tests

Während die Signifikanz der Autokorrelationskoeffizienten unter Verwendung des Konfidenzintervalls (2.25) einzeln getestet wird, lassen sich mit Portmanteau-Tests Autokorrelationskoeffizienten bis zur k-ten Ordnung gleichzeitig auf Signifikanz testen. Hierzu sind der Box-Pierce- und Ljung-Box-Test konzipiert worden. Sie stellen wichtige Ergänzungen der Einzeltests dar, da hier je nach Wahl der maximalen Laglänge ein oder zwei Autokorrelationen allein zufallsbedingt signifikant sein können.

Beispielhaft seien die beiden Portmanteautests hier auf die saisonbereinigte Zeitreihe Lohn.sb angewendet. Standardmäßig wird mit der R-Funktion Box.test der Box-Pierce-Test angewendet. Zunächst wird der Test nur für den Autokorrelationskoeffi-zienten r_1 durchgeführt,

```
>Box.test(Lohn.sb, lag=1)
Box-Pierce test
data:  Lohn.sb
X-squared = 12.6573, df = 1, p-value = 0.0003741,
```

und anschließend für r_1 und r_2 gleichzeitig:

```
>Box.test(Lohn.sb, lag=2)
Box-Pierce test
data:  Lohn.sb
X-squared = 20.5741, df = 2, p-value = 3.407e-05.
```

Die unter X-squared ausgewiesenen Prüfgrößen des Tests, Q*(1) und Q*(2), sind beide hochsignifikant und bestätigen damit die aus dem Korrelogramm erhaltenen Ergebnisse der Signifikanztests.

Die Prüfgrößen Q(1) und Q(2) des asymptotisch äquivalenten Ljung-Box-Tests erweisen sich ebenfalls als hochsignifikant:

```
>Box.test(Lohn.sb, lag=1, type="Ljung-Box")
Box-Ljung test
data:  Lohn.sb
X-squared = 14.6558, df = 1, p-value = 0.0001290

>Box.test(Lohn.sb, lag=2, type="Ljung-Box")
Box-Ljung test
data:  Lohn.sb
X-squared = 24.3319, df = 2, p-value = 5.205e-06
```

Übungsaufgaben

2.4.1 Erstellen Sie ein Korrelogramm für die Zeitreihe kfz und interpretieren Sie es!

2.4.2 Definieren Sie eine eigene R-Funktion zur Berechnung des Autokorrelationskoeffizienten 1. Ordnung für die Zeitreihe kfz! Vergleichen Sie den aus dem Korrelogramm mit dem aus der eigenen R-Funktion erhaltenen Wert für r_1!

2.4.3 Wie lässt sich die in Aufg. 2.4.2 erstellte R-Funktion für Autokorrelationskoeffizienten k-ter Ordnung mit $k=1,2,..,k_{max}$ verallgemeinern?

2.4.4 Testen Sie die Prüfgrößen Q*(4) und Q*(8) des Box-Pierce-Testes für die Zeitreihe Lohn auf Signifikanz ($\alpha=0,05$)!

2.4.5 Lässt sich das in Aufg. 2.4.4 erzielte Ergebnis mit dem Ljung-Box-Test bestätigen?

3. Stochastische Prozesse und ARIMA-Modelle

3.1 White-Noise-Prozess und Random-Walk-Prozess

Um eine einheitliche Zufallszahlenreihe zu erhalten, wird der Pseudozufallszahlengenerator mit dem set.seed-Befehl initialisiert, d.h. auf einen festen Startwert gesetzt:

```
>set.seed(seed=147358)
```

Um einen White-Noise-Prozess zu simulieren, erzeugen wir eine Reihe von normalverteilten Zufallszahlen (Gauß-Prozess) der Länge 200,

```
>u = rnorm(n=200, mean = 0, sd = 1),
```

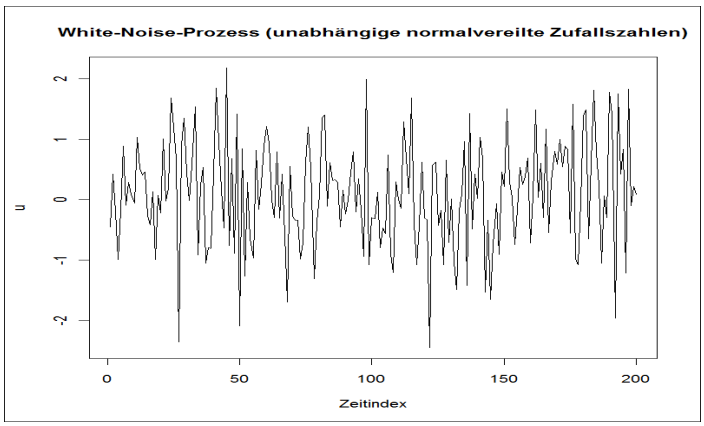
die wir anschließend als Zeitreihe definieren:

```
> u = as.ts(u, start=1, end=200)
>class(u)
[1] "ts"
>str(u)
Time-Series [1:200] from 1 to 200: -0.454 0.425 -0.199 -0.994 -0.179 ...
```

Der Plot des White-Noise-Prozesses ist in Abb. 3.1 wiedergegeben:

```
>plot(u, xlab="Zeitindex", main="White-Noise-Prozess (normalverteilte Zufallszahlen)")
> box(which="outer")
```

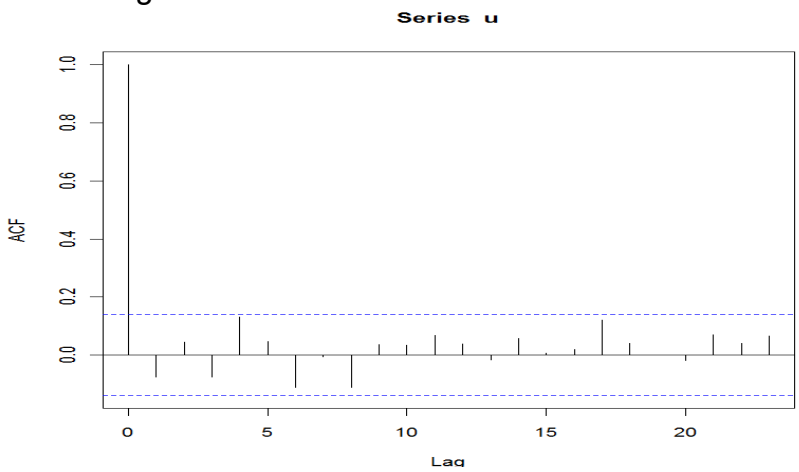
Abbildung 3.1: White-Noise-Prozess (unabhängige normalverteilte Zufallszahlen)



Das Korrelogramm des White-Noise-Prozesses wird mit der acf-Funktion aufgerufen:

```
>u.acf = acf(u)
>class(u.acf)
[1] "acf"
>str(u.acf)
List of 6
 $ acf   : num [1:24, 1, 1] 1 -0.0767 0.0438 -0.0753 0.13 ...
 $ type  :chr "correlation"
 $ n.used: int 200
$ lag   : num [1:24, 1, 1] 0 1 2 3 4 5 6 7 8 9 ...
 $ series: chr "u"
$ snames: NULL
- attr(*, "class")= chr "acf"
>u.acf
Autocorrelations of series 'u', by lag
0     1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18    19    20
21    22    23
 1.000 -0.077  0.044 -0.075  0.130  0.047 -0.112 -0.006 -0.112  0.036  0.034  0.068  0.039 -0.016  0.056  0.006
0.020  0.119  0.040 -0.001 -0.020  0.069  0.039  0.065
```

Abbildung 3.2: Korrelogramm eines White-Noise-Prozesses



Aus Abb. 3.2 geht hervor, dass keine der dargestellten Autokorrelationen 1. bis 24. Ordnung auf dem 5%-Niveau signifikant ist, da sie innerhalb des 2σ-Intervalls liegen. Die Eigenschaft der Unkorreliertheit des White-Noise-Prozesses spiegelt sich somit in der erzeugten Zufallsreihe u wider.

Zur Bildung des Lags1. Ordnung der Zufallsreihe u innerhalb des gegebenen Zeitraums 1 bis 200 kann wie folgt vorgegangen werden:

```
> u1 = ts(c(NA, lag(u,-1)), start=1, end=200)
>str(u1)
 Time-Series [1:200] from 1 to 200: NA -0.454 0.425 -0.199 -0.994 ...
```

Um einen Random Walk aus der Zufallsreihe u zu erzeugen, wird zunächst ein Zeitindex t bestimmt:

```
> t = 2:200
>class(t)
[1] "integer"
>str(t)
int [1:199] 2 3 4 5 6 7 8 9 10 11 ...
```

Der Random Walk ohne Drift,

(3.1) $x_t = x_{t-1} + u_t$,

wird dann nach Definition des Vektors x,

```
> x = rep(0, 200)
>class(x)
[1] "numeric"
>str(x)
num [1:200] 0 0 0 0 0 0 0 0 0 0 ...
```

und Setzen des Anfangswerts

```
>x[1] = u[1]
```

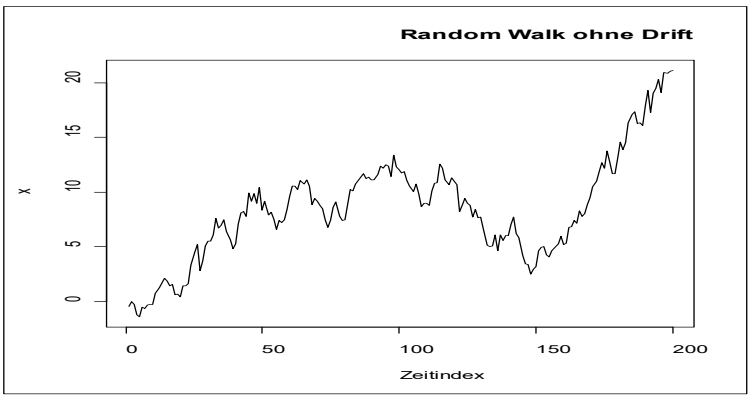
unter Verwendung der for-Schleife generiert:

```
>for (i in t) {x[i] = x[i-1] + u[i]}
head (x)
[1] -0.45412248 -0.02895118 -0.22754817 -1.22196955 -1.40048287 -0.51608999

> x = as.ts(x)
>class(x)
[1] "ts"
> str(x)
Time-Series [1:200] from 1 to 200: -0.454 -0.029 -0.228 -1.222 -1.4 ...
```

Der Random Walk ohne Drift, x, ist in Abbildung 3.3 dargestellt:

```
>plot(x, xlab="Zeitindex", main="Random Walk ohne Drift")
> box(which="outer").
```

Abbildung 3.3: Random Walk ohne Drift



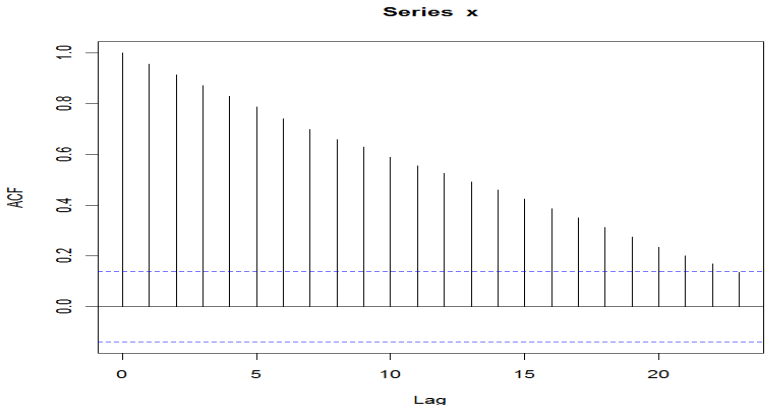
Das Korrelogramm des Random Walk x wird mit der acf-Funktion aufgerufen:

```
>x.acf = acf(x)
>class(x.acf)
[1] "acf"
>str(x.acf)
List of 6
 $ acf   : num [1:24, 1, 1] 1 0.956 0.915 0.873 0.83 ...
 $ type  :chr "correlation"
 $ n.used: int 200
$ lag   : num [1:24, 1, 1] 0 1 2 3 4 5 6 7 8 9 ...
 $ series: chr "x"
 $ snames: NULL
 - attr(*, "class")= chr "acf".
```

Da die berechneten Autokorrelationen beim Aufruf der acf-Funktion dem Objekt x.acf zugewiesen wurden, können sie ebenfalls ausgegeben werden:

```
>x.acf
Autocorrelations of series ‘x’, by lag
0    1    2    3    4    5    6    7    8    9    10   11   12   13   14   15   16   17   18   19   20   21   22   23
1.000 0.956 0.915 0.873 0.830 0.787 0.740 0.699 0.660 0.629 0.590 0.556 0.525 0.493 0.461 0.424 0.386
0.350 0.312 0.274 0.234 0.201 0.168 0.134
```

Abbildung 3.4: Korrelogramm eines Random Walk ohne Drift



Das in Abbildung 3.4 dargestellte Korrelogramm zeigt den typischen Verlauf einer trendbehafteten Zeitreihe. Aufgrund des in Abb. 3.3 wiedergegebenen Zeitreihendiagramms könnte man allenfalls für einzelne Teilperioden deterministische Trends vermuten. Die gesamte Zeitreihe wird jedoch aufgrund des zugrundeliegenden stochastischen Prozesses durch einen stochastischen Trend erzeugt.

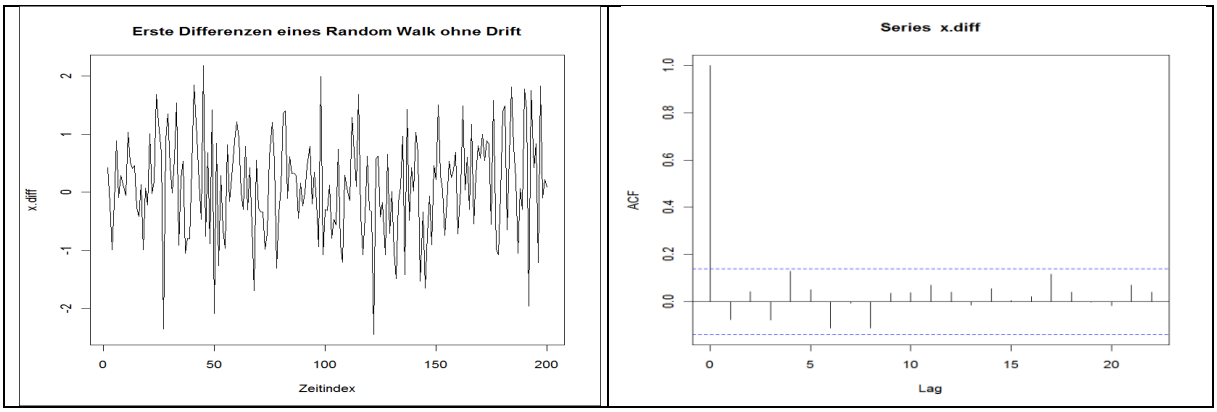
Differenziert man den Random Walkx, $\Delta x_t = x_t - x_{t-1}$, so ergibt sich wiederum ein White-Noise-Prozess,

```
>x.diff = diff(x)
>class(x.diff)
[1] "ts"
>str(x.diff)
 Time-Series [1:199] from 2 to 200: 0.425 -0.199 -0.994 -0.179 0.884 ...
>str(u)
Time-Series [1:200] from 1 to 200: -0.454 0.425 -0.199 -0.994 -0.179 ...,
```

dessen Verlauf zusammen mit dem zugehörigen Korrelogramm in Tab. 3.1 darstellt ist.

Tabelle 3.1: Erste Differenzen eines Random Walk ohne Drift

Zeitreihendiagramm	Korrelogramm
>plot(x.diff, xlab="Zeitindex", main="Erste Differenzen eines Random Walk ohne Drift") > box(which="outer")	>delx.acf = acf(x.diff) >delx.acf Autocorrelations of series ‘x.diff’, by lag 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 1.000 -0.076 0.043 -0.080 0.129 0.050 -0.113 - 0.006 -0.113 0.036 0.038 0.069 0.040 -0.015 0.055 0.004 0.020 0.116 0.040 -0.002 -0.017 0.069 0.040



Der Random Walkmit Drift,

(3.2) $x_t = b + x_{t-1} + u_t$,

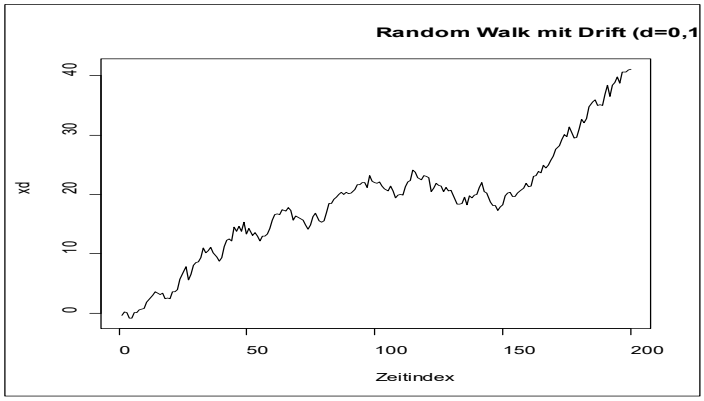
wird kann wie der Random Walk ohne Drift unter zusätzlicher Einbeziehung des Driftparameters b generiert werden:

```
> xd = rep(0, 200)
> xd[1] = 0.1 + u[1]
>for (i in t) {xd[i] = 0.1 + xd[i-1] + u[i]}
> head(xd)
[1] -0.35412248  0.17104882  0.07245183 -0.82196955 -0.90048287  0.08391001
> xd = as.ts(xd)
```

Der Driftparameter b ist hier gleich 0,1 gesetzt worden. Die Realisationen des Random Walk mit Drift, xd, sind in Abbildung 3.5 dargestellt:

```
> plot(xd, xlab="Zeitindex", main="Random Walk mit Drift (d=0,1)")
> box(which="outer").
```

Abbildung 3.5: Random Walk mit Drift (d=0,1)

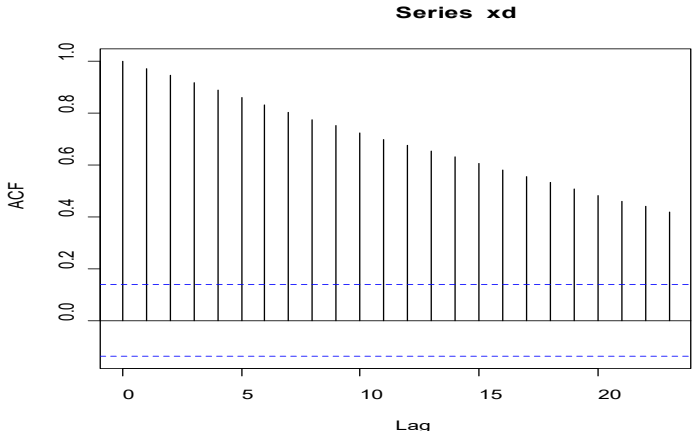


Das Korrelogrammder Zeitreihe xd wird mit der acf-Funktion erstellt (Abb. 3.6). Hierbei werden die Autokorrelationskoeffizienten in dem Objekt xd.acf gespeichert:

```
> xd.acf = acf(xd)
```

```
> xd.acf
Autocorrelations of series 'xd', by lag
0     1     2     3     4     5     6     7     8     9     10    11    12
1.000 0.971 0.944 0.916 0.887 0.859 0.829 0.801 0.774 0.750 0.723 0.699 0.676
  13    14    15    16    17    18    19    20    21    22    23
0.653 0.630 0.605 0.580 0.556 0.531 0.507 0.482 0.460 0.438 0.417
```

Abbildung 3.6: Korrelogramm eines Random Walk mit Drift (d=0,1)



Die Autokorrelationsstrukturen des Random Walk ohne und mit Drift ähneln sich. In beiden Fällen fallen die Autokorrelationskoeffzienten nur langsam ab und sind bis zu Lags größer als 20 signifikant. Allerdings liegen die Autokorrelationskoeffizienten des Random Walks mit Drift über denen des Random Walk ohne Drift.

Übungsaufgaben

3.1.1 Verwenden Sie einen `seed` von 426793 und erzeugen Sie eine Vektor standardnormalverteilter Zufallsvariablen der Länge 120! Interpretieren Sie das Zeitreihendiagramm des Gauß-Prozesses!

3.1.2 Spiegelt das Korrelogramm der Realisationen des Gauß-Prozesses (s. Aufg. 3.1.1) die Eigenschaften eines White-Noise-Prozesses wider?

3.1.3 Erzeugen Sie

- einen linearen Trend x.lt = 1:220 und
- einen Random-Walk `xd` mit dem Driftparameter 0.2 und `xd[0] = 0` mit der in Aufg. 3.1.1 generierten White-Noise-Prozess!

Vergleichen Sie die Zeitreihendiagramme und Korrelogramme der beiden Prozesse und interpretieren Sie die Unterschiede und Gemeinsamkeiten!

3.2 AR-Prozesse und MA-Prozesse

Ein AR(p)-Prozess steht abkürzend für einen autoregressiven Prozess p-ter Ordnung, bei dem der Wert x_t durch p zeitverzögerte Werte x_{t-1}, x_{t-2}, …, x_{t-p} erklärt wird:

(3.3) $x_t = \phi_1 \cdot x_{t-1} + \phi_2 \cdot x_{t-2} + ... + \phi_p \cdot x_{t-p} + u_t$

Für die Störvariable u_t wird dabei ein reiner Zufallsprozess (White-Noise-Prozess) angenommen. Die Koeffizienten ϕ_1, ϕ_2, …, ϕ_p heißen autoregressive Parameter. Im Folgenden werden AR(1)- und AR(2)-Prozesse mit R simuliert. Der AR(1)-Prozess ist durch die Autoregression 1. Ordnung,

(3.4) $x_t = \phi_1 \cdot x_{t-1} + u_t$

gegeben.

