Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
IDP Institut für Datenanalyse
und Prozessdesign

Zeitreihen und Prognosen

Modul B - Multiple Regression und