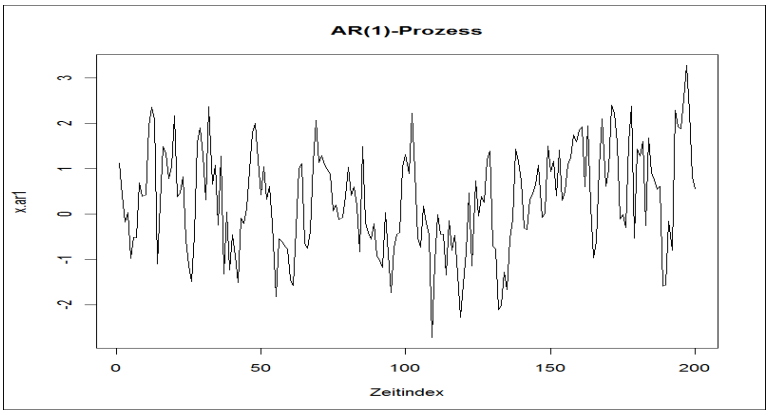
Autoregressive Prozesse könnten im Prinzip wie Random-Walk-Prozesse simuliert werden. Das Programm R stellt hierzu jedoch mit arima.sim eine spezielle Funktion zur Verfügung. Sie wird hier zur Erzeugung eines AR(1)-Prozesses mit dem autoregressiven Parameter ϕ_1 von 0,6 nach Initialisierung des Zufallszahlengenerators eingesetzt:

```
> set.seed(seed=147358)
> x.ar1 = arima.sim(model=list(ar=0.6), n=200)
> class(x.ar1)
[1] "ts"
> head(x.ar1)
[1]  1.12765823  0.41827288 -0.16936098  0.03314966 -0.97260792 -0.50740934
```

Abbildung 3.7 gibt ein Zeitreihendiagramm des AR(1)-Prozesses wieder, das mit dem plot-Befehl erstellt worden ist:

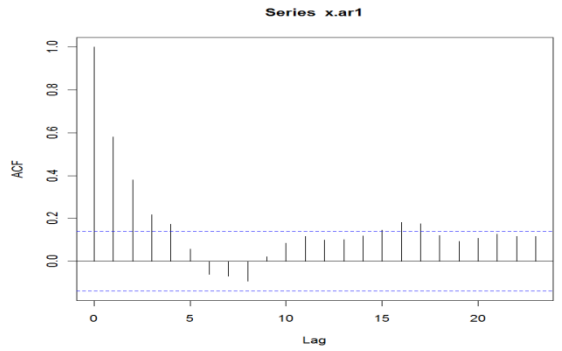
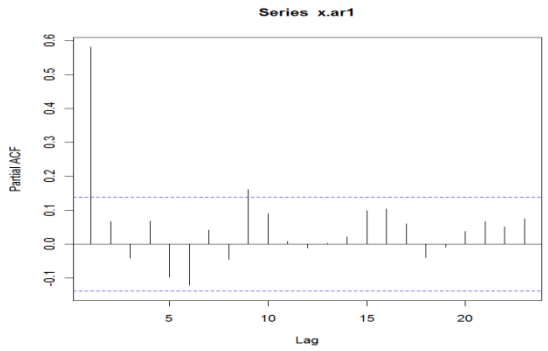
```
> plot(x.ar1, xlab="Zeitindex", main="AR(1)-Prozess")
> box(which="outer")
```

Abbildung 3.7: AR(1)-Prozess (ϕ_1=0,6)



In der Tabelle 3.2 werden die Autokorrelationskoeffizienten und die partiellen Autokorrelationskoeffizienten des AR(1)-Prozesses x.ar1 grafisch und numerisch wiedergegeben:

Tabelle 3.2: ACF und PACF eines simulierten AR(1)-Prozesses

AR(1)-Prozess mit ϕ_1=0,6	
Autokorrelationsfunktion	Partielle Autokorrelationsfunktion
>xar1.acf = acf(x.ar1)	> xar1.pacf = acf(x.ar1, type="partial") oder > xar1.pacf = pacf(x.ar1)
	
> xar1.acf Autocorrelations of series 'x.ar1', by lag 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 1.000 0.581 0.381 0.217 0.172 0.057 -0.061 - 0.069 -0.093 0.021 0.084 0.116 0.099 0.101 0.118 0.146 0.181 0.176 0.121 0.093 0.107 0.127 0.115 0.117	> xar1.pacf Partial autocorrelations of series 'x.ar1', by lag 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0.581 0.066 -0.042 0.068 -0.098 -0.122 0.041 - 0.046 0.161 0.089 0.008 -0.012 0.002 0.020 0.098 0.104 0.060 -0.041 -0.010 0.036 0.066 0.051 0.074

Anhand der partiellen Autokorrelationsfunktion lässt sich der AR(1)-Prozess identifizieren. Der geschätzte partielle Autokorrelationskoeffizient 1. Ordnung von 0,581 ist signifikant, während die nachfolgenden partiellen Autokorrelationskoeffizienten – mit Ausnahme eines einzelnen beim Lag 9 - innerhalb des 2σ-Intervalls liegen. Dass ein einzelner partieller Autokorrelationskoeffizient höherer Ordnung signifikant ist, lässt sich bei einem AR(1)-Prozess zufallsbedingt durchaus erwarten.

Die theoretischen ACF und PACF können in R mit der ARMAacf-Funktion ermittelt werden:

```
> AR06.acf = ARMAacf(ar=0.6, lag.max=23)
> AR06.pacf = ARMAacf(ar=0.6, lag.max=23, pacf=TRUE)
```

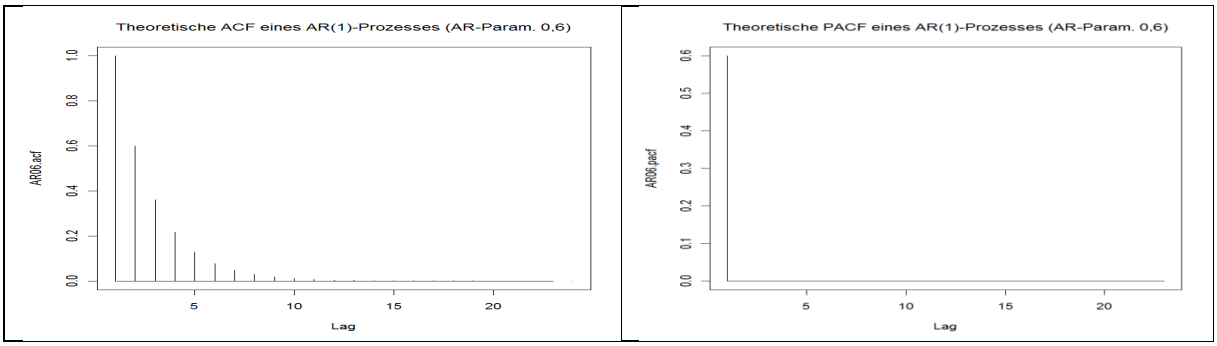
Auf eine Ausgabe der konkreten Werte wird hier verzichtet. Stattdessen sind in Abbildung 3.8 die Plots der theoretischen Autokorrelations- und partiellen Autokorrelationskoeffizienten bis zum Lag 23 für den AR(1)-Prozess mit ϕ_1=0,6 wieder-gegeben:

```
> y0= rep(0, 23)
> plot(AR06.acf, type='h', xlab="Lag", main="Theoretische ACF eines AR(1)-Prozesses (AR-Param. 0,6)",
font.main=1)
> lines(y0)
> plot(AR06.pacf, type='h', xlab="Lag", main="Theoretische PACF eines AR(1)-Prozesses (AR-Param. 0,6)",
font.main=1)
```

```
> lines(y0)
```

Die Spezifikation des Arguments type='h" fordert eine Darstellung der Autokorrelationen und partiellen Autokorrelationen in Form eines Stabdiagramms an.

Abbildung 3.8: Theoretische ACF und PACF eines AR(1)-Prozesses



Analog kann der AR(2)-Prozess

(3.5) $x_t = \phi_1 \cdot x_{t-1} + \phi_2 \cdot x_{t-2} + u_t$

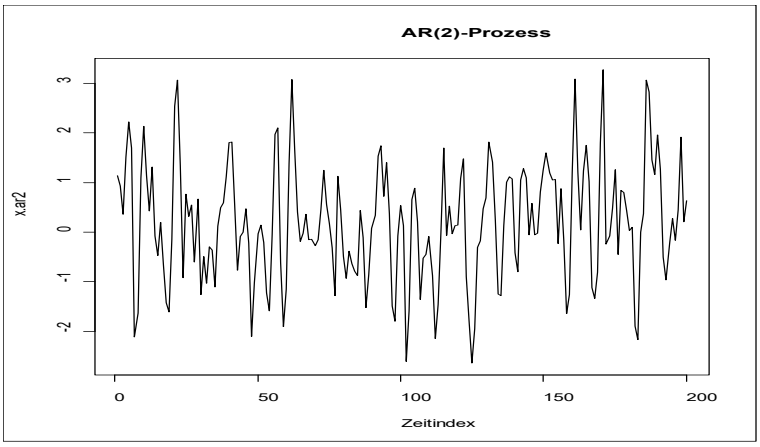
simuliert werden. Nach Initialisierung des Zufallszahlengenerators mit dem set.seed-Befehl wird ein AR(2)-Prozess x.ar2 mit den autoregressiven Parametern ϕ_1= 0,8 und ϕ_2= -0,5 erzeugt:

```
> set.seed(seed=147358)
> x.ar2 = arima.sim(model=list(ar=c(0.8, -0.5)), n=200)
> class(x.ar2)
[1] "ts"
> head(x.ar2)
[1] 1.1410103 0.9352431 0.3641408 1.5102744 2.2197865 1.6968726
```

Das in Abbildung 3.8 wiedergegebene Zeitreihendiagramm des AR(2)-Prozesses ist mit dem plot-Befehl erstellt worden ist:

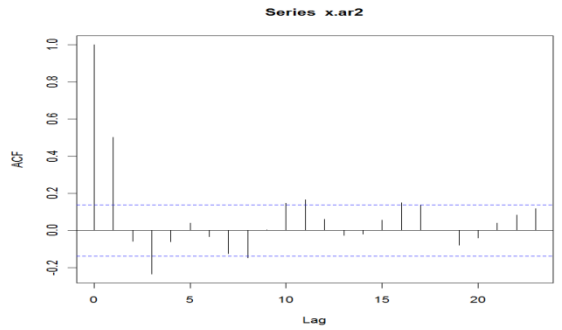
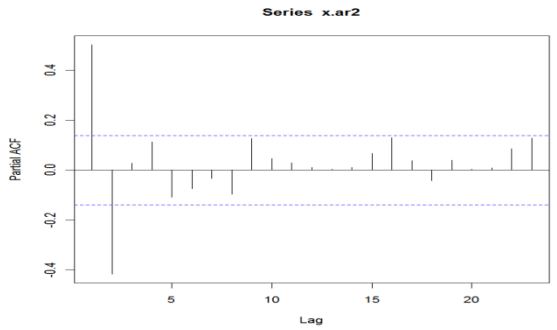
```
>plot(x.ar2, xlab="Zeitindex", main="AR(2)-Prozess")
> box(which="outer")
```

Abbildung 3.9: AR(2)-Prozess (ϕ_1=0,8, ϕ_2=-0,5)



Die Verläufe der ACF- und PACF sowie die Werte der Autokorrelationskoeffizienten und partiellen Autokorrelationskoeffzienten des AR(2)-Prozesses x.ar2 gehen aus der nachfolgenden Tabelle 3.3 hervor:

Tabelle 3.3: ACF und PACF eines AR(2)-Prozesses

AR(2)-Prozess mit ϕ_1=0,8 und ϕ_2=-0,5	
Autokorrelationsfunktion	Partielle Autokorrelationsfunktion
>xar2.acf = acf(x.ar2)	> xar2.pacf = acf(x.ar2, type="partial") oder > xar2.pacf = pacf(x.ar2)
	
> xar2.acf Autocorrelations of series 'x.ar2', by lag 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 1.000 0.502 -0.059 -0.234 -0.060 0.041 -0.034 - 0.126 -0.147 0.003 0.147 0.166 0.059 -0.028 - 0.020 0.055 0.149 0.136 0.002 -0.081 -0.040 0.040 0.084 0.118	> xar2.pacf Partial autocorrelations of series 'x.ar2', by lag 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0.502 -0.416 0.027 0.113 -0.108 -0.074 -0.034 - 0.096 0.126 0.047 0.030 0.010 0.003 0.011 0.068 0.130 0.038 -0.042 0.040 0.004 0.009 0.086 0.129

Man erkennt klar den für AR(2)-Prozesse zyklischen Verlauf der ACF-Funktion. Obwohl auch einige signifikante Autokorrelationskoeffizienten bei höheren Lags auftreten, weist sie insgesamt gedämpfte Schwingungen auf. Die Abnahme der Amplitude ist für einen stationären Prozess charakteristisch. Dagegen bricht die PACF-Funktion nach einem Lag von 2 ab, wodurch ein AR(2)-Prozess identifiziert wird.

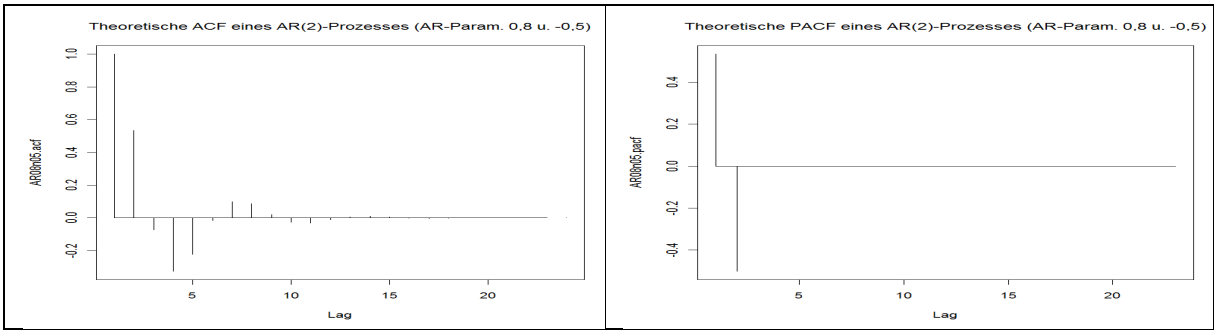
Wie schon für den AR(1)-Prozess werden für den AR(2)-Prozess die theoretischen Autokorrelationskoeffizienten und partiellen Autokorrelationskoeffizienten berechnet,

```
> AR08n05.acf = ARMAacf(ar=c(0.8, -0.5), lag.max=23)
> AR08n05.pacf = ARMAacf(ar=c(0.8, -0.5), lag.max=23, pacf=TRUE),
```

und grafisch dargestellt (Abbildung 3.10):

```
> plot(AR08n05.acf, type='h', xlab="Lag", main="Theoretische ACF eines AR(2)-Prozesses (AR-Param. 0,8
u. -0,5)", font.main=1)
> lines(y0)
> plot(AR08n05.pacf, type='h', xlab="Lag", main="Theoretische PACF eines AR(2)-Prozesses (AR-Param.
0,8 u. -0,5)", font.main=1)
> lines(y0)
```

Abbildung 3.10: Theoretische ACF und PACF eines AR(2)-Prozesses



Bei Moving-Average-Prozessen (MA-Prozesse) werden die Werte einer Zeitreihe (x_t) durch die kontemporäre Störgröße u_t und zeitlich verzögerten Störgrößen u_{t-i} modelliert, die einem reinen Zufallsprozess entstammen. Ein MA(q)-Prozess ist durch eine Einbeziehung von Lags der Störgröße u bis zur Ordnung q gegeben:

(3.6) $x_t = u_t - \theta_1 \cdot u_{t-1} - \theta_2 \cdot u_{t-2} - \ldots - \theta_q \cdot u_{t-q}$

Die Koeffizienten $\theta_1, \theta_2, \ldots, \theta_q$ sind Parameter des MA(q)-Prozesses. Die Prozessgleichung des MA(1)-Prozesses lautet

(3.7) $x_t = u_t - \theta_1 \cdot u_{t-1}$.

Zur Simulation des MA(1)-Prozesses setzen wir den Koeffizienten θ_1 nach Initialisierung des Zufallszahlengenerators gleich 0,5:

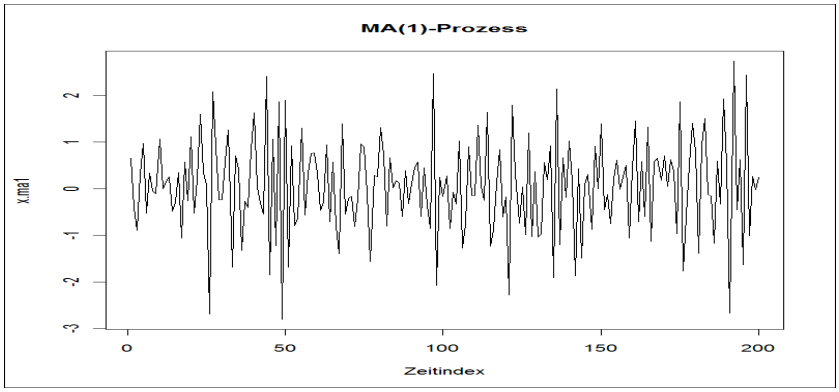
```
> set.seed(seed=147358)
> x.ma1 = arima.sim(model=list(ma=-0.5), n=200)
> class(x.ma1)
[1] "ts"
> head(x.ma1)
[1] 0.6522325 -0.4111826 -0.8951229 0.3186974 0.9736495 -0.5309553
```

Das Zeitreihendiagramm des MA(1)-Prozesses, das mit dem plot-Befehl erstellt worden ist,

```
> plot(x.ma1, xlab="Zeitindex", main="MA(1)-Prozess")
> box(which="outer"),
```

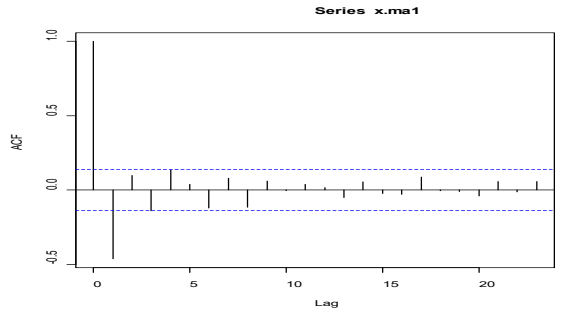
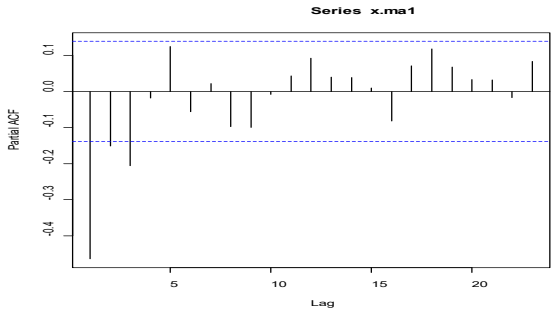
zeigt Abbildung 3.11.

Abbildung 3.11: MA(1)-Prozess (θ_1=0,5)



Die ACF und PACF des erzeugten MA(1)-Prozesses gehen aus der nachfolgenden Tabelle 3.4 hervor:

Tabelle 3.4: ACF und PACF eines MA(1)-Prozesses

MA(1)-Prozess mit θ_1=0,5	
Autokorrelationsfunktion	Partielle Autokorrelationsfunktion
> xma1.acf = acf(x.ma1)	> xma1.pacf = acf(x.ma1, type="partial") oder xma1.pacf = pacf(x.ma1)
	
> xma1.acf Autocorrelations of series 'x.ma1', by lag 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 1.000 -0.465 0.098 -0.140 0.136 0.039 -0.122 0.079 -0.115 0.061 -0.004 0.038 0.017 -0.050 0.054 -0.024 -0.028 0.089 -0.005 -0.010 -0.041 0.058 -0.013 0.059	> xma1.pacf Partial autocorrelations of series 'x.ma1', by lag 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 -0.465 -0.151 -0.206 -0.019 0.125 -0.057 0.022 - 0.098 -0.099 -0.009 0.043 0.093 0.040 0.038 0.010 -0.082 0.071 0.118 0.067 0.033 0.032 - 0.017 0.083

Die ACF bricht nach einem Lag von 1 ab, da – abgesehen von der Eigenkorrelation r_0 – nur der Autokorrelationskoeffizient 1. Ordnung von -0,465 außerhalb des 2σ-Bandes liegt. Dagegen verläuft die PACF in gedämpften Schwingungen, wobei die partiellen Autokorrelationskoeffizienten bis zur dritten Ordnung signifikant sind. Aufgrund der Verläufe der ACF und PACF lässt sich für die Zeitreihe x-ma1 ein MA(1)-Prozess identifizieren.

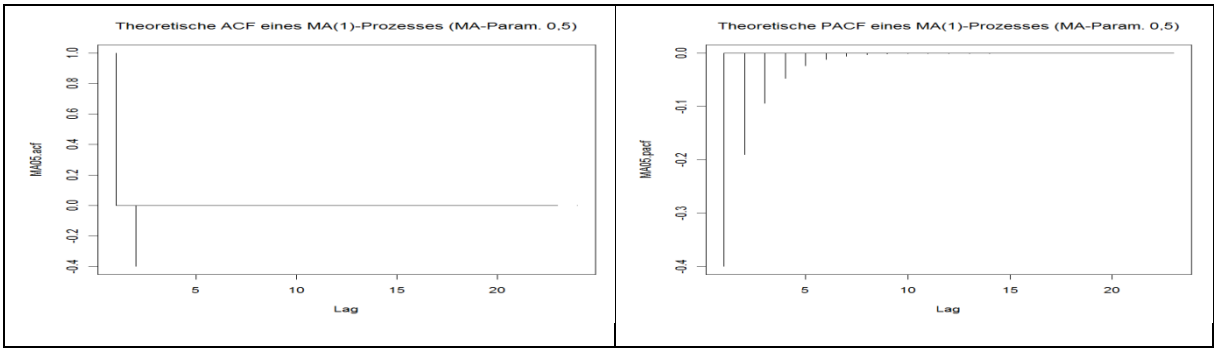
Auch für den MA(1)-Prozess sollen wiederum die theoretischen Autokorrelationskoeffizienten und partiellen Autokorrelationskoeffizienten berechnet,

```
> MA05.acf = ARMAacf(ma=-0.5, lag.max=23)
> MA05.pacf = ARMAacf(ma=-0.5, lag.max=23, pacf=TRUE),
```

und grafisch dargestellt werden (Abbildung 3.12):

```
> plot(MA05.acf, type='h', xlab="Lag", main="Theoretische ACF eines MA(1)-Prozesses (MA-Param. 0,5)",
font.main=1)
> lines(y0)
> plot(MA05.pacf, type='h', xlab="Lag", main="Theoretische PACF eines MA(1)-Prozesses (MA-Param.
0,5)", font.main=1)
> lines(y0).
```

Abbildung 3.12: Theoretische ACF und PACF eines MA(1)-Prozesses



Übungsaufgaben

3.2.1 Setzen Sie den seed gleich147358 und erzeugen Sie einen MA(2)-Prozess der Länge 200 mit den autoregressiven Parametern θ_1=-0,7 und θ_2=-0,3 und erstellen Sie ein geeignet gestaltetes Zeitreihendiagramm!

3.2.2 Vergleichen Sie das Zeitreihendiagramm des MA(2)-Prozesses mit dem Zeitreihendiagramm eines Gauß-Prozesses (standardnormalverteilte Zufallszahlen) der Länge 200!

3.2.3 Identifizieren Sie das der in Aufg. 3.2.1 erzeugten Zeitreihe zugrundeliegende ARMA-Modell anhand der ACF- und PACF-Funktion!

3.2.4 Stellen Sie die theoretischen Autokorrelationskoeffizienten und partiellen Autokorrelationskoeffizienten für den der Aufg. 3.2.1 zugrundeliegenden MA(2)-Prozess grafisch dar und interpretieren Sie die beiden Diagramme!

3.3 ARIMA-Modellierung ökonomischer Zeitreihen

Für eine ARIMA-Modellierung ökonomischer Zeitreihen werden saisonbereinigte Quartalsdaten der Volkswirtschaftlichen Gesamtrechnung in Österreich für den Zeitraum 1998/I bis 2011/IV herangezogen, die in der CSV-Datei VGRA verfügbar sind. Sie werden mit dem read.csv-Befehl in R eingelesen und unter dem Objektnamen VGRA als R-Objekt gespeichert:

```
> VGRA = read.csv(file="VGRA.csv", dec=".", sep=";", header=TRUE)
> class(VGRA)
[1] "data.frame"
```

Der Datensatz enthält Produktionsdaten (Bruttowertschöpfung (BWS) in Mrd. €) für die gesamte Volkswirtschaft sowie für ausgewählte Wirtschaftsbereiche:

```
> str(VGRA)
'data.frame':  56 obs. of  8 variables:
 $ Zeit   : num  1998 1998 1998 1998 1999 ...
 $ BAU    : num  25.7 25.7 26.1 26.6 27.6 ...
 $ BWS    : num  402 400 399 402 412 ...
 $ DIENST : num  81.6 82.4 83.8 85.1 85.9 ...
```

```
 $ FINANZ  : num  99.9 100.6 101.4 101.7 103.3 ...
 $ HANDEL  : num  72 71.5 69.2 70.5 74 ...
 $ LANDW   : num  4.99 5 5.02 5.06 5.26 5.32 5.33 5.35 5.36 5.43 ...
 $ PROD_GEW: num  139 141 140 140 142 ...
```

Unter Verwendung des head-Befehls erhalten wir eine Übersicht über die in dem data.frame-Objekt VGRA enthaltenen Produktionswerte in den Anfangsperioden:

```
> head(VGRA)
   Zeit   BAU    BWS DIENST FINANZ HANDEL LANDW PROD_GEW
1 1998.1 25.71 401.74  81.57  99.92  72.02  4.99  139.440
2 1998.2 25.68 400.20  82.42 100.62  71.51  5.00  141.157
3 1998.3 26.05 398.61  83.76 101.39  69.23  5.02  139.962
4 1998.4 26.55 402.15  85.09 101.68  70.46  5.06  140.039
5 1999.1 27.58 411.93  85.91 103.32  73.96  5.26  142.437
6 1999.2 27.57 408.84  86.04 104.08  73.29  5.32  139.777
```

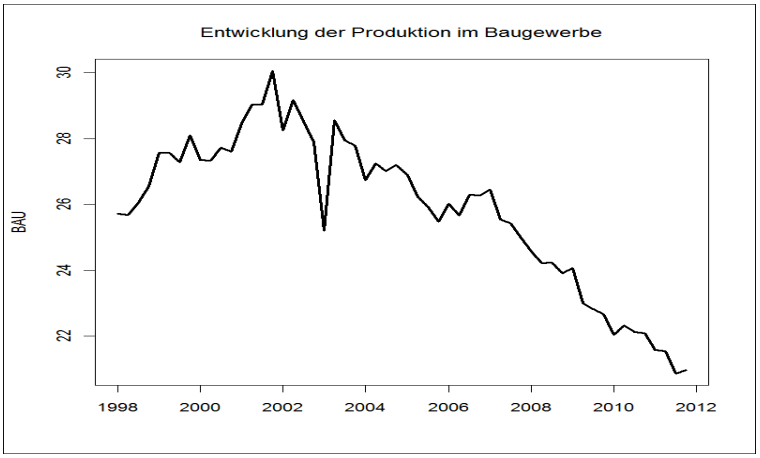
Für die ARIMA-Analyse wird die Variable BAU betrachtet, die daher als Zeitreihe definiert wird:

```
>BAU = ts(VGRA$BAU, start=c(1998,1), end=c(2011,4), frequency=4)
>class(BAU)
[1] "ts"
>str(BAU)
 Time-Series [1:56] from 1998 to 2012: 25.7 25.7 26.1 26.6 27.6 ...
> BAU
       Qtr1  Qtr2  Qtr3  Qtr4
1998 25.71 25.68 26.05 26.55
1999 27.58 27.57 27.28 28.10
2000 27.35 27.33 27.73 27.60
2001 28.46 29.04 29.03 30.05
2002 28.26 29.18 28.53 27.89
2003 25.21 28.56 27.95 27.79
2004 26.74 27.25 27.01 27.20
2005 26.89 26.23 25.92 25.48
2006 26.03 25.66 26.29 26.28
2007 26.45 25.54 25.43 24.98
2008 24.57 24.22 24.24 23.91
2009 24.06 23.01 22.82 22.66
2010 22.03 22.32 22.12 22.08
2011 21.58 21.55 20.87 20.97
```

Das Zeitreihendiagramm 3.13 zeigt die Entwicklung der BAU-Reihe in dem Zeitraum vom 1. Quartal 1998 bis zum 4. Quartal 2011 auf:

```
> plot(BAU, main="Entwicklung der Produktion im Baugewerbe", xlab="", lwd=3, cex.main = 1.2,
font.main=1)
> box(which="figure")
```

Abbildung 3.13: Zeitreihendiagramm der Produktion im Baugewerbe

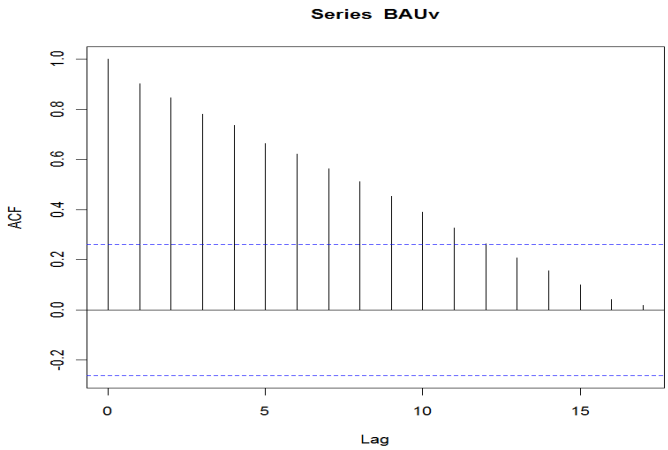


Um das Korrelogramm mit der Zeitreihe BAU mit der üblichen Lagspezifikation zu erhalten, wird sie in der acf-Funktion als Vektor verwendet:

```
> BAUv = as.vector(BAU)
> BAUv.acf = acf(BAUv)
```

Wie aus Abbildung 3.14 ersichtlich ist, überdeckt der Trend alle anderen eventuell in der Zeitreihe BAU enthaltenen Strukturen.

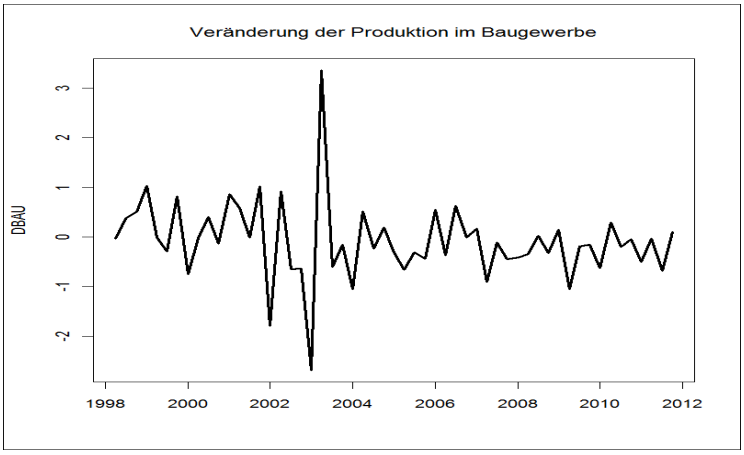
Abbildung 3.14: Korrelogramm der Zeitreihe BAU



Aufgrund des vorhandenen abnehmenden Trends ist die saisonbereinigte Zeitreihe der Bauproduktion nichtstationär. Aus diesem Grund werden im Weiteren die ersten Differenzen der Zeitreihe BAU analysiert (Abbildung 3.15):

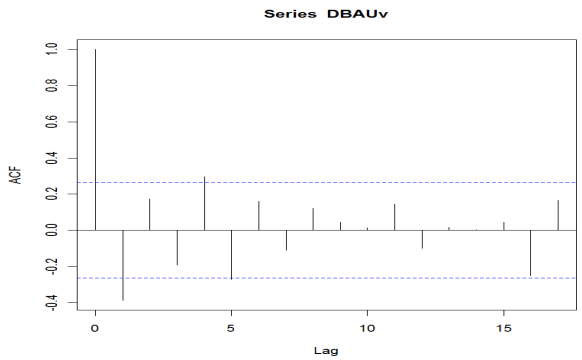
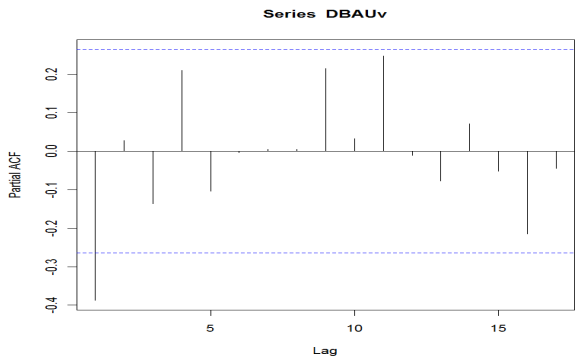
```
> DBAU = diff(BAU)
> plot(DBAU, main="Veränderung der Produktion im Baugewerbe", xlab="", lwd=3, cex.main = 1.2,
font.main=1)
> box(which="figure").
```

Abbildung 3.15: Zeitreihendiagramm der ersten Differenzen der Bauproduktion



Um die gewöhnliche Lagspezifikation in der ACF und PACF in der nachstehenden Tabelle auszuweisen, verwenden wir die differenzierte Baureihe als Vektor:

```
>DBAUv = as.vector(DBAU)
```

Zeitreihe DBAU	
Autokorrelationsfunktion	Partielle Autokorrelationsfunktion
> DBAUv.acf = acf(DBAUv)	> DBAUv.pacf = acf(DBAUv, type="partial") oder DBAUv.pacf = pacf(DBAUv)
	
>DBAUv.acf Autocorrelations of series 'DBAUv', by lag 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 1.000 -0.387 0.174 -0.192 0.295 -0.272 0.160 - 0.110 0.122 0.045 0.014 0.145 -0.100 0.016 0.002 0.044 -0.252 0.165	> DBAUv.pacf Partial autocorrelations of series 'DBAUv', by lag 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 -0.387 0.029 -0.136 0.210 -0.104 -0.003 0.005 0.006 0.216 0.033 0.248 -0.010 -0.078 0.071 - 0.052 -0.214 -0.044

Man erkennt, dass die ACF der ersten Differenzen der BAU-Reihe alternierend mit abnehmender Amplitude verläuft. Der exponentiell abnehmende Verlauf ist durch einige signifikante Autokorrelationskoeffizienten niedriger Ordnung und nicht-signifikante Autokorrelationskoeffizienten höherer Ordnung bei wechselnden Vorzeichen charakterisiert. Dagegen bricht die PACF nach einem Lag von 1 ab. Aus den Verläufen der ACF und PACF lässt sich für die DBAU-Reihe ein AR(1)-Prozess identifizieren.

Da die Zeitreihe BAU nach Bildung einfacher Differenzen in eine stationäre Zeitreihe überführt werden konnte, wird für sie ein ARIMA(1,1,0)-Modell angepasst. Hierzu steht in R die arima-Funktion zur Verfügung:

```
> BAU.ar1 = arima(BAU, order = c(1,1,0))
> BAU.ar1
Call:
arima(x = BAU, order = c(1, 1, 0))
Coefficients:
        ar1
   -0.3642
s.e.  0.1239
sigma^2 estimated as 0.5609:  log likelihood = -62.21,  aic = 128.43
> class(BAU.ar1)
[1] "Arima"
```

Der geschätzte autoregressive Parameter ar1 (= $\hat{\phi}_1$) von -0,3642 ist erwartungsgemäß negativ. Die arima-Funktion stellt hierzu keine z-Werte zur Verfügung, jedoch sind sie aus ihren Komponenten BAU.ar1$coef und BAU.ar1$var.coef,

```
> BAU.ar1$coef
      ar1
-0.3641763
> class(BAU.ar1$coef)
[1] "numeric"
> BAU.ar1$var.coef
ar1
ar1 0.01533949
> class(BAU.ar1$var.coef)
[1] "matrix",
```

berechenbar. Da BAU.ar1$var.coef die Varianz des geschätzten Regressionskoeffizienten enthält, ist die Quadratwurzel der Standardfehler:

```
> BAU.ar1$se = diag(sqrt(BAU.ar1$var.coef))
> BAU.ar1$se
     ar1
0.1238527.
```

Der z-Wert von

```
> BAU.ar1$z = BAU.ar1$coef/BAU.ar1$se
> BAU.ar1$z
     ar1
-2.940399
```

ist absolut größer als das 0,975-Quantil der Standardnormalverteilung von $z_{0,975}$=1,96, so dass der geschätzte autoregressive Parameter ar1 (= $\hat{\phi}_1$) auf dem 5%-Niveau

signifikant ist (zweiseitiger Test). Die Residualvarianz sigma^2 nimmt einen Wert von 0,5609 an, so dass der Standardfehler der Regression (SER) 0,7489 beträgt.

Mit dem log likelihood-Wert von -62,21 lässt sich Akaikes Informationskriterium aic bestimmen:

```
> n = length(BAU)
>n
[1] 56
> npar = length(BAU.ar1$coef)+1   # Anzahl der Parameter
> npar
[1] 2
> AIC = -2*BAU.ar1$loglik + 2*npar
> AIC
[1] 128.4276.
```

Dieser Wert ist nach Ausführung der arima-Funktion gerundet unter dem Komponentennamen aic verfügbar. Er lässt sich auch über die AIC-Funktion ermitteln, in der das Argument k standardmäßig auf den Wert 2 gesetzt ist:

```
> AIC(BAU.ar1)   oder   > AIC(BAU.ar1, k=2)
[1] 128.4276
```

Das Bayessche Informationskriterium (BIC) wird dagegen nicht ausgewiesen, jedoch lässt es sich unter Verwendung des log likelihood-Werts einfach berechnen,

```
> BIC = -2*BAU.ar1$loglik + log(n)*npar
> BIC
[1] 132.4783,
```

oder nach Setzung von k=log(n) mit der AIC-Funktion bestimmen:

```
> AIC(BAU.ar1, k=log(n))
[1] 132.4783.
```

Die Zeitreihe der Residuen BAU.res,

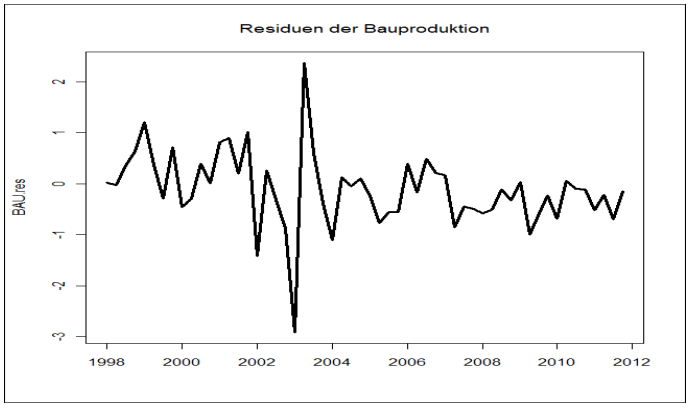
```
> BAU.res = BAU.ar1$residuals,
```

ist unter Verwednung der plot-Funktion,

```
> plot(BAU.res, main="Residuen  der Bauproduktion", xlab="", lwd=3, cex.main = 1.2, font.main=1)
> box(which="figure").
```

in der Abbildung 3.16 dargestellt:

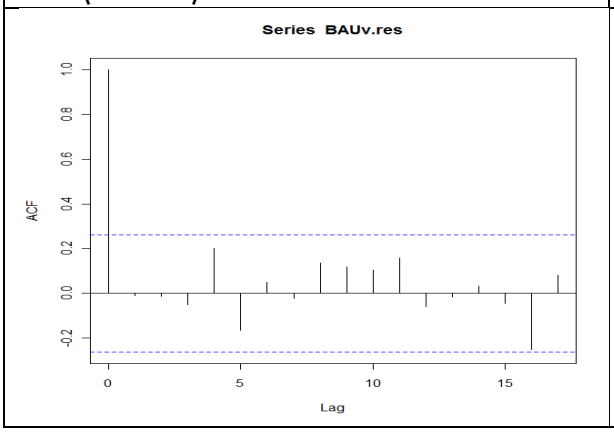
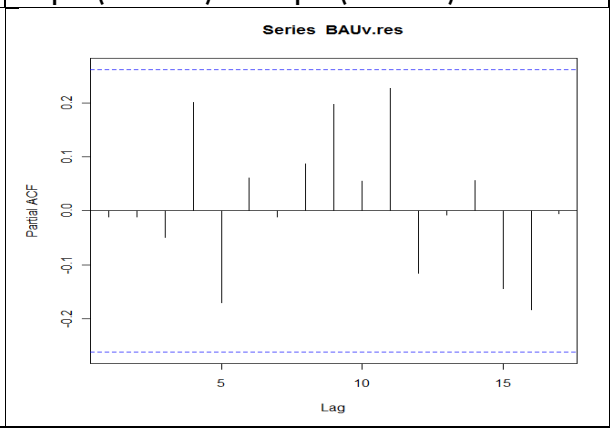
Abbildung 3.16: Zeitreihendiagramm der Residuen der Bauproduktion



Die ACF und PACF der Reihe BAUv.res,

```
> BAUv.res = as.vector(BAU.res),
```

lassen keine systematischen Komponenten der Residualreihe mehr erkennen:

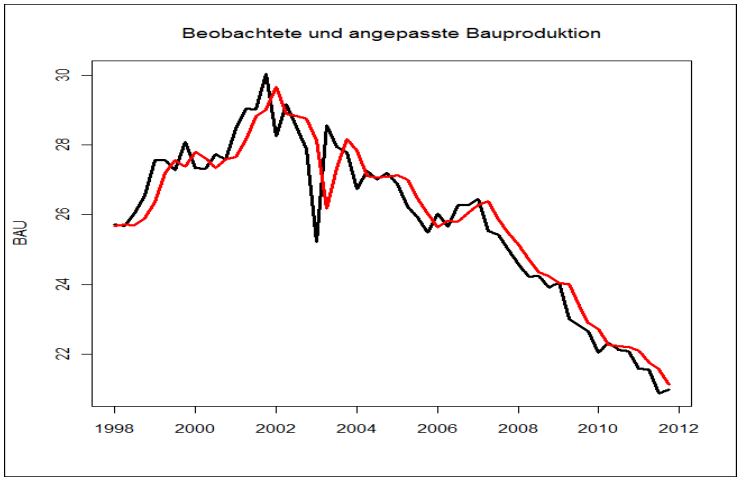
> acf(BAUv.res)	> pacf(BAUv.res) oder pacf(BAUv.res)
	

Aus Abbildung 3.17 lässt sich grafisch die Güte der Anpassung der beobachteten und angepassten Bauproduktion entnehmen:

```
> BAU.fit = BAU - BAU.res
> plot(BAU, main="Beobachtete und angepasste Bauproduktion", xlab="", lwd=3, cex.main = 1.2,
font.main=1)
>lines(BAU.fit, col="red", lwd=3)
>box(which="figure").
```

Zur besseren Vergleichbarkeit wird die beobachtete Bauproduktion durch einen schwarzen Kurvenzug wiedergegeben, während die angepasste Bauproduktion durch roten Kurvenzug dargestellt ist.

Abbildung 3.17: Beobachtete und angepasste Bauproduktion



Übungsaufgaben

3.3.1 Definieren Sie die Variable DIENST des data.frame-Objekts VGRA, die die Wertschöpfung des Dienstleistungssektors wiedergibt, als Zeitreihe! Legen Sie anhand des Zeitreihenplots und Korrelogramms der Zeitreihe und ihrer Differenzen die Ordnung d des integrierten Teils des ARIMA(p,d,q)-Modells fest!

3.3.2 Identifizieren sie anhand der Autokorrelationsfunktion und partiellen Autokorrelationsfunktion der d-fach differenzierten Zeitreihe der Dienstleistungsproduktion (mit in Aufg. 3.3.1 festgelegter Ordnung d) ein adäquates ARMA-Modell!

3.3.3 Berechnen Sie für das in Aufg. 3.3.2 identifizierte ARMA-Modell die Informationskriterien AIC und BIC! Vergleichen Sie die AIC- und BIC-Werte mit denen eines ARMA(1,1)-Modells!

3.4 ARIMA-Prognose ökonomischer Zeitreihen

Um die Qualität einer Prognose auf der Grundlage des identifizierten ARIMA(1,1,0)-Modells für die Zeitreihe BAU zu bewerten, wird eine Anpassung im verkürzten Stützzeitraum vom 1. Quartal 1998 bis zum 4. Quartal 2010 durchgeführt:

```
> BAU.ar1k = arima(BAU[-c(53:56)], order = c(1,1,0))


> BAU.ar1k
Call:
arima(x = BAU[-c(53:56)], order = c(1, 1, 0))
Coefficients:
       ar1
  -0.3707
s.e.  0.1281
sigma^2 estimated as 0.589:  log likelihood = -58.94,  aic = 121.88
```

Unter Nichtberücksichtigung der Quartalswerte des Jahres 2011 verringert sich der Schätzer des autoregressiven Parameters ϕ_1 geringfügig von -0.3642 auf -0.3707.

Bei der Prognose der Quartalswerte des Jahres 2011 gehen wir von dem Informationsstand am Ende des Jahres 2010 aus. Die Ex-post-Prognose ist in diesem Fall eine Kombination aus einer 1-, 2-, 3- und 4-Schritt-Prognose und erfolgt unter Verwendung der predict-Funktion:

```
> BAU.pred = predict(BAU.ar1k, n.ahead=4, se.fit=FALSE)
> BAU.pred
Time Series:
Start = 53
End = 56
Frequency = 1
[1] 22.09483 22.08933 22.09137 22.09061
> BAU.predv = as.vector(BAU.pred)
> BAU.predv
[1] 22.09483 22.08933 22.09137 22.09061
```

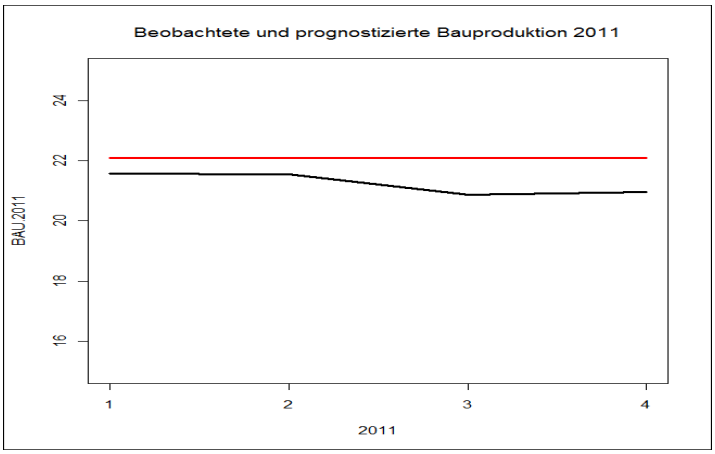
Wenn die Prognosewerte der Bauproduktion für die vier Quartale des Jahres 2011 auch nicht identisch sind, unterscheiden sie sich doch nur geringfügig voneinander. Nach Definition der Reihe BAU.2011 mit den Quartalswerten der Bauproduktion des Jahres 2011,

```
> BAU.2011 = BAU[-c(1:52)]
> BAU.2011
[1] 21.58 21.55 20.87 20.97,
```

lassen sich die beobachteten und prognostizierten Quartalswerte der Bauproduktion 2011 in einem Zeitreihendiagramm (Abbildung 3.18) gegenüberstellen:

```
> plot(BAU.2011, main="Beobachtete und prognostizierte Bauproduktion 2011", type="l", xlab="2011",
ylim=c(15, 25), lwd=3, cex.main = 1.2, font.main=1, xaxt="n")
>lines(BAU.predv, col="red", lwd=3)
>axis(side=1, at=c(1,2,3,4))
>box(which="figure")
```

Abbildung 3.18: Beobachtete und prognostizierte Bauproduktion 2011



Zusätzlich zu dem visuellen Vergleich soll die Genauigkeit der Vorhersage der Bauproduktion für die Quartale des Jahres 2011 mit dem ARIMA(1, 1, 0)-Modell anhand von Prognosefehlermaßen quantifiziert werden. Tabelle 3.5 enthält die gängigen Fehlermaße, die für die hier verwendete Kombination aus Ein- und Mehr-Schritt-Prognosen definiert sind.

Tabelle 3.5: Prognosefehlermaße bei einer Kombination von Ein- und Mehr-Schritt-Prognosen

Mittlerer absoluter Fehler (MAE)	Mittlerer absoluter prozentualer Fehler (MAPE)
(3.8) $\mathrm{MAE} = \frac{1}{h}\sum_{j=1}^{h}\left\lvert x_{t+j} - \hat{x}_{t,j}\right\rvert$	(3.9) $\mathrm{MAPE} = \frac{1}{h}\sum_{j=1}^{h}\left\lvert \frac{x_{t+j} - \hat{x}_{t,j}}{x_{t+j}}\right\rvert \cdot 100\%$
Mean Square Error (MSE)	Root Mean Square Error (RMSE)
(3.10) $\mathrm{MSE} = \frac{1}{h}\sum_{j=1}^{h}\left(x_{t+j} - \hat{x}_{t,j}\right)^2$	(3.11) $\mathrm{RMSE} = \sqrt{\frac{1}{h}\sum_{j=1}^{h}\left(x_{t+j} - \hat{x}_{t,j}\right)^2}$

Um die Prognosefehlermaße zu berechnen, definieren wir sie als eigene R-Funktionen, die in Skriptdateien gespeichert und mittels des Befehls source("filename.R") geladen werden. Eine eigene R-Funktion lässt sich mit dem Befehl function() erstellen,

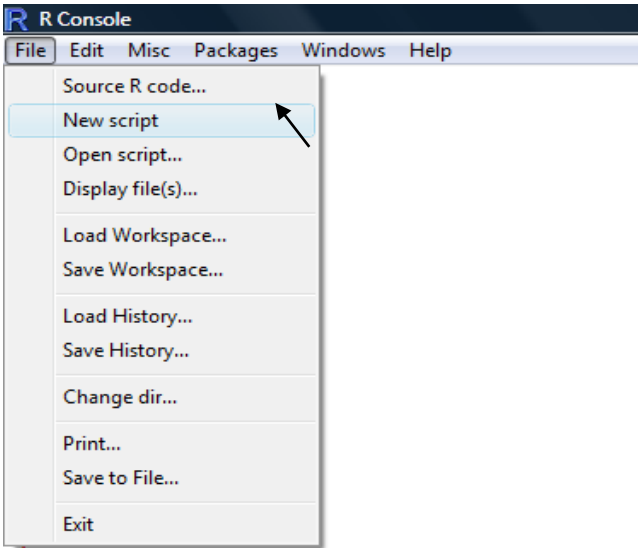
```
myfunc = function(x, y, ...) {
        Anweisung1;
        Anweisung2;
        ...
    }
```

die unter Angabe eines oder mehrerer Parameter x,. y,. … aufgerufen werden kann. Soll das Ergebnis in dem Objekt erg gespeichert werden, lautet der Aufruf

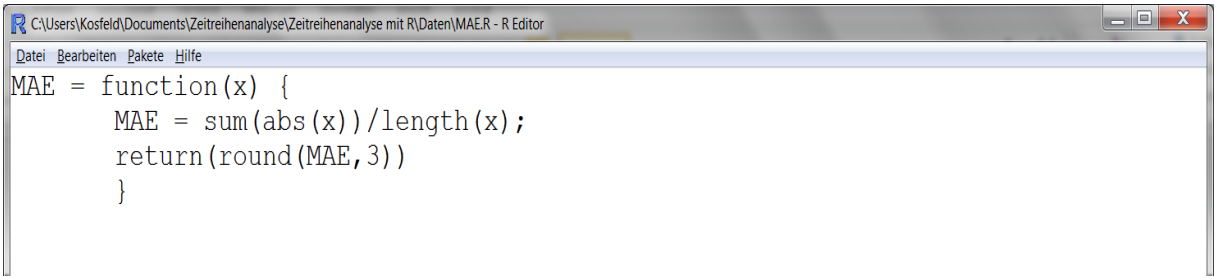
```
> erg = myfunc(x, y,  ).
```

Eine oder mehrere Anweisungen, die durch Semikola voneinander getrennt werden, stehen innerhalb geschweifter Klammern. Unter Verwendung der Anweisung return(erg) wird das Ergebnis erg an die R-Konsole zurückgemeldet.

Um eine eigene Funktion zur Berechnung des mittleren absoluten Fehlers (MAE) zu schreiben, wählen wir in dem R-Menü File – New script:



In dem sich öffnenden R-Editor wird die Anweisung zur Berechnung des mittleren absoluten Fehlers (MAE) eingegeben und getrennt durch ein Semikolon der return-Befehl:



Anschließend wird die R-Funktion MAE nach Wahl der Menüpunkte File – Save as ... unter dem Dateinamen MAE.R gespeichert. Nach Definition der Differenz zwischen der tatsächlichen und prognostizierten Bauproduktion des Jahres 2011 (BAU.err),

```
> BAU.err = BAU.2011 - BAU.pred
> BAU.err
[1] -0.5148297 -0.5393317 -1.2213701 -1.1206144,.
```

und Laden des R-Scripts MAE.R,

```
> source("MAE.R")
```

lässt sich der mittlere absolute Prognosefehler der Ex-post-Prognose der Bauproduktion durch Aufruf der MAE-Funktion berechnen und in der R-Konsole ausgeben:

```
> MAE(BAU.err)
[1] 0.849.
```

Eine Speicherung des berechneten Wertes in einem R-Objekt z.B. unter dem Namen MAE erfolgt durch eine entsprechende Zuweisung bei dem Funktionsaufruf:

```
> MAE = MAE(BAU.err)
```

```
> MAE
[1] 0.849.
```

Analog lassen sich eigene R-Funktionen für die Fehlermaße MAPE, MSE und RMSE definieren und berechnen. Die Funktion des mittleren absoluten prozentualen Fehlers (MAPE) ist durch

```
MAPE = function(x,y) {
MAPE = (sum(abs(x/y))/length(x))*100;
return(round(MAPE,3))
   }
```

definiert. Der Parameter x steht hierbei für den Prognosefehler und der Parameter y für die tatsächlichen Zeitreihenwerte in dem Ex-post-Prognosezeitraum. Nach Speicherung der Funktion MAPE unter dem Dateinamen MAPE.R und anschließendes Laden dieses R-Scripts,

```
> source("MAPE.R"),
```

lässt sich der mittlere absolute prozentuale Prognosefehler (MAPE) für die Bauprognose berechnen:

```
> MAPE = MAPE(BAU.err, BAU.2011)
> MAPE
[1] 4.021
```

Der MAPE-Wert weist einen mittleren prozentualen Prognosefehler von 4,021% aus.

Mit der Funktion MSE,

```
MSE = function(x) {
MSE = sum(x^2)/length(x);
return(round(MSE,3))
 }
```

erhält man den mittleren quadratischen Prognosefehler

```
>source("MSE.R")
> MSE = MSE(BAU.err)
> MSE
[1] 0.826
```

und mit der Funktion RMSE

```
RMSE = function(x) {
RMSE = sqrt(sum(x^2)/length(x));
return(round(RMSE,3))
        }
```

die Wurzel des mittleren quadratischen Fehler

```
>source("RMSE.R")
> RMSE = RMSE(BAU.err)
> RMSE
[1] 0.909.
```

Allgemein sind MAE und RMSE in der Größenordnung auf gleichem Niveau. Auf-grund der größeren Ausreißersensitivität gilt hierbei jedoch die Ungleichung MAE ≤ RMSE.

Übungsaufgaben

3.4.1 Passen Sie die Zeitreihe DIENST in den Zeiträumen

- 1998/I bis 2010/IV,
- 1998/I bis 2011/I,
- 1998/I bis 2011/II,
- 1998/I bis 2011/3

mit dem in den Aufg. 3.3.1 und 3.3.2 identifiziertem ARIMA-Modell an! Interpretieren Sie die zeitliche Veränderung der Regressionskoeffizienten!

3.4.2 Erstellen Sie für die Zeitreihe DIENST Ein-Schritt-Prognosen mit den in Aufg. 3.4.1 durchgeführten Schätzungen des identifizierten ARIMA-Modells in den angegebenen Zeiträume!

3.4.3 Berechnen Sie die Prognosefehlermaße MAE, MAPE und RMSE für die in Aufg. 3.4.2 erstellten Ein-Schritt-Prognosen für die Zeitreihe DIENST!

3.4.4 Erstellen Sie für die Zeitreihe DIENST 1-, 2-, 3- und 4-Schritt-Prognosen für das Jahr 2011 auf der Basis der in Aufg. 3.4.1 erfolgten Modellschätzung im Zeitraum 1998/I bis 2010/IV und berechnen Sie die Prognosefehlermaße MAE, MAPE und RMSE!

3.4.5 Vergleichen Sie die Werte der Prognosefehlermaße der 1-, 2-, 3- und 4-Schritt-Prognose (s. Aufg. 3.4.4) mit denen der Ein-Schritt-Prognosen (s. Aufg. 3.4.3)! Interpretieren Sie die Unterschiede!

4. Nichtstationarität und Kointegration
4.1 Einheitswuzeltests (Unit-Root-Tests)

Bei der Spezifizierung von ARIMA-Modellen ist die Ordnung der Integration einer Zeitreihe anhand der ACF und PACF ermittelt worden. Hierzu sind zusätzlich eine Reihe von Einheitswurzeltests (Unit-Root-Tests) entwickelt worden, die vor allem in der Kointegrationsanalyse eingesetzt werden. Sie prüfen, ob der stochastische Prozess, der die Zeitreihe erzeugt, eine oder mehrere Einheitswurzeln besitzt oder nicht. Stochastische Prozesse mit Einheitswurzel sind nichtstationär und heißen integriert. Ohne Einheitswurzel ist ein Prozess stationär. In R sind verschiedene Einheitswurzeltests in dem Paket urca verfügbar.[2]

Der **Dickey-Fuller-Test** prüft die Nullhypothese eines Einheitswurzelprozesses gegen die Alternativhypothese eines stationären Prozesses. Während der standardmäßige Dickey-Fuller-Test (DF-Test) keine zeitlich verzögerten Terme in der Test-gleichung berücksichtigt, bezieht der erweiterte (augmented) Test (ADF-Test) Differenzen der Zeitreihe bis zu einem maximalen Lag k ein. Auf diese Weise soll einer evtl. Autokorrelation der Störgröße Rechnung getragen werden.

Vor Ausführung des Dickey-Fuller-Tests wird das R-Paket urca geladen:

```
> library(urca).
```

Anschließend ist die Funktion ur.df zur Durchführung des Dickey-Fuller-Tests verfügbar. Da die Zeitreihe BAU eine trendbehaftete Zeitreihe ist, wird das Argument type gleich "trend" gesetzt. Mit der Angabe der Laglänge lags=0 der standardmäßige Dickey-Fuller-Test (DF-Test) durchgeführt:

```
> BAU.df = ur.df(BAU, type="trend", lags=0)
> summary(BAU.df)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression trend

Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt)

Residuals:
   Min     1Q  Median     3Q    Max
-2.4458 -0.3151 -0.0861  0.3109  3.0354

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.54033    2.20207   2.970  0.00450 **
z.lag.1     -0.21781    0.07528  -2.894  0.00555 **
tt          -0.03499    0.01071  -3.268  0.00192 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[2] Zwar lassen sichmit der Funktion adf.test des Pakets tseries ebenfalls Einheitswurzeltests durchführen, doch besteht hierbei keine Möglichkeit einer Differenzierung der Form der Testgleichung z.B. mit oder ohne Trend bzw. mit oder ohne Konstante.

```
Residual standard error: 0.7498 on 52 degrees of freedom
Multiple R-squared: 0.173,     Adjusted R-squared: 0.1412
F-statistic: 5.441 on 2 and 52 DF,  p-value: 0.007154

Value of test-statistic is: -2.8935 3.8693 5.4406

Critical values for test statistics:
   1pct  5pct 10pct
tau3 -4.04 -3.45 -3.15
phi2  6.50  4.88  4.16
phi3  8.73  6.49  5.47.
```

Die übliche Prüfgröße des DF-Tests ist der erste Wert der in der unteren Hälfte des R-Outputs berechneten Teststatistiken:[3]

Value of test-statistic is: -2.8935.

Zur Prüfung der Nullhypothese einer Einheitswurzel ist er mit einem der unter tau3 ausgewiesenen kritischen Werte zu vergleichen. Bei einem Signifikanzniveau von 5% ist dies der Wert -3.45. Wegen

$$t = -2{,}8935 \quad > \quad tau3_{0.05} = -3{,}45$$

kann die Nullhypothese einer Einheitswurzel für die Zeitreihe BAU auf dem 5%-Signifkanzniveau nicht abgelehnt werden. Ein Vergleich mit den beiden anderen kritischen Werten zeigt, dass die Nichtstationarität der Bauproduktion auf keinem gängigen Signifikanzniveau α abgelehnt werden kann.

Führt man den DF-Test für die ersten Differenzen der Zeitreihe, ΔBAU, mit der Spezifikation type="drift"durch, ergibt sich dagegen ein anderes Bild:

```
> DBAU.df = ur.df(diff(BAU), type="drift", lags=0)
> summary(DBAU.df)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift

Call:
lm(formula = z.diff ~ z.lag.1 + 1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.80591 -0.39188 -0.03824  0.35411  2.43408
```

[3] Die beiden anderen ausgewiesenen Werte von 3,8693 und 5,4406sind F-Statistiken, die sich auf einen simultanen Test der Nullhypothese (ρ=1, β=0) bzw. (ρ=1, β=0, μ=0) beziehen. Die kritischen Werte finden sich hierzu unter phi2 bzw. phi3. Insbesondere bestätigt sich in der Beibehaltung der Nullhypothese (ρ=1, β=0), dass die Zeitreihe BAU nicht durch einen deterministischen, sondern einen stochastischen Trend getrieben wird.

```
Coefficients:
        Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.1219    0.1041  -1.171    0.247
z.lag.1     -1.3873    0.1279 -10.845 5.87e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7603 on 52 degrees of freedom
Multiple R-squared: 0.6934,    Adjusted R-squared: 0.6875
F-statistic: 117.6 on 1 and 52 DF,  p-value: 5.87e-15

Value of test-statistic is: -10.8445 58.802

Critical values for test statistics:
1pct  5pct 10pct
tau2 -3.51 -2.89 -2.58
phi1  6.70  4.71  3.86.
```

Für die differenzierte Zeitreihe, ΔBAU, wird die Nullhypothese einer Einheitswurzel auf dem 5%-Niveau verworfen, da der Wert der üblichen Prüfgröße des DF-Tests,

Value of test-statistic is: -10.8445

den kritischen Wert tau2 von -2.89 unterschreitet:[4]

$$t = -10{,}8445 < \text{tau2}_{0.05} = -2{,}89.$$

Dies trifft für alle drei gängigen Signifikanzniveaus α zu. Aufgrund der Ergebnisse des DF-Tests lässt sich daher schließen, dass die Zeitreihe der Bauproduktion eine Einheitswurzel besitzt.

Da das Testergebnis durch eine potenzielle Autokorrelation der Störgröße beeinflusst wird, liefert der erweiterte (augmented) Dickey-Fuller-Test (ADF-Test) eine bessere Basis für die Beurteilung der Nichtstationarität. Unter Vorgabe eines maximalen Lags von 4 Quartalen bestimmen wir dabei die Laglänge anhand der signifikanten verzögerten Differenzen der Bauproduktion, was hier zu einer Laglänge von 1 führt. Für die langfristig wachsende Niveaureihe BAU ergibt sich mit den Argumenten type="trend" und lags=1

```
> BAU.adf = ur.df(BAU, type="trend", lags=1)
> summary(BAU.adf)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend

Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
```

[4] Der ausgewiesene Wert der Teststatistik von 58,802 ist die F-Statistik des simultanen Tests der Nullhypothese (ρ=1, μ=0). Ein Vergleich mit dem kritischen Werte phi1 bestätigt eine Ablehnung der Nullhypothese einer Einheitswurzel und eines Mittelwerts von 0 für die differenzierte Zeitreihe der Bauproduktion.

```
Residuals:
   Min      1Q  Median      3Q     Max
-2.73698 -0.33407 -0.01229  0.32655  2.03421

Coefficients:
        Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.17407    2.23242   2.766 0.007938 **
z.lag.1     -0.19999    0.07590  -2.635 0.011175 *
tt          -0.03892    0.01080  -3.602 0.000725 ***
z.diff.lag  -0.37341    0.12226  -3.054 0.003613 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6891 on 50 degrees of freedom
Multiple R-squared: 0.3285,    Adjusted R-squared: 0.2882
F-statistic: 8.152 on 3 and 50 DF,  p-value: 0.0001618

Value of test-statistic is: -2.6349 4.99 6.6498

Critical values for test statistics:
1pct  5pct 10pct
tau3 -4.04 -3.45 -3.15
phi2  6.50  4.88  4.16
phi3  8.73  6.49  5.47.
```

Für die ersten Differenzen der Bauproduktion, ΔBAU, ergibt sich mit der Spezifikation type="drift"bis zu einem maximalen Lag von 4 kein signifikanter zeitlich verzögerter Differenzenterm, so dass der ADF-Test hier mit dem DF-Test identisch ist. Insgesamt bestätigt damit der ADF-Test das Ergebnis des DF-Tests. Während für die Niveauwerte der Bauproduktion die Nullhypothese einer Einheitswurzel nicht verworfen werden kann, wird sie für die differenzierten Werte auf allen gängigen Signifikanzniveaus abgelehnt. Demzufolge ist die Zeitreihe der Bauproduktion nichtstationär. Sie besitzt eine Einheitswurzel, d.h. sie wird durch einen I(1)-Prozess erzeugt.

Die Nullhypothese einer Einheitswurzel in dem einer Zeitreihe zugrundeliegenden stochastischen Prozess kann alternativ mit dem **Phillips–Perron-Test** geprüft werden. Im Unterschied zum Dickey-Fuller-Test wird hierbei allerdings einer potenziellen Autokorrelation der Störterme nicht durch Aufnahme zeitlich verzögerter Variablen in die Testgleichung Rechnung getragen. Autokorrelation und Heteroskedastizität der Störmterme werden nichtparametrisch durch Korrektur des Varianzschätzers (Newey-West-Schätzer) berücksichtigt.

Der Phillips–Perron-Test kann unter Verwendung der Funktion ur.ppt ausgeführt werden, die dem bereits geladenen R-Pakt urca verfügbar ist.[5] Mit der Funktion ur.pp lassen sich zwei Teststatistiken, Z-alpha und Z-tau, berechnen, die unter der Nullhypothese einer Einheitswurzel dieselbe asymptotische Verteilung wie die ADF-Statistik haben. Außerdem lässt sich spezifizieren, ob bei der Berechnung des Varianzschätzers ein kurzer oder langer Lag einbezogen werden soll (Voreinstellung:

[5] Bei Verwendung der Funktion pp.test des Pakets tseries wird stets eine Konstante und ein linearer Trend in die Testgleichung einbezogen.

lags="short"). Unter Verwendung der Argumente type="Z-tau" und model="trend" erhält man für die Zeitreihe BAU das Testergebnis

```
> BAU.pp = ur.pp(BAU, type="Z-tau", model="trend")
> summary(BAU.pp)
###################################
# Phillips-Perron Unit Root Test #
###################################
Test regression with intercept and trend

Call:
lm(formula = y ~ y.l1 + trend)

Residuals:
   Min     1Q  Median     3Q    Max
-2.4458 -0.3151 -0.0861  0.3109  3.0354


Coefficients:
         Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.57813    1.95847   2.848  0.00628 **
y.l1         0.78219    0.07528  10.391 2.72e-14 ***
trend       -0.03499    0.01071  -3.268  0.00192 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7498 on 52 degrees of freedom
Multiple R-squared: 0.904,     Adjusted R-squared: 0.9003
F-statistic: 244.8 on 2 and 52 DF,  p-value: < 2.2e-16

Value of test-statistic, type: Z-tau  is: -2.6131

aux. Z statistics
Z-tau-mu           5.2120
Z-tau-beta        -3.7981

Critical values for Z statistics:
1pct     5pct    10pct
critical values -4.131355 -3.491931 -3.174369,
```

das das Ergebnis des ADF-Tests bestätigt.

Für die differenzierte Zeitreihe der Bauproduktion, diff(BAU), ist das Argument model gleich „constant" zu setzen, womit sich das Testergebnis

```
> DBAU.pp = ur.pp(diff(BAU), type="Z-tau", model="constant")
> summary(DBAU.pp)
```

```
###################################
# Phillips-Perron Unit Root Test #
###################################

Test regression with intercept

Call:
lm(formula = y ~ y.l1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.80591 -0.39188 -0.03824  0.35411  2.43408

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.1219     0.1041  -1.171  0.24678
y.l1         -0.3873     0.1279  -3.027  0.00383 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7603 on 52 degrees of freedom
Multiple R-squared: 0.1498,    Adjusted R-squared: 0.1335
F-statistic: 9.164 on 1 and 52 DF,  p-value: 0.003833

Value of test-statistic, type: Z-tau  is: -10.9226

aux. Z statistics
Z-tau-mu          -1.1799

Critical values for Z statistics:
1pct     5pct    10pct
critical values -3.554623 -2.915671 -2.595266
```

ergibt, das ebenfalls mit demjenigen des ADF-Tests in Einklang ist. Auch der Phillips-Perron-Test bestätigt damit, dass die Bauproduktion einem Einheitswurzelprozess folgt.

Übungsaufgaben

4.1.1 Prüfen Sie unter Berücksichtigung der Vorabinformation einer langfristig wachsenden oder nichtwachsenden Zeitreihe mit dem

- DF-Test,
- ADF-Test,

ob die Dienstleistungsproduktion (Zeitreihe DIENST) einem nichtstationärem Prozess folgt (α=0,05)! Bestimmen Sie beim ADF-Test ausgehend von einem maximalen Lag von 4 die Laglänge durch die signifikanten zeitlich verzögerten Differenzen der Dienstleistungsproduktion (α=0,05)!

4.1.2 Geben Sie die geschätzten Testgleichungen der in Aufg. 4.1.1 durchgeführten Dickey-Fuller-Tests wieder und erläutern Sie die Unterschiede! Wie lässt sich die Erweiterung der Testgleichung des ADF-Tests gegenüber dem DF-Test interpretieren und was bezweckt sie?

4.1.3 Welche Nullhypothese liegt dem zweiten bei Value of test-statistic is: ausgewiesenen Wert des in Aufg. 4.1.1 durchgeführten ADF-Tests zugrunde? Wie fällt das Testergebnis hier aus (α=0,05)?

4.1.4 Prüfen Sie in Fortsetzung der in Aufg. 4.1.1 begonnenen Testprozedur, ob die differenzierte Zeitreihe der Dienstleistungsproduktion eine Einheitswurzel besitzt oder nicht (α=0,05)! Bestimmen Sie beim ADF-Test wiederum ausgehend von einem maximalen Lag von 4 die Laglänge anhand der Signifikanz der zeitlich verzögerten Differenzenterme (α=0,05)!

4.1.5 Welche Schlüsse über die Nichtstationarität der Dienstleistungsproduktion lassen auf der Grundlage des Phillips-Perron-Tests ziehen (α=0,05)?

4.2 Kointegration

Eine Regressionsfunktion zweier nichtstationärer Variablen lässt sich nur dann als Gleichgewichtsbeziehung im Sinne der ökonomischen Theorie interpretieren, wenn sie kointegriert sind. Voraussetzung hierzu ist, dass beide Variablen dieselbe Integrationsordnung besitzen. Hier wird beispielhaft eine Kointegration zwischen dem Privaten Konsum und dem verfügbaren Einkommen mit dem Engle-Granger-Verfahren geprüft. Die Anpassung an das Gleichgewicht und die kurzfristige Dynamik werden in einem Fehlerkorrekturmodell (Error-Correction-Modell (ECM)) abgebildet.

Zur Prüfung der Kointegration verwenden wir saisonbereinigte Quatalsdaten des Privaten Konsums und verfügbaren Einkommens für den Zeitraum 1991/I bis 1995/IV, die aus der Datei konsumfkt_q.csv eingelesen werden:

```
> konsumfkt_q = read.csv(file="konsumfkt_q.csv", dec=".", sep=";", header=TRUE)
> pkons = ts(konsumfkt_q$pkonsum, start=c(1991,1), end=c(1995,4), freq=4)
> class(pkons)
[1] "ts"
> pkons
   Qtr1  Qtr2  Qtr3  Qtr4
1991 70.44 72.32 71.89 75.22
1992 76.28 77.04 77.60 80.93
1993 79.38 80.33 81.72 83.58
1994 84.44 83.97 85.11 86.18
1995 86.60 88.04 88.35 88.59

> veink = ts(konsumfkt_q$veink, start=c(1991,1), end=c(1995,4), freq=4)
> class(veink)
[1] "ts"
> veink
 Qtr1  Qtr2  Qtr3  Qtr4
1991 73.34 76.11 75.10 78.14
1992 79.47 80.22 81.13 83.75
1993 82.41 83.73 84.51 86.61
1994 86.58 86.31 87.56 88.99
1995 88.73 90.00 90.38 90.29
```

Im ersten Schritt des Engle-Granger-Verfahrens werden die Integrationsordnung des Privaten Konsums, pkons, und des verfügbaren Einkommens, veink, mit Hilfe des Dickey-Fuller-Tests bestimmt. Bei Vorgabe eines maximalen Lags von 4 zeigt sich, dass keine zeitlich verzögerten Terme der abhängigen Variablen in der Testgleichung signifikant sind, so dass der ADF-Test mit dem DF-Test übereinstimmt.

Die DF-Tests der Niveauvariablen pkons und ihrer Veränderungen diff(pkons) lassen für den Privaten Konsum auf eine Einheitswurzel schließen:

```
> library(urca)
> pkons.df = ur.df(pkons, type="trend", lags=0)
> summary(pkons.adf)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression trend
Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt)

Residuals:
    Min      1Q  Median      3Q     Max
-2.06090 -0.46992  0.05502  0.29235  2.25099

Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  55.3401    17.0345    3.249   0.00503 **
z.lag.1      -0.7619     0.2407   -3.165   0.00600 **
tt            0.6941     0.2380    2.916   0.01010 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9551 on 16 degrees of freedom
Multiple R-squared: 0.4181,    Adjusted R-squared: 0.3453
F-statistic: 5.748 on 2 and 16 DF,  p-value: 0.01315

Value of test-statistic is: -3.1652 10.168 5.7478

Critical values for test statistics:
      1pct  5pct  10pct
tau3 -4.38 -3.60 -3.24
phi2  8.21  5.68  4.67
phi3 10.61  7.24  5.91

> dpkons.df = ur.df(diff(pkons), type="drift", lags=0)
> summary(dpkons.adf)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression drift
Call:
lm(formula = z.diff ~ z.lag.1 + 1)
```

```
Residuals:
   Min     1Q  Median     3Q    Max
-1.4348 -0.8133 -0.2538  0.4360  2.2298

Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)   1.353      0.339     3.991   0.00105 **
z.lag.1      -1.451      0.221    -6.567   6.48e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.095 on 16 degrees of freedom
Multiple R-squared: 0.7294,    Adjusted R-squared: 0.7125
F-statistic: 43.13 on 1 and 16 DF,  p-value: 6.479e-06

Value of test-statistic is: -6.5673 21.6273

Critical values for test statistics:
      1pct  5pct  10pct
tau2 -3.75 -3.00 -2.63
phi1  7.88  5.18  4.12.
```

Gleichermaßen kann aus den DF-Tests der Variablen veink und diff(veink) beim verfügbaren Einkommen auf die Existenz einer Einheitswurzel geschlossen werden:

```
> veink.adf = ur.df(veink, type="trend", lags=0)
> summary(veink.adf)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend

Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt)

Residuals:
   Min     1Q  Median     3Q    Max
-2.4193 -0.4717  0.1767  0.3024  1.8957

Coefficients:
            Estimate   Std. Error  t value  Pr(>|t|)
(Intercept) 53.8223    16.8369    3.197   0.00562 **
z.lag.1     -0.7036     0.2265   -3.106   0.00679 **
tt           0.5694     0.2083    2.734   0.01472 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9773 on 16 degrees of freedom
Multiple R-squared: 0.4321,    Adjusted R-squared: 0.3611
F-statistic: 6.086 on 2 and 16 DF,  p-value: 0.01083

Value of test-statistic is: -3.1063 9.3343 6.0858
```

```
Critical values for test statistics:
      1pct  5pct  10pct
tau3  -4.38 -3.60 -3.24
phi2   8.21  5.68  4.67
phi3  10.61  7.24  5.91

> dveink.df = ur.df(diff(veink), type="drift", lags=0)
> summary(dveink.adf)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression drift

Call:
lm(formula = z.diff ~ z.lag.1 + 1)

Residuals:
   Min     1Q  Median     3Q    Max
-1.5352 -0.7550 -0.1200  0.6365  1.8143

Coefficients:
            Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  1.2505     0.3095     4.040   0.000949 ***
z.lag.1     -1.4888     0.2026    -7.349   1.64e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.031 on 16 degrees of freedom
Multiple R-squared: 0.7715,    Adjusted R-squared: 0.7572
F-statistic: 54.01 on 1 and 16 DF,  p-value: 1.638e-06

Value of test-statistic is: -7.3492 27.2191

Critical values for test statistics:
      1pct  5pct  10pct
tau2 -3.75 -3.00 -2.63
phi1  7.88  5.18  4.12.
```

Da beide Zeitreihen pkons und veink I(1)-Variablen sind, d.h. die Integrationsordnung eins besitzen, lässt sich im zweiten Schritt die makroökonomische Konsumfunktion,

(4.1) $Y_t = \beta_1 + \beta_2 \cdot X_t + U_t$,

als potenzielle Langfristbeziehung zwischen dem Privaten Konsum (Y) und dem verfügbaren Einkommen (X) mit der gewöhnlichen Methode der kleinsten Quadrate (ordinary least squares (OLS)) schätzen:

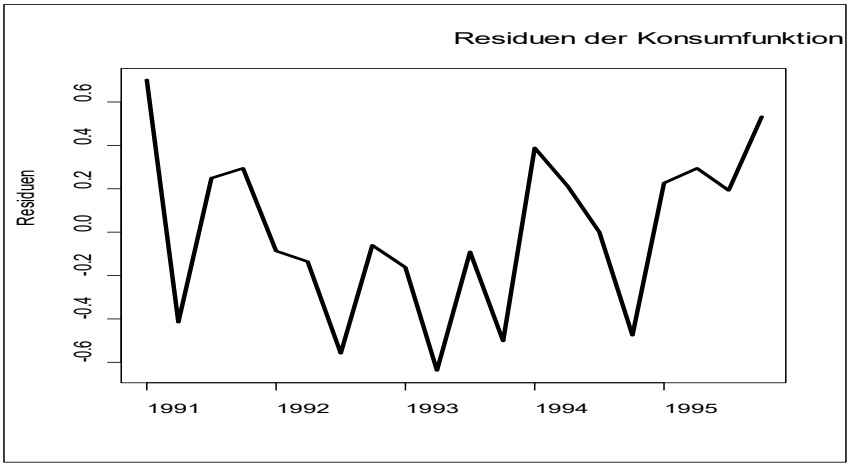
```
> konsfktq.lm = lm(pkons~veink)
> summary(konsfktq.lm)

Call:
lm(formula = pkons ~ veink)
```

```
Residuals:
   Min      1Q  Median      3Q     Max
-0.63751 -0.22358 -0.02806  0.26093  0.70205
Coefficients:
            Estimate  Std. Error  t value   Pr(>|t|)
(Intercept) -9.52829   1.41921   -6.714    2.7e-06 ***
veink        1.08080   0.01693   63.838   < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3891 on 18 degrees of freedom
Multiple R-squared: 0.9956,    Adjusted R-squared: 0.9954
F-statistic:  4075 on 1 and 18 DF,  p-value: < 2.2e-16.
```

Die Residuen der makroökonomischen Konsumfunktion sind in Abbildung 4.1 wiedergegeben:

```
> konsfktq.res = resid(konsfktq.lm)
> konsfktq.res = ts(konsfktq.res, start=c(1991,1), end=c(1995,4), freq=4)
> plot(konsfktq.res, main="Residuen der Konsumfunktion", xlab="", ylab="Residuen", lwd=3,
cex.main=1.25, font.main=1)
> box(which="figure").
```

Abbildung 4.1: Residuen der Konsumfunktion



Ihre Stationarität ist im dritten Schritt zu überprüfen. Da sie einen Mittelwert von null besitzen, ist das Argument type in der Funktion ur.df gleich "none" zu setzen. Erneut ist der Dickey-Fuller-Test ohne zeitlich verzögerte Terme der abhängigen Variablen durchzuführen:

```
> konsfktqres.df = ur.df(konsfktq.res, type="none", lags=0)
> summary(konsfktqres.adf)
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################
Test regression none
Call:
lm(formula = z.diff ~ z.lag.1 - 1)
```

```
Residuals:
   Min      1Q  Median      3Q     Max
-0.63757 -0.28619 -0.05934  0.23898  0.53248

Coefficients:
        Estimate  Std. Error  t value  Pr(>|t|)
z.lag.1  -1.0004     0.2254   -4.439   0.000317 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3522 on 18 degrees of freedom
Multiple R-squared: 0.5226,    Adjusted R-squared: 0.4961
F-statistic:  19.7 on 1 and 18 DF,  p-value: 0.000317

Value of test-statistic is: -4.4389

Critical values for test statistics:
1pct  5pct 10pct
tau1 -2.66 -1.95  -1.6.
```

Da die Residuen nicht beobachtbar, sondern geschätzt sind, können bei dem Test auf Kointegration nicht die mit der Funktion urca ausgewiesenen kritischen Werte benutzt werden. Sie können jedoch mit der Gleichung

$$CV(\alpha) = c_\infty + c_1 \cdot n^{-1} + c_2 \cdot n^{-2}$$

unter Verwendung der Koeffizienten bestimmt werden, die McKinnon (1991) durch Response-Surface-Simulationen ermittelt hat. Bei einer Kointegrationsbeziehung ohne Trend erhält man für α=0,05 mit den Koeffizienten c_∞=-3,34, c_1=-5,97 und c_2= -8,98 bei einem adjustierten Sichtprobenumfang von n=19 den kritischen Wert

```
> n = 19
> cu = -3.34
> c1 = -5.97
> c2 = -8.98
> CV.05 = cu + c1/n + c2/n^2
> CV.05
[1] -3.679086.
```

Da der Wert der Prüfgröße t von -4,4389 das 0,05-Quantil ihrer simulierten Vertei-lung unterschreitet,

$$t = -4{,}4389 \quad < \quad CV(0{,}05) = -3{,}67,$$

wird die Nullhypothese einer Einheitswurzel der Residuen der makroökonomischen Konsumfunktion wird auf dem 5%-Signifikanzniveau zurückgewiesen. Die Residuen sind damit von der Integrationsordnung null, so dass auf einen stationären Prozess der Störvariablen geschlossen werden kann. Aufgrund der Testergebnisse des Granger-Engle-Verfahrens lässt sich die makroökonomische Konsumfunktion daher als eine Langfristbeziehung im Sinne der ökonomischen Theorie interpretieren. Der normierte Kointegrationsvektor ist hier durch (1 9,528 -1,081) gegeben.

In einem Fehlerkorrekturmodell wird die Veränderung des Konsums in Abhängigkeit der Gleichgewichtsabweichung in der Vorperiode, u_{t-1}, der kontemporären Verände-

rung des verfügbaren Einkommens, ΔX_t, sowie der zeitlich verzögerten Differenzen des Privaten Konsums, ΔY_{t-i}, und des verfügbaren Einkommens, ΔX_{t-i}, $i \geq 1$, erklärt:

$$(4.2)\quad \begin{aligned} \Delta Y_t &= \mu + \gamma u_{t-1} + \alpha_1 \Delta Y_{t-1} + \ldots + \alpha_p \Delta Y_{t-p} \\ &\quad + \beta_0 \Delta X_t + \beta_1 \Delta X_{t-1} + \ldots + \beta_q \Delta X_{t-q} + V_t \end{aligned}$$

Der Regressionskoeffizient γ des Fehlerkorrekturterms u_{t-1} heißt Anpassungskoeffizient. Er misst die Geschwindigkeit, mit der Abweichungen vom Gleichgewicht zurückgebildet werden. Hierzu muss er einen negativen Wert annehmen. Die Differenzenterme ΔY_{t-i}, $i \geq 1$, und, ΔX_{t-i}, $i \geq 0$, bilden die kurzfristige Dynamik ab.

Bei Vorgabe einer maximalen Lagordnung von 4 erweist sich bei der kurzfristigen Dynamik allein die kontemporäre Veränderung des verfügbaren Einkommens als signifikant. Um das Fehlerkorrekturmodell für den Privaten Konsum mit der Funktion lm zu schätzen, ist auf eine einheitliche Länge der involvierten Zeitreihen zu achten. Die Zeitreihen der ersten Differenzen des Konsums und Einkommens, dpkons und dveink, lassen sich hierzu für den Zeitraum 1991/II bis 1995/IV einfach mit der Funktion diff bilden:

```
> dpkons = diff(pkons)
> dpkons
      Qtr1   Qtr2   Qtr3   Qtr4
1991         1.88  -0.43   3.33
1992  1.06   0.76   0.56   3.33
1993 -1.55   0.95   1.39   1.86
1994  0.86  -0.47   1.14   1.07
1995  0.42   1.44   0.31   0.24
```

und

```
> dveink = diff(veink)
> dveink
      Qtr1   Qtr2   Qtr3   Qtr4
1991         2.77  -1.01   3.04
1992  1.33   0.75   0.91   2.62
1993 -1.34   1.32   0.78   2.10
1994 -0.03  -0.27   1.25   1.43
1995 -0.26   1.27   0.38  -0.09.
```

Bei der Bestimmung der zeitlich verzögerten Residualreihe u1 wird von der lag-Funktion Gebrauch gemacht, bei der angeben werden muss, wie viel Perioden der Wert einer Zeitreihe, der der Periode t zugeordnet werden soll, in der Vergangenheit zurückliegt (negative Zahl):[6]

```
> u1 = lag(konsfktq.res, -1)
> u1
```

[6] Wird ein zeitlich vorlaufender Zeitreihenwert der Periode t zugeordnet, spricht man von einem Lead. Er lässt sich ebenfalls mit der lag-Funktion unter Angabe einer positiven Vorlaufszahl bilden.

```
          Qtr1          Qtr2          Qtr3          Qtr4
1991                0.702053951 -0.411775875  0.249837166
1992  0.294189992 -0.083280646 -0.133884390 -0.557416932
1993 -0.059126009 -0.160847321 -0.637509909 -0.090537803
1994 -0.500228285  0.392195865  0.214013213  0.003006974
1995 -0.472544164  0.228465134  0.295842795  0.195136898
1996  0.532409347.
```

Die zeitlich verzögerte Residualreihe u1 umfasst Werte für den Zeitraum von 1991/II bis 1996/I. Eine Eliminierung des letzten Wertes der Zeitreihe u1 bei der Regression kann direkt in der Funktion lm durch den Suffix [-20] erfolgen:

```
> konsfktq.ecm = lm(dpkons~u1[-20]+dveink).
```

Alternativ könnte die Variable u1 nach der Neudefiniton

```
> u1 = ts(u1, start=c(1991,2), end=c(1995,4), freq=4)
```

ohne Suffix in die Modellspezifikation aufgenommen werden. Bei größeren Regressionsmodellen lassen sich unterschiedlich lange Zeitreihen effizient mit der Funktion ts.intersect auf die gleiche Länge bringen.[7] Mit dieser Funktion werden Perioden eliminiert, in denen bei mindestens einer Variablen ein Wert fehlt (missing value). Hier würde man die multiple Zeitreihe u1dx bilden,

```
> u1dx = ts.intersect(u1, dveink)
> class(u1dx)
[1] "mts" "ts" "matrix"
> u1dx
              u1      dveink
1991 Q2  0.702053951  2.77
1991 Q3 -0.411775875 -1.01
1991 Q4  0.249837166  3.04
1992 Q1  0.294189992  1.33
1992 Q2 -0.083280646  0.75
1992 Q3 -0.133884390  0.91
1992 Q4 -0.557416932  2.62
1993 Q1 -0.059126009 -1.34
1993 Q2 -0.160847321  1.32
1993 Q3 -0.637509909  0.78
1993 Q4 -0.090537803  2.10
1994 Q1 -0.500228285 -0.03
1994 Q2  0.392195865 -0.27
1994 Q3  0.214013213  1.25
1994 Q4  0.003006974  1.43
1995 Q1 -0.472544164 -0.26
1995 Q2  0.228465134  1.27
```

[7] Mit der Funktion ts.union wird eine multiple Zeitreihe durch Einfügung der Kennzeichnung NA bei fehlenden Werten in einzelnen Zeitreihen gebildet.

```
1995 Q3  0.295842795  0.38
1995 Q4  0.195136898 -0.09,
```

bei der das erste Quartal 1996 eliminiert worden ist. Die multiple Zeitreihe u1dx kann in der Modellspezifikation nach der Tilde anstelle der Einzelvariablen, formula = dpkons~u1dx, eingesetzt werden.

Unter Verwendung der Zeitreihen u1[-20] und dveink als Regressoren in der Funktion lm ergibt sich folgende numerische Spezifizierung des Fehlerkorrekturmodells für den Privaten Konsum:

```
> summary(konsumfktq.ecm)

Call:
lm(formula = dpkons ~ u1[-20] + dveink)

Residuals:
 Min      1Q   Median      3Q     Max
-0.53835 -0.27324  0.05788  0.20187  0.55217

Coefficients:
             Estimate  Std. Error  t value  Pr(>|t|)
(Intercept)  0.06870    0.10084    0.681   0.50541
u1[-20]     -0.90342    0.22844   -3.955   0.00114 **
dveink       0.96541    0.06861   14.072  1.98e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3423 on 16 degrees of freedom
Multiple R-squared: 0.9252,    Adjusted R-squared: 0.9159
F-statistic: 99.01 on 2 and 16 DF,  p-value: 9.755e-10.
```

Der geschätzte Anpassungskoeffizient ist als Regressionskoeffizient des Fehlerkorrekturterms u1 hochsignifikant und hat erwartungsgemäß ein negatives Vorzeichen. Das bedeutet, dass sich Abweichungen von der Gleichgewichtsbeziehung zwischen Konsum und Einkommen im Zeitablauf zurückbilden. Sein Wert von 0,903 gibt an, dass im Mittel pro Quartal 90,3% der verbleibenden Abweichung abgebaut wird.

Ebenfalls hochsignifikant ist der Regressionskoeffizient der ersten Differenz des Einkommens. Der Wert von 0,965 besagt, dass sich kontemporäre Veränderungen des Einkommens mit Mittel zu 96,5% in eine gleichgerichtete Veränderung des Konsums niederschlagen.

Übungsaufgaben

4.2.1 Prüfen Sie mit dem ADF-Test unter Berücksichtigung der signifikanten zeitlich verzögerten Differenzen der abhängigen Variablen bei einer maximalen Laglänge von 4, ob die Integrationsordnung der Zeitreihe HANDEL mit derjenigen der Zeitreihe DIENST übereinstimmt (α=0,05)!

4.2.2 Beurteilen Sie anhand der Residuen der Regression der Variablen DIENST auf die Variable HANDEL, ob eine Kointegration zwischen den beiden Variablen besteht (α=0,05)!

4.2.3 Bilden Sie den dynamischen Anpassungsprozess der in Aufg. 4.2.2 betrachteten Kointegrationsbeziehung in Form eines Fehlerkorrekturmodells ab! Spezifi-zieren Sie hierbei die kurzfristige Dynamik anhand der Signifikanz der zeitlich verzögerten Differenzenterme bei einer maximalen Laglänge von 4 (α=0,05)!

4.2.4 Interpretieren Sie den Fehlerkorrekturmechanismus und die kurzfristige Dynamik des in Aufg. 4.2.3 numerisch spezifizierten Fehlerkorrekturmodells!

5. ARCH- und GARCH-Modelle
5.1 Volatilitätscluster und Nichtlinearitäten

Bei ökonometrischen Modellen wird ebenso wie bei den ARIMA-Modellen standardmäßig eine zeitkonstante Varianz der Störvariablen U_t (Homoskedastizität) vorausgesetzt:

(5.1) $\mathrm{Var}(U_t) = E(U_t^2) = \sigma^2$.

Die Annahme homoskedastischer Störvariablen ist jedoch insbesondere bei Preis- und Finanzmarktvariablen häufig nicht gerechtfertigt ist. Vielmehr sind die quadrierten Residuen als Varianzkomponenten im Zeitablauf korreliert. Es sind Cluster hoher quadrierter Residuen zu beobachten, die aus großen positiven und negativen Abweichungen mit oder ohne Vorzeichenwechsel auftreten können, die einen hohen Grad an Unsicherheit zu den entsprechenden Zeitpunkten widerspiegeln. Man spricht in diesem Fall von **Volatilitätsclustern**. Zu anderen Zeitpunkten ist dagegen ein geringerer „noise“ der Zeitreihe zu konstatieren, was durch eine niedrigere Varianz indiziert wird.

Um eine Analyse nicht äquidistanter Zeitreihen durchführen zu können, laden wir das Paket zoo,

```
> library(zoo).
```

Unterschiedliche zeitliche Abstände zwischen den Zeitreihenwerte liegen z.B. bei Tagesdaten vor, die nur an Werktagen oder Börsentagen ermittelt werden.

Um Volatilitätscluster und Nichtlinearitäten in Zeitreihen zu illustrieren verwenden wir die Tageskurse der KFH-Aktie für einen langen Zeitraum vom 4.1.1988 bis zum 19.3.1996 (n=2142). Den Aktienkursen werden US-amerikanische Datumsangaben zugeordnet, wenn in dem R-Kommando Sys.setlocale das Argument locale gleich C gesetzt wird:

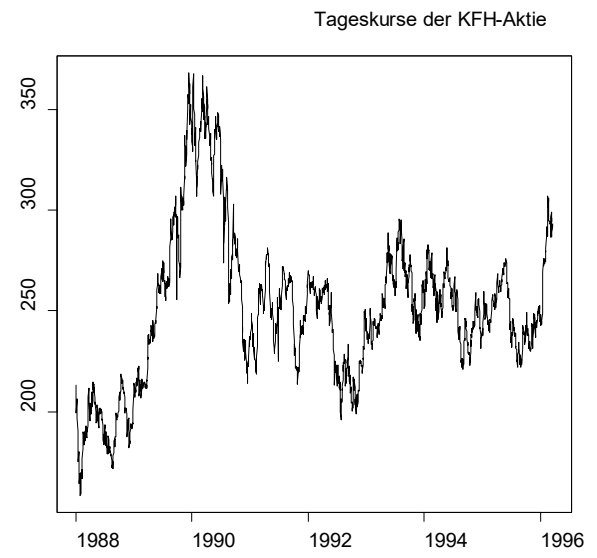
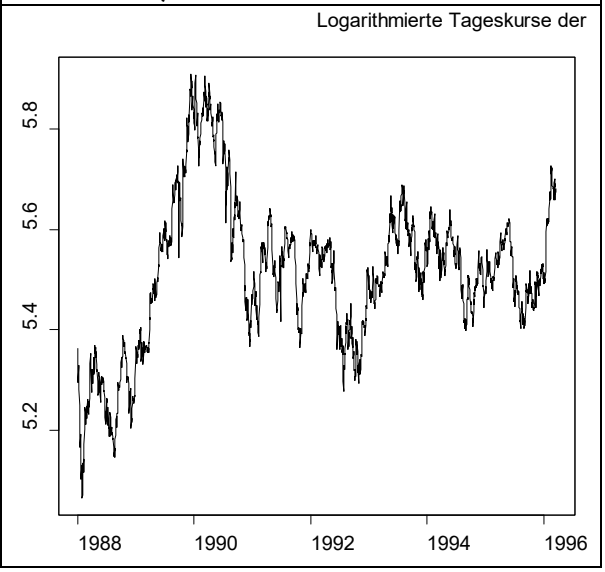
```
> Sys.setlocale("LC_TIME", locale = "C")
[1] "C"
```

Die Kursdaten, die in der Datei KFH-Kurse.csv gespeichert sind, lassen sich mit dem Befehl read.zoo als zoo-Object (irreguläre Zeitreihe) einlesen:

```
> KFH.zoo = read.zoo("KFH-Kurse.csv", dec=".", sep = ";", tz = "", header=TRUE)
> class(KFH.zoo)
[1] "zoo"
> length(KFH.zoo)
[1] 2142
> head(KFH.zoo)
1988-01-04 1988-01-05 1988-01-06 1988-01-07 1988-01-08 1988-01-11 1988-01-12
    213.31     199.40     205.33     206.05     201.45     196.59     190.46
1988-01-13 1988-01-14 1988-01-15
    184.07     181.00     175.12
```

```
> tail(KFH.zoo)
1996-03-06 1996-03-07 1996-03-08 1996-03-11 1996-03-12 1996-03-13 1996-03-14
    288.88     286.83     296.55     299.11     293.99     296.04     286.83
1996-03-15 1996-03-18 1996-03-19
    289.39     291.44     292.97
```

Ausgangspunkt der Illustration von Volatilitätsclustern sind die Zeitreihendiagramme der originären und logarithmierten Tageskurse der KFH-Aktie:

>plot(KFH.zoo, main="Tageskurse der KFH-Aktie", xlab="", ylab="", lwd=1.25, cex.main = 1.25, cex.axis = 1.25, font.main=1)	>plot(log(KFH.zoo), main="Logarithmierte Tageskurse der KFH-Aktie", xlab="", ylab="", lwd=1.25, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	

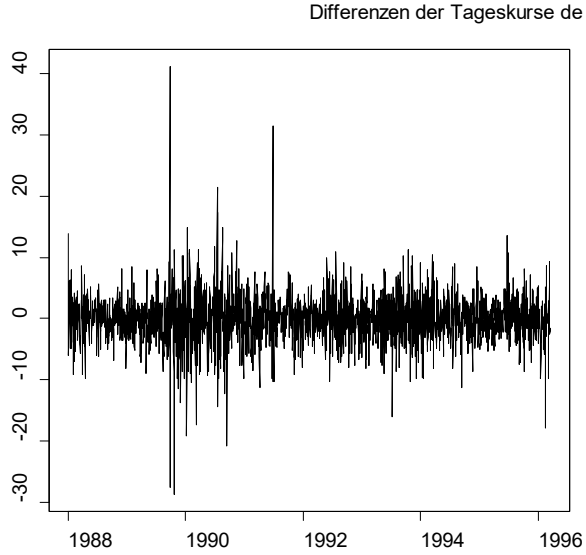
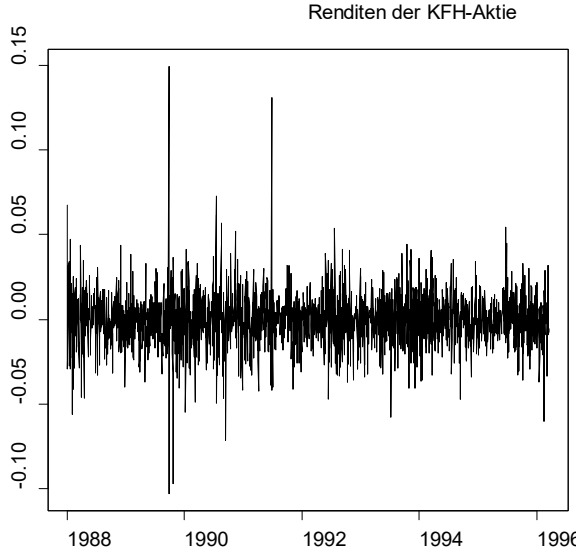
Beide Zeitreihen weisen die gleiche Struktur auf. Jedoch ist die Verwendung der logarithmierten Zeitreihenwerte bei Kursdaten vorteilhaft, da die ersten Differenzen unmittelbar die stetigen Renditen wiedergeben:

(5.2) $\Delta(\log Y_t) = \log Y_t - \log Y_{t-1} = \log(Y_t/Y_{t-1})$.

Bei der differenzierten originären und logarithmierten Zeitreihe der KFH-Kurse,

```
> DKFH.zoo = KFH.zoo - lag(KFH.zoo)
> class(DKFH.zoo)
[1] "zoo"
> length(DKFH.zoo)
[1] 2141
> DlnKFH.zoo = log(KFH.zoo) - log(lag(KFH.zoo))
> class(DlnKFH.zoo)
[1] "zoo"
> length(DlnKFH.zoo)
[1] 2141
```

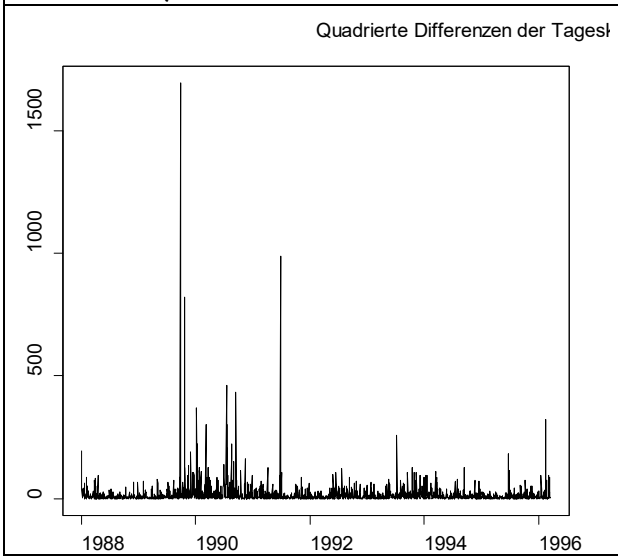
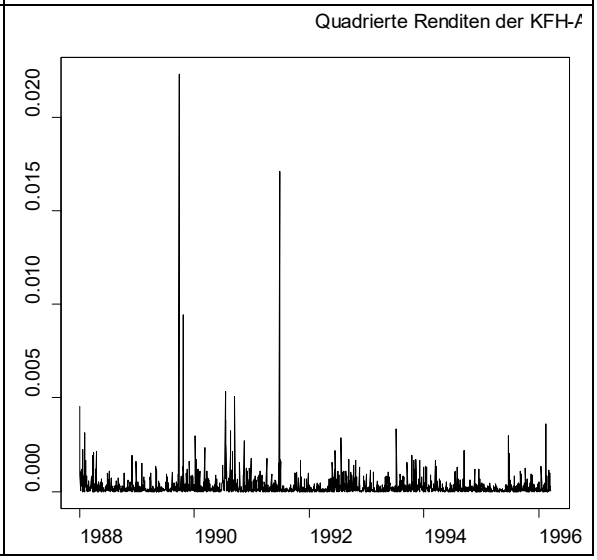
treten in den Zeitreihendiagrammen Auffälligkeiten hervor:

> plot(DKFH.zoo, main="Differenzen der Tageskurse der KFH-Aktie", xlab="", ylab="", lwd=1.25, cex.main = 1.25, cex.axis = 1.25, font.main=1)	> plot(DlnKFH.zoo, main="Renditen der KFH-Aktie", xlab="", ylab="", lwd=1.25, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	

Deutlich zu erkennen sind stärkere Kursveränderungen („Volalitätscluster“) insbesondere im III. Quartal 1989, im II. und III. Quartal 1990 und im II. Quartal 1991, im II. Quartal 1993 und im I. Quartal 1996.

Die Volatilitätscluster gehen besonders deutlich aus den Zeitreihen der quadrierten Kursdifferenzen und quadrierten logarithmierten Kursdifferenzen hervor:

```
> QDKFH.zoo = DKFH.zoo^2
> QDlnKFH.zoo = DlnKFH.zoo^2
```

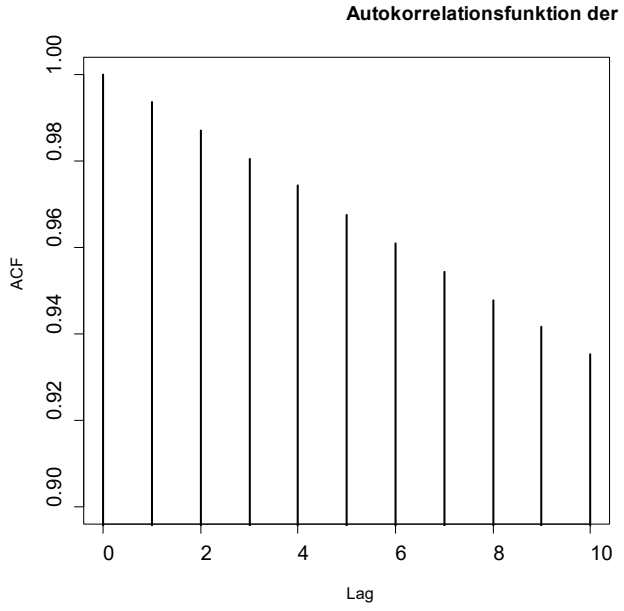
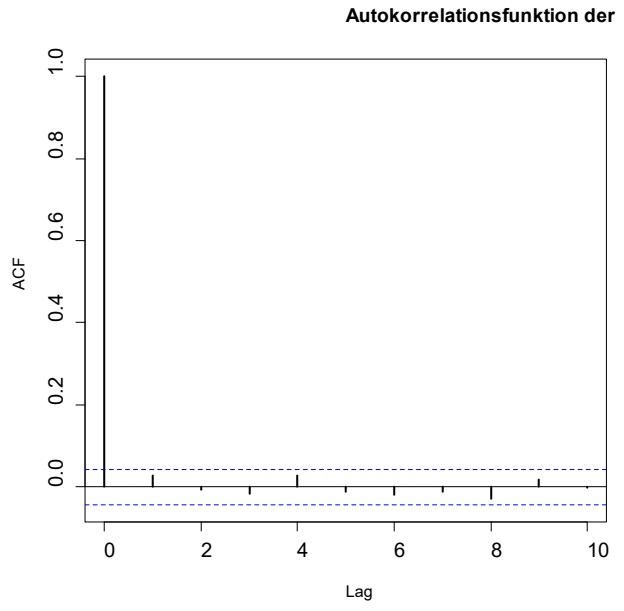
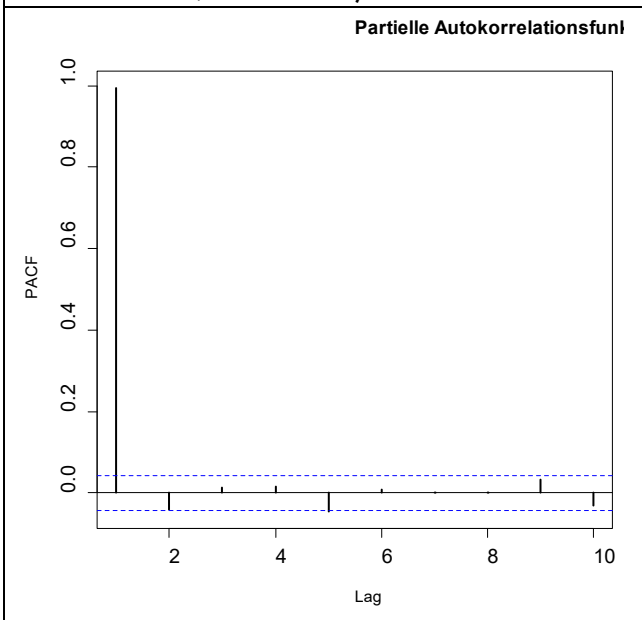
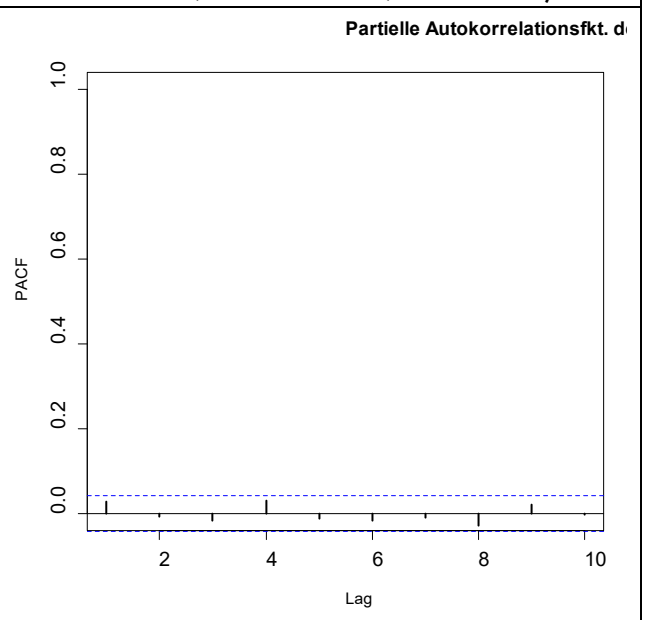
> plot(QDKFH.zoo, main="Quadrierte Differenzen der Tageskurse der KFH-Aktie", xlab="", ylab="", lwd=1.25, cex.main = 1.25, cex.axis = 1.25, font.main=1)	> plot(QDlnKFH.zoo, main="Quadrierte Renditen der KFH-Aktie", xlab="", ylab="", lwd=1.25, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	

Die Annahme gleicher Varianzen (Homoskedastizität) in allen Perioden ist bei der Existenz von Volatilitätsclustern nicht mehr gegeben. In bestimmten Phasen ist die Varianz, die das Risiko in den Aktienrenditen zum Ausdruck bringt, erheblich höher als in „normalen“ Phasen. Darin spiegelt sich die Zeitabhängigkeit der Varianz wider.

Nichtlineare Abhängigkeiten bei der KFH-Aktie lassen sich mit Hilfe von **Korrelogrammen** aufdecken. Zur Berechnung der Autokorrelationen und parteillen Autokorrelationen verwenden wir die Zeitreihenwerte und logarithmierten Zeitreihenwerte in Form von numerischen Vektoren:

```
> KFH = KFH.df$KFH
> KFH.acf = acf(KFH, plot=FALSE, lag.max=10)
> KFH.acf
Autocorrelations of series ‘KFH’, by lag
   0     1     2     3     4     5     6     7     8     9    10
1.000 0.994 0.987 0.981 0.974 0.968 0.961 0.954 0.948 0.942 0.935
> KFH.pacf = pacf(KFH, plot=FALSE, lag.max=10)
or
> KFH.pacf = acf(KFH, plot=FALSE, lag.max=10, type ="partial")
> KFH.pacf
Partial autocorrelations of series ‘KFH’, by lag
    1      2      3      4      5      6      7      8      9     10
 0.994 -0.042  0.013  0.015 -0.045  0.007  0.001  0.001  0.031 -0.032

> DKFH = as.numeric(DKFH.zoo)
> DKFH.acf = acf(DKFH, plot=FALSE, lag.max=10)
> DKFH.acf
Autocorrelations of series ‘DKFH’, by lag
    0      1      2      3      4      5      6      7      8      9     10
 1.000  0.028 -0.007 -0.017  0.028 -0.010 -0.019 -0.011 -0.028  0.019 -0.002
> DKFH.acf = pacf(DKFH, plot=FALSE, lag.max=10)
or
> DKFH.pacf = acf(DKFH, plot=FALSE, lag.max=10, type ="partial")
> DKFH.pacf
Partial autocorrelations of series ‘DKFH’, by lag
    1      2      3      4      5      6      7      8      9     10
 0.028 -0.008 -0.017  0.029 -0.012 -0.018 -0.009 -0.029  0.021 -0.003
```

Die **Korrelogramme der Tageskurse (Y_t) und ihrer Veränderungen (ΔY_t)** der KFH-Aktie weisen folgende Strukturen auf:

Tageskurse (Y_t) der KFH-Aktie	Kursdifferenzen (ΔY_t) der KFH-Aktie
> plot(KFH.acf, main="Autokorrelationsfunktion der KFH-Aktienkurse", xlab="Lag", ylab="ACF", ylim = c(0.9, 1), lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)	> plot(DKFH.acf, main="Autokorrelationsfunktion der KFH-Aktienkursdifferenzen", xlab="Lag", ylab="ACF", lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	
> plot(KFH.pacf, main="Partielle Autokorrelationsfunktion der KFH-Aktienkurse", xlab="Lag", ylab="PACF", lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)	> plot(DKFH.pacf, main="Partielle Autokorrelationsfkt. der KFH-Aktienkursdifferenzen", xlab="Lag", ylab="PACF", ylim = c(0, 1), lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	

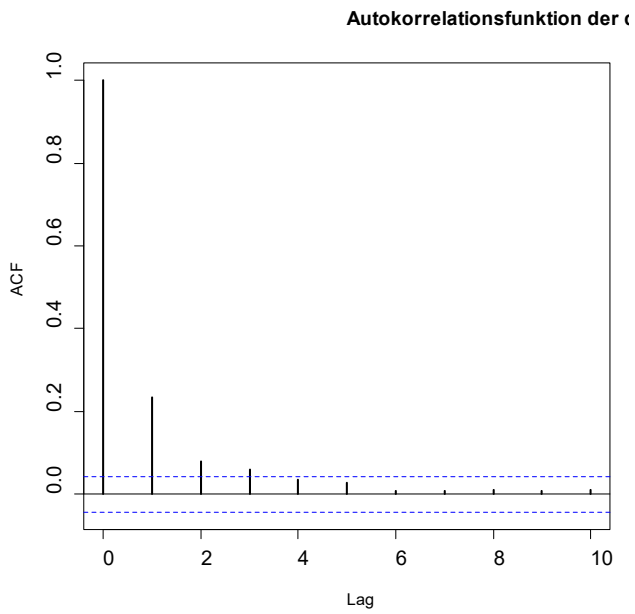
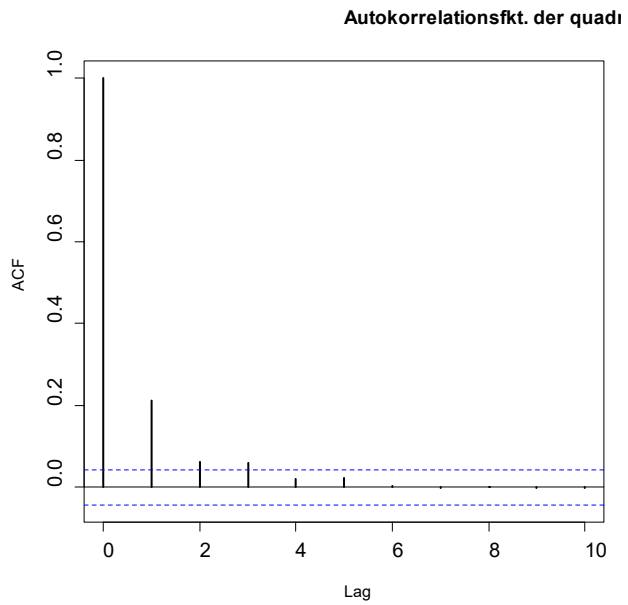
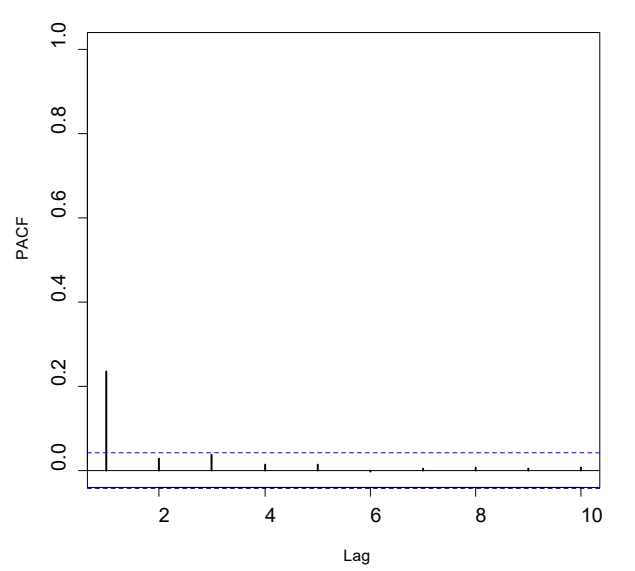
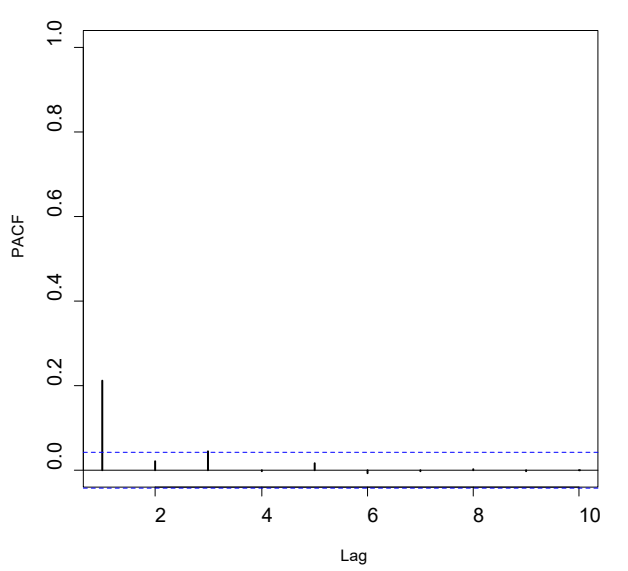
Obwohl aus dem Zeitreihendiagramm der Aktienkurse, (Y_t), kein deterministischer Trend erkennbar ist, sind alle Autokorrelationen bis zur hier überprüften Ordnung 10 hochsignifikant. Außerdem nehmen die Autokorrelationen in ihrer Größenordnung nur langsam ab, was für eine trendbehaftete Zeitreihe typisch ist. Es handelt sich hier um einen stochastischen Trend, der durch Differenzenbildung eliminiert werden kann.

Bei den Veränderungen der Aktienkurse, (ΔY_t), liegen dagegen alle Autokorrelationen und partielle Autokorrelationen innerhalb des 95%-Konfidenzbands, so dass sie nicht signifikant sind. Das bedeutet, dass die **differenzierte Zeitreihe** (ΔY_t) **keine linearen Abhängigkeiten** enthält. Dies ist dann der Fall, wenn die Zeitreihe (ΔY_t) die Realisation eines White-Noise-Prozesses ist.

Für die **quadrierten differenzierten Zeitreihen** der KFH-Aktie, $(\Delta Y_t)^2$ und $(\Delta \log Y_t)^2$, ergeben sich folgende Autokorrelationen und partielle Autokorrelationen:

```
> QDKFH = as.numeric(QDKFH.zoo)
> QDKFH.acf = acf(QDKFH, plot=FALSE, lag.max=10)
> QDKFH.acf
Autocorrelations of series ‘QDKFH’, by lag
    0     1     2     3     4     5     6     7     8     9    10
1.000 0.235 0.080 0.060 0.036 0.028 0.009 0.009 0.011 0.008 0.011
> QDKFH.pacf = pacf(QDKFH, plot=FALSE, lag.max=10)
or
> QDKFH.pacf = acf(QDKFH, plot=FALSE, lag.max=10, type ="partial")
> QDKFH.pacf
Partial autocorrelations of series ‘QDKFH’, by lag
    1     2     3     4     5     6     7     8     9    10
 0.235 0.026 0.037 0.013 0.014 -0.004 0.005 0.006 0.004 0.007

> QDlnKFH = as.numeric(QDlnKFH.zoo)
> QDlnKFH.acf = acf(QDlnKFH, plot=FALSE, lag.max=10)
> QDlnKFH.acf
Autocorrelations of series ‘QDlnKFH’, by lag
    0     1     2     3     4     5     6     7     8     9    10
 1.000 0.211 0.063 0.060 0.020 0.022 0.003 -0.002 0.001 -0.002 -0.001
> QDlnKFH.pacf = pacf(QDlnKFH, plot=FALSE, lag.max=10)
or
> QDlnKFH.pacf = acf(QDlnKFH, plot=FALSE, lag.max=10, type ="partial")
> QDlnKFH.pacf
Partial autocorrelations of series ‘QDlnKFH’, by lag
    1     2     3     4     5     6     7     8     9    10
 0.211 0.019 0.044 -0.003 0.016 -0.008 -0.003 0.001 -0.002 0.000
```

deren Strukturen grafisch anhand von Korrelogrammen ersichtlich werden:

Quadrierte Kursdifferenzen $(\Delta Y_t)^2$ der KFH-Aktie	Quadrierte logarithmierte Kurs-differenzen $(\Delta \log Y_t)^2$ der KFH-Aktie
> plot(QDKFH.acf, main="Autokorrelationsfunktion der quadr. KFH-Kursdifferenzen", xlab="Lag", ylab="ACF", lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)	> plot(QDlnKFH.acf, main="Autokorrelationsfkt. der quadr. log. KFH-Kursdifferenzen", xlab="Lag", ylab="ACF", lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	
> plot(QDKFH.pacf, main="Partielle Autokorrelationsfkt. der quadr. KFH-Kursdifferenzen", xlab="Lag", ylab="PACF", ylim = c(0, 1), lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)	> plot(QDlnKFH.pacf, main="Partielle Autokorr.fkt. der quadr. log. KFH-Kursdifferenzen", xlab="Lag", ylab="PACF", ylim = c(0, 1), lwd=2, cex.main = 1.25, cex.axis = 1.25, font.main=1)
	

Der Übersichtlichkeit halber sind die signifkanten Autokorrelationen und partiellen Autokorrelationen (5%-Signifikanzniveau) in einer Tabelle zusammengestellt:

Signifikante Autokorrelationen und partielle Autokorrelationen von$(\Delta Y_t)^2$ und $(\Delta \log Y_t)^2$ ($\hat{=}\ \hat{\rho}_\tau$ und $\hat{\phi}_{\tau\tau}$ außerhalb der 95%-Konfidenzbänder)			
ACF von $(\Delta Y_t)^2$	PACF von $(\Delta Y_t)^2$	ACF $(\Delta \log Y_t)^2$	PACF von $(\Delta \log Y_t)^2$
$\hat{\rho}_1$ = 0,235	$\hat{\phi}_{11}$ = 0,235	$\hat{\rho}_1$ = 0,211	$\hat{\phi}_{11}$ = 0,211
$\hat{\rho}_2$ = 0,080		$\hat{\rho}_2$ = 0,063	$\hat{\phi}_{33}$ = 0,045
$\hat{\rho}_3$ = 0,060		$\hat{\rho}_3$ = 0,060	

Obwohl die Kursdifferenzen (ΔY_t) und logarithmierten Kursdifferenzen ($\Delta \log(Y_t)$) (hier aus Platzgründen unterdrückt) „white noise“ sind, d.h. keine linearen Abhängigkeiten enthalten, trifft dies für die quadrierten Kursdifferenzen nicht zu. Nach Beseitigung des stochastischen Trends durch Differenzenbildung bleiben somit **nichtlineare Abhängigkeiten** in den Kursdifferenzen und Aktienrenditen bestehen. Das bedeutet, dass die Kursveränderungen und Renditen **unkorreliert, aber nicht unabhängig** sind.

Übungsaufgaben

5.1.1 Lesen Sie die Datei THY-Kurse.csv mit den Kursdaten der THY-Aktien als `zoo`-Objekt `THY.zoo` in R ein und ordnen Sie ihnen englische Datumsangaben zu!

5.1.2 Stellen Sie die Zeitreihen der logarithmierten Kurse (`lnTHY.zoo`) und der stetigen Renditen (`DlnTHY.zoo`) der THY-Aktie jeweils geeignet in einem Zeitreihendiagramm dar! Interpretieren Sie das Trendverhalten und die Volatilitäten!

5.1.3 Bestimmen Sie die Autokorrelations- und partiellen Autokorrelationsfunktionen der Zeitreihen `lnTHY.zoo` und `DlnTHY.zoo` und stellen Sie sie grafisch dar! Welche Informationen über den erzeugenden stochastischen Prozess lassen sich daraus gewinnen?

5.1.4 Berechnen Sie die Autokorrelations- und partiellen Autokorrelationsfunktion der quadrierten Renditen der THY-Aktie und geben Sie die signifikanten Koeffizienten an (α=0,05)! Interpretieren Sie den Befund in Bezug auf nichtlineare Abhängigkeiten bei der THY-Aktie!

5.2 ARCH-Modelle und ARCH-LM-Test

In diesen Situationen ist es zweckmäßiger, die Varianz der Störgröße zu einem Zeitpunkt t in Abhängigkeit von der Intensität der Störung zu modellieren, die prozessual entsteht und aktuell wirksam wird. Es handelt sich dann um eine bedingte Varianz bei dem zum Zeitpunkt t–1 gegebenen Informationsstand I_{t-1}. In einfachster Form ist die bedingte Varianz zum Zeitpunkt allein von der Störung der Vorperiode, u_{t-1}^2, abhängig:

(5.3) $\mathrm{Var}(U_t | I_{t-1}) = \sigma_t^2 = \alpha_0 + \alpha_1 \cdot u_{t-1}^2$,

Man spricht dann von einem ARCH(1)-Prozess, was als Abkürzung für einen Autoregressive Conditional Heteroskedastic-Prozess 1. Ordnung steht.

Zur ökonometrischen Schätzung von ARCH-Modellen sind in R verschiedene Pakete verfügbar. Die Schätzung standardmäßiger ARCH-Modelle (variance equation) ohne die zugehörige Mittelwertgleichung (mean equation) ist mit der Funktion garch des Pakets tseries möglich. Das Paket fGarch bietet mit der R-Funktion garchFit zusätzlich die Möglichkeit einer kombinierten Schätzung der mean equation und variance equation. Doch sind hierbei einige Probleme bei der nichtlinearen Schätzung der Modelparameter dokumentiert worden.[8] Mächtiger und umfangreicher sind die beiden R-Pakete rugarch und rmgarch, die eine ökonometrische Schätzung einer Vielzahl von Modellvarianten eines Autoregressive Conditional Heteroskedastic-Prozesses ermöglichen. Die Mittelwertgleichungen werden hierbei in Form von ARIMA-Modellen spezifiziert. Außerdem können auch andere erklärende Variablen (x-Variablen) berücksichtigt werden. Während rugarch für univariate GARCH-Modelle konzipiert worden ist, können multivariate GARCH-Modelle mit rmgarch analysiert werden. Da hier der Fokus auf univariate Zeitreihen liegt, laden wir nur das rugarch-Paket:

```
> library(rugarch)
```

Als erstes modellieren wir die logarithmierten Kurse der KFH-Aktien als autoregressiven Prozess 1. Ordnung (AR(1)-Prozess) in Verbindung mit einem ARCH(1)-Prozess der Störgröße. Diese Modellspezifikation lässt sich mit der R-Funktion ugarchspec vornehmen;[9]

```
> spec100_10 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 0)), mean.model
= list(armaOrder = c(1, 0), arfima = FALSE))
```

Die Modellschätzung mit dem ugarchfit-Befehl,

```
> lnKFH.arch1AR1 = ugarchfit(spec = spec100_10, data = lnKFH)
```

führt zu folgendem – verkürzten – Output:

```
> lnKFH.arch1AR1

*---------------------------------*
*          GARCH Model Fit        *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model    : sGARCH(1,0)
```

[8] S. hierzu Curtis Miller (2019), Problems in Estimating GARCH Parameters in R (Part 2; rugarch), abgerufen 23.3.2020 von https://ntguardian.wordpress.com/2019/01/28/problems-estimating-garch-parameters-r-part-2-rugarch/.

[9] Das Argument arfima ist hierbei per default auf FALSE eingestellt. Der Buchstabe „f“ in dem Argument arfima steht für „fractional“ und bedeutet, dass bei einer Setzung auf TRUE die Modellanpassung durch Bildung fraktionaler statt ganzzahliger Differenzen d optimiert wird. ARFIMA(p, d, q)-Modelle können mit dem Paket rugarch auch völlig eigenständig als Verallgemeinerungen der ARIMA(p, d, q)-Modelle geschätzt werden.

```
Mean Model     : ARFIMA(1,0,0)
Distribution   : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value  Pr(>|t|)
mu      5.328665  0.018455  288.7411     0
ar1     0.997893  0.001425  700.3788     0
omega   0.000209  0.000008   25.5301     0
alpha1  0.163511  0.031749    5.1501     0

LogLikelihood : 5883.107

Information Criteria

Akaike       -5.4894
Bayes        -5.4788
Shibata      -5.4894
Hannan-Quinn -5.4855
```

Hier steht die Tabelle der Optimal Parameters im Blickpunkt des Interesses. Die Spalte der p-Werte zeigt, dass alle Koeffizienten hoch signifikant geschätzt worden sind. Die ersten beiden mit mu und ar1 angegebenen Koeffizienten beziehen sich auf das AR(F)IMA(1, 0, 0)-Modell (Mean Model) und die letzten beiden mit omega und alpha1 bezeichneten Koeffizienten auf das s(G)ARCH(1, 0)-Modell (GARCH Model).[10] mu und omega geben das jeweilige absolute Glied in der Mittelwert- bzw. Varianzgleichung an. ar1 ist der geschätzte autoregressive Parameter der Mittelwertgleichung, der dem Regressionskoeffizienten des um eine Periode verzögerten logarithmierten Aktienkurses entspricht. alpha1 ist der geschätzte ARCH(1)-Parameter, d.h. der geschätzte Koeffizient der um eine Periode verzögerten Residuenquadrate. Der Parameterschätzung von 0.163511 zufolge Schätzung gehen die quadrierten Residuen der Vorperiode zu approximativ 1/6 in die bedingte Varianz der Störgröße ein.

Mit einem Wert von 0.997893 liegt der geschätzte autoregressive Parameter des Mean Models nahe bei 1. Tatsächlich weicht der AR(1)-Koeffizient nicht signifikant von 1 ab, was die Modellierung des logarithmierten Kurses der KFH-Aktien durch einen Random Walk nahe legt. In diesem Fall gehorchen die Renditen (logarithmierte Differenzen) einem White-Noise-Prozess (weißes Rauschen). Bei Beibehaltung eines ARCH(1)-Prozesses lautet dann die Modellspezifikation:

```
> spec000_10 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 0)), mean.model
= list(armaOrder = c(0, 0), arfima = FALSE))
```

Hiermit ergeben sich die folgenden Schätz- und Testergebnisse:

[10] Das Präfix „s“ bei dem (G)ARCH(1, 0)-Model steht für „standard“ und weist damit darauf hin, dass hier die Standversion der Modellvaianten verwendet worden ist. „G“ steht für „generalized“ und gibt eine Verallgemeinerung der ARCH-Modelle an, auf die wir im Folgenden noch eingehen werden.

```
> QDlnKFH.arch1RW = ugarchfit(spec = spec000_10, data = QDlnKFH)
> QDlnKFH.arch1RW

*---------------------------------*
*        GARCH Model Fit          *
*---------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,0)
Mean Model      : ARFIMA(0,0,0)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value  Pr(>|t|)
mu      0.000198  0.000007   29.651    0
omega   0.000000  0.000000   26.318    0
alpha1  0.319302  0.016025   19.925    0

LogLikelihood : 13456.59

Information Criteria

Akaike       -12.568
Bayes        -12.560
Shibata      -12.568
Hannan-Quinn -12.565
```

Wird statt des AR(1)-Prozesses ein Random-Walk-Prozess der logarithmierten Aktienkurse angenommen, ist der geschätzte ARCH(1)-Koeffizient gleichermaßen hoch signifikant. Jedoch gehen bei einem alpha1 von 0.319302 die quadrierten Residuen der Vorperiode jetzt mit einem Anteil von knapp 1/3 in die bedingte Varianz der Störgröße ein.

Ob ARCH-Effekte höherer Ordnung vorliegen, kann mit einem Lagrange-Multiplier-Test (LM-Test) überprüft werden. In diesem Fall würde die bedingte Störvarianz auch durch zeitlich weiter zurückliegende Residuenquadrate beeinflusst werden. Ein ARCH-LM-Test, der sich direkt auf eine Zeitreihe anwenden lässt, ist in dem R-Paket FinTS verfügbar. Beispielhaft über prüfen wir hier die Renditen der KFH-Aktie nach einer Mittelwertbereinigung (= Residuen eines White-Noise-Prozesses) auf ARCH(1)- und ARCH(3)-Effekte.

```
> library(FinTS)
```

```
> ArchTest(QDlnKFH, lags=1, demean=TRUE)

      ARCH LM-test; Null hypothesis: no ARCH effects

data:  QDlnKFH
Chi-squared = 48.707, df = 1, p-value = 2.972e-12

> ArchTest(QDlnKFH, lags=3, demean=TRUE)

      ARCH LM-test; Null hypothesis: no ARCH effects

data:  QDlnKFH
Chi-squared = 49.8, df = 3, p-value = 8.811e-11
```

Da signifikante ARCH-Effekte nicht nur für einen Lag von 1, sondern auch für einen Lag von 3 auftreten, lässt sich eine ökonometrische Schätzung des AR(F)IMA(0,0,0)-ARCH(3)-Modell der Aktienrenditen begründen:

```
> spec000_30 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(3, 0)), mean.model
= list(armaOrder = c(0, 0), arfima = FALSE))
> QDlnKFH.arch5RW = ugarchfit(spec = spec000_30, data = QDlnKFH)
> QDlnKFH.arch5RW

*---------------------------------------*
*          GARCH Model Fit          *
*---------------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(3,0)
Mean Model      : ARFIMA(0,0,0)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error  t value  Pr(>|t|)
mu      0.000157  0.000004   39.6543  0.00000
omega   0.000000  0.000000    5.0418  0.00000
alpha1  0.276631  0.034137    8.1036  0.00000
alpha2  0.007141  0.006757    1.0569  0.29056
alpha3  0.714931  0.054450   13.1301  0.00000

LogLikelihood : 13400.95
```

```
Information Criteria
------------------------------------

Akaike       -12.514
Bayes        -12.500
Shibata      -12.514
Hannan-Quinn -12.509
```

Bei einer leichten Verminderung von 0.319302 auf 0.276631 gegenüber dem ARCH(1)-Modell bleibt der Koeffizient alpha1 im ARCH(3)-Modell hoch signifikant. Dagegen haben die um zwei Perioden verzogerten Residuenquadrate keinen signifikanten Effekt auf die bedingte Störvarianz. Jedoch weist der geschätzte Koeffiziente alpha3 von 0.714931 einen starken und hoch signifikanten Einfluss der um drei Perioden verzögerten Residuenquadrate auf die bedingte Varianz der Rendite der KFH-Aktie aus.

Übungsaufgaben

5.2.1 Spezifizieren Sie aufgrund der in Aufg. 5.1.3 gewonnenen Informationen die Argumente mean.model und variance.model der R-Funktion ugarchspec und passen Sie das resultierende AR(F)IMA-ARCH-Modell an die Zeitreihe DlnTHY.zoo an! Interpretieren Sie die geschätzten Koeffizienten (Optimal Parameters)!

5.2.2 Testen Sie die Residuen des in Aufg. 5.2.1 geschätzten mean.model auf ARCH-Effekte bis zur sechsten Ordnung und interpretieren Sie die Testergebnisse!

5.2.3 Schätzen Sie ein ARCH(q)-Modell für die Renditen der THY-Aktie unter Berücksichtigung der in Aufg. 5.2.2 ermittelten Testergebnisse und interpretieren Sie die geschätzten Koeffizienten (Optimal Parameters)!

5.2 GARCH-Modelle

Ähnlich wie ein AR-Modell zu einem ARMA-Modell verallgemeinert werden kann, lässt sich ein ARCH-Modell zu einem GARCH-Modell (Generalized Autoregressive Conditional Heteroskedastic Model) erweitern. Bei einem GARCH-Modell ist die bedingte Varianz nicht mehr allein von den zeitlich verzögerten quadrierten Störgrößen abhängig, sondern ebenfalls von den verzögerten bedingten Varianzen.

Wir richten hier unseren Fokus auf das wohl am häufegsten eingesetzte GARCH(1,1)-Modell:

(5.4) $\mathrm{Var}(U_t \mid I_{t-1}) = \sigma_t^2 = \alpha_0 + \alpha_1 \cdot u_{t-1}^2 + \gamma_1 \cdot \sigma_{t-1}^2,$

bei dem der Koeffizient α_1 den Effekt der um eine Periode verzögerten quadrierten Störgröße und der Koeffizient γ_1 den Einfluss der um eine Periode verzögerten bedingten Varianz auf die aktuelle Varianz messen.

Das Prinzip der Sparsamkeit stand wie beim ARMA-Modell auch beim GARCH-Modell Pate. So hat Bollerslev (1986) gezeigt, dass ein GARCH (p,q)-Modell einem ARCH(∞)-Modell entspricht, sofern Stationarität gegeben ist. Für die Anwendung bedeutet dies, dass ein ARCH-Prozess hoher Ordnung oft durch ein GARCH-Modell niedrigerer Prozessordnungen p und q approximiert werden kann.

Mit dieser Ausrichtung setzen wir hier das GARCH(1, 1)-Modell zum Zweck der Einschätzung des zeitlich bedingten Risikos in der Kursentwicklung bzw. Rendite der KFH-Aktie ein. Unter Verwendung der Spezifikation

```
> spec000_11 = ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(1, 1)), mean.model
= list(armaOrder = c(0, 0), arfima = FALSE))
```

erhält man nach der ökonometrischen Schätzung mit der Funktion ugarchfit des Pakets rugarch den – hier verkürzten – Output mit den geschätzten Koeffizienten des ARFIMA(0, 0, 0)-GARCH(1, 1) Modells:

```
> QDlnKFH.garch11RW = ugarchfit(spec = spec000_11, data = QDlnKFH)
> QDlnKFH.garch11RW

*---------------------------------------*
*          GARCH Model Fit              *
*---------------------------------------*

Conditional Variance Dynamics
-----------------------------------
GARCH Model     : sGARCH(1,1)
Mean Model      : ARFIMA(0,0,0)
Distribution    : norm

Optimal Parameters
------------------------------------
        Estimate  Std. Error   t value   Pr(>|t|)
mu      0.000380  0.000005   73.9705   0.00000
omega   0.000000  0.000000    0.0001   0.99992
alpha1  0.056022  0.001112   50.3796   0.00000
beta1   0.902640  0.003999  225.7081   0.00000

LogLikelihood : 12849.07

Information Criteria
------------------------------------

Akaike       -11.999
Bayes        -11.989
Shibata      -11.999
Hannan-Quinn -11.995
```

Die Modellschätzung kann zur Prognose der Aktienkurse bzw. Renditen und ihrer Volatilität genutzt werden. Hierzu ist die Funktion ugarchforecast einsetzbar, in der wir einen Prognosehorizont von 10 Perioden (Börsentagen) spezifizieren:

```
> QDlnKFH.forecast = ugarchforecast(QDlnKFH.garch11RW, n.ahead=10)
> QDlnKFH.forecast

*------------------------------------*
*     GARCH Model Forecast           *
*------------------------------------*
Model: sGARCH
Horizon: 10
Roll Steps: 0
Out of Sample: 0

0-roll forecast [T0=1975-11-12 01:00:00]:
        Series     Sigma
T+1   0.00038 0.0003807
T+2   0.00038 0.0003728
T+3   0.00038 0.0003650
T+4   0.00038 0.0003574
T+5   0.00038 0.0003500
T+6   0.00038 0.0003427
T+7   0.00038 0.0003355
T+8   0.00038 0.0003286
T+9   0.00038 0.0003217
T+10  0.00038 0.0003150
```

Während die prognostizierte Rendite in dem Element SeriesFor (im Output Series) des Slots forecast gespeichert ist, enthält das Element SigmaFor (im Output Sigma) die Quadratwurzeln der bedingten Varianz. Die Konstanz der prognostizierten Renditen resultiert aus der Zugrundelegung eines White-Noise-Prozesses der logarithmierten Kursdifferenzen (= Random Walk der logarithmierten Zeitreihe) als Mean Model. Die veränderlichen Werte der bedingten Standardabweichung ergeben sich dagegen aus den Punktschätzern der Parameter des GARCH(1, 1)-Modells.

Um die Prognose der Volatilität der KFH-Aktie in Form der bedingten Varianz grafisch darzustellen, extrahieren wir die Variable sigmaFor aus dem Slot forecast:

```
> QDlnKFH.sigmaf = QDlnKFH.varf@forecast$sigmaFor
> class(QDlnKFH.sigmaf)
[1] "matrix"
> dim(QDlnKFH.sigmaf)
[1] 10  1
```

```
> QDlnKFH.sigmaf
   1975-11-12 01:00:00
T+1      0.0003807337
T+2      0.0003727990
T+3      0.0003650304
T+4      0.0003574244
T+5      0.0003499777
T+6      0.0003426870
T+7      0.0003355489
T+8      0.0003285604
T+9      0.0003217183
T+10     0.0003150195
```

Wie sich zeigt, ist das Element sigmaFor ein Objekt des Typs matrix. Nach vorheriger Umwandlung in einen Vektor (Typ numeric) definieren wir es als zoo-Objekt:

```
> QDlnKFH.sigmaf = as.numeric(QDlnKFH.sigmaf)
> class(QDlnKFH.sigmaf)
[1] "numeric"
> length(QDlnKFH.sigmaf)
[1] 10
> QDlnKFH.sigmaf
 [1] 0.0003807337 0.0003727990 0.0003650304 0.0003574244 0.0003499777
 [6] 0.0003426870 0.0003355489 0.0003285604 0.0003217183 0.0003150195

> timef = as.Date(c("1996-03-20", "1996-03-21", "1996-03-22", "1996-03-25", "1996-03-26", "1996-03-
27", "1996-03-28","1996-04-01", "1996-04-02", "1996-04-03"))
> class(timef)
[1] "Date"
> length(timef)
[1] 10
> timef
 [1] "1996-03-20" "1996-03-21" "1996-03-22" "1996-03-25" "1996-03-26"
 [6] "1996-03-27" "1996-03-28" "1996-04-01" "1996-04-02" "1996-04-03"

> QDlnKFH.sigmaf.zoo = zoo(QDlnKFH.sigmaf, order.by = timef)
> class(QDlnKFH.sigmaf.zoo)
[1] "zoo"
> length(QDlnKFH.sigmaf.zoo)
[1] 10
> QDlnKFH.sigmaf.zoo
  1996-03-20   1996-03-21   1996-03-22   1996-03-25   1996-03-26   1996-03-27
0.0003807337 0.0003727990 0.0003650304 0.0003574244 0.0003499777 0.0003426870
  1996-03-28   1996-04-01   1996-04-02   1996-04-03
0.0003355489 0.0003285604 0.0003217183 0.0003150195
```

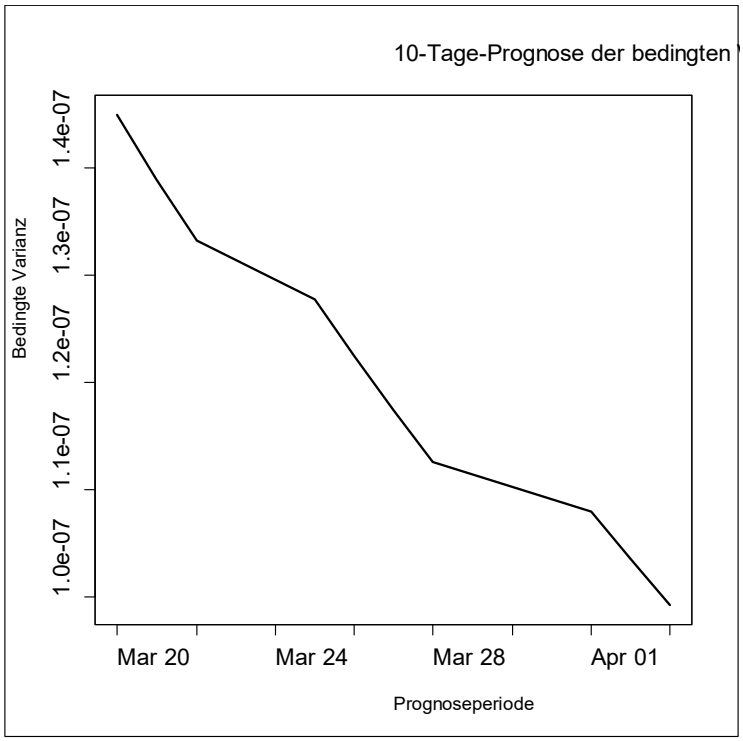
Durch Quadrierung erhalten wir aus den bedingten Standardabweichungen die Prognosewerte der bedingten Varianz des GARCH(1, 1)-Modells,

```
> QDlnKFH.varf.zoo = QDlnKFH.sigmaf.zoo^2
```

die mit der plot-Funktion grafisch ilustriert werden können:

```
> plot(QDlnKFH.varf.zoo, main="10-Tage-Prognose der bedingten Varianz der KFH-Aktie", type="l",
xlab="Prognoseperiode", ylab="Bedingte Varianz", lwd=2, cex.main = 1.25, cex.axis = 1.25,
font.main=1)
> box(which="figure")
```

Abbildung 5.1: 10-Tage-Prognose der bedingten Varianz der KFH-Aktie



Aus Abb. 5.1 geht eine abnehmende Volatilität der KFH-Aktie im Prognosezeitraum hervor.

Abschließend soll die prognostizierte Entwicklung der Volatilität in einen Kontext mit ihrem Verlauf in der jüngsten Vergangenheit gebracht werden. Sie wird durch einen 50-Tage-Zeitraum vor der Prognoseperiode definiert, in der die quadrierten Residuen den bedingten Varianzen gegenübergestellt werden. Sie vermitteln gleichzeitig einen visuellen Eindruck über die Güte der Anpassung des geschätzten GARCH-Modells in der Subperiode.

Um die drei Zeitreihen für den gesamten 60-Tage-Zeitraum zu definieren und gegenüberstellen zu können, sind vorab NA-Subzeitreihen (NA: not available) und Zeitindizes zu bilden.

```
> NAf.zoo = zoo(rep(NA, 10), order.by = timef)
> class(NAf.zoo)
[1] "zoo"
> length(NAf.zoo)
[1] 10

> NAf.zoo
1996-03-20 1996-03-21 1996-03-22 1996-03-25 1996-03-26 1996-03-27 1996-03-28
        NA         NA         NA         NA         NA         NA         NA
1996-04-01 1996-04-02 1996-04-03
        NA         NA         NA

> time.50Date = tail(as.Date(as.character(time(KFH.zoo))), 50)
> class(time.50Date)
[1] "Date"
> length(time.50Date)
[1] 50
> tail(time.50Date)
[1] "1996-03-12" "1996-03-13" "1996-03-14" "1996-03-15" "1996-03-18"
[6] "1996-03-19"

> NA50.zoo = zoo(rep(NA, 50), order.by = time.50Date)
> class(NA50.zoo)
[1] "zoo"
> length(NA50.zoo)
[1] 50
> head(NA50.zoo)
1996-01-10 1996-01-11 1996-01-12 1996-01-15 1996-01-16 1996-01-17
        NA         NA         NA         NA         NA         NA
> tail(NA50.zoo)
1996-03-12 1996-03-13 1996-03-14 1996-03-15 1996-03-18 1996-03-19
        NA         NA         NA         NA         NA         NA
```

Hiermit erhält man die Zeitreihen QDlnKFH.60res2.zoo, QDlnKFH.60var.zoo und QDlnKFH.60varf.zoo für den zu betrachtenden 60-Tage-Zeitraum:

```
> QDlnKFH.res2 = QDlnKFH.garch11RW@fit$residuals)^2
> class(QDlnKFH.res2)
[1] "numeric"
> length(QDlnKFH.res2)
[1] 2141
> QDlnKFH.50res2 = tail(QDlnKFH.res2, 50)
> class(QDlnKFH.50res2)
[1] "numeric"
> length(QDlnKFH.50res2)
[1] 50
> QDlnKFH.50res2.zoo = zoo(QDlnKFH.50res2, order.by = time.50Date)
```

```
> class(QDlnKFH.50res2.zoo)
[1] "zoo"
> length(QDlnKFH.50res2.zoo)
[1] 50
> tail(QDlnKFH.50res2.zoo)
  1996-03-12   1996-03-13   1996-03-14   1996-03-15   1996-03-18   1996-03-19
6.714491e-09 1.100636e-07 3.829426e-07 9.065606e-08 1.090430e-07 1.243466e-07
> QDlnKFH.60res2.zoo = rbind(QDlnKFH.50res2.zoo, NAf.zoo)
> class(QDlnKFH.60res2.zoo)
[1] "zoo"
> length(QDlnKFH.60res2.zoo)
[1] 60
> tail(QDlnKFH.60res2.zoo, 15)
  1996-03-13   1996-03-14   1996-03-15   1996-03-18   1996-03-19   1996-03-20
1.100636e-07 3.829426e-07 9.065606e-08 1.090430e-07 1.243466e-07           NA
  1996-03-21   1996-03-22   1996-03-25   1996-03-26   1996-03-27   1996-03-28
          NA           NA           NA           NA           NA           NA
  1996-04-01   1996-04-02   1996-04-03
          NA           NA           NA

> QDlnKFH.var =QDlnKFH.garch11RW@fit$var
> class(QDlnKFH.var)
[1] "numeric"
> length(QDlnKFH.var)
[1] 2141
> QDlnKFH.50var = tail(QDlnKFH.var, 50)
> class(QDlnKFH.50var)
[1] "numeric"
> length(QDlnKFH.50var)
[1] 50
> QDlnKFH.50var.zoo = zoo(QDlnKFH.50var, order.by = time.50Date)
> class(QDlnKFH.50var.zoo)
[1] "zoo"
> length(QDlnKFH.50var.zoo)
[1] 50
> tail(QDlnKFH.50var.zoo)
  1996-03-12   1996-03-13   1996-03-14   1996-03-15   1996-03-18   1996-03-19
2.000168e-07 1.809325e-07 1.694961e-07 1.744604e-07 1.625668e-07 1.528614e-07
> QDlnKFH.60var.zoo = rbind(QDlnKFH.50var.zoo, NAf.zoo)
> class(QDlnKFH.60var.zoo)
[1] "zoo"
> length(QDlnKFH.60var.zoo)
[1] 60
```

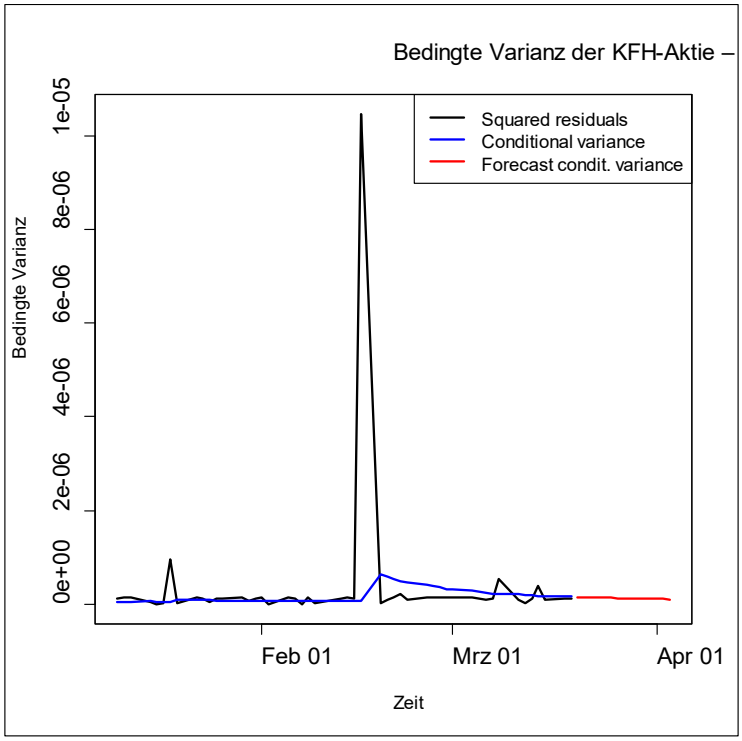
```
> tail(QDlnKFH.60var.zoo, 15)
  1996-03-13   1996-03-14   1996-03-15   1996-03-18   1996-03-19   1996-03-20
1.809325e-07 1.694961e-07 1.744604e-07 1.625668e-07 1.528614e-07           NA
  1996-03-21   1996-03-22   1996-03-25   1996-03-26   1996-03-27   1996-03-28
          NA           NA           NA           NA           NA           NA
  1996-04-01   1996-04-02   1996-04-03
          NA           NA           NA

> QDlnKFH.60varf.zoo =  rbind(NA50.zoo, QDlnKFH.varf.zoo)
> class(QDlnKFH.60varf.zoo)
[1] "zoo"
> length(QDlnKFH.60varf.zoo)
[1] 60
> tail(QDlnKFH.60varf.zoo, 15)
  1996-03-13   1996-03-14   1996-03-15   1996-03-18   1996-03-19   1996-03-20
          NA           NA           NA           NA           NA 1.449581e-07
  1996-03-21   1996-03-22   1996-03-25   1996-03-26   1996-03-27   1996-03-28
1.389791e-07 1.332472e-07 1.277522e-07 1.224844e-07 1.174344e-07 1.125931e-07
  1996-04-01   1996-04-02   1996-04-03
1.079519e-07 1.035026e-07 9.923727e-08
```

Unter Verwendung der Funktionen plot und lines lässt sich das Diagramm multipler Zeitreihen erstellen:

```
> plot(QDlnKFH.60res2.zoo, main="Bedingte Varianz der KFH-Aktie – Anpassung und Prognose",
type="l", xlab="Zeit", ylab="Bedingte Varianz", lwd=2, cex.main = 1.25, cex.axis = 1.25,
font.main=1)
> lines(QDlnKFH.60var.zoo, col="blue", lwd=2)
> lines(QDlnKFH.60varf.zoo, col="red", lwd=2)
> legend("topright", legend=c("Squared residuals", "Conditional variance", "Forecast condit. variance"),
col=c("black", "blue", "red"), lwd=2, cex=1.0)
> box(which="figure")
```

In Abb. 5.2 lassen sich die dargestellten Zeitreihen durch die in der Legende angegebenen Farben identifizieren:

Abbildung 5.2: Bedingte Varianz der KFH-Aktie – Anpassung und Prognose



An den meisten Tagen des abgebildeten Beobachtungszeitraums ist eine gute Anpassung der Volatilität durch das geschätzte GARCh(1,1)-Modell gegeben. Man erkennt jedoch auch das typische GARCH-Muster, dass hierbei auftretende hohe Volatilitäten nicht unmittelbar, sondern erst in der nächsten Periode und in erheblich abgeschwächter Form eine Adjustierung der bedingten Varianz bewirken. Sie klingt in den Folgeperioden wieder allmählich ab. Der Kontext vermittelt auch, dass die bedingte Varianz in der Prognoseperiode nur geringfügig fallend verläuft.

Übungsaufgaben

5.3.1 Schätzen Sie ein GARCH(1,1)-Modell für die Renditen der THY-Aktie unter Verwendung der in Aufg. 5.2.1 spezifizierten Mittelwertgleichung! Interpretieren Sie die geschätzten Koeffizienten (Optimal Parameters)!

5.3.2 Stellen Sie die quadrierten Residuen und die bedingten Varianzen des in Aufg. 5.3.1 angepassten GARCH-Modells für die letzten 60 Tage des Beobachtungszeitraums in einem Zeitreihendiagramm dar und interpretieren Sie es!

5.3.2 Erstellen Sie eine 12-Tage-Prognose der bedingten Varianz der THY-Aktie und erläutern Sie ihren zeitlichen Verlauf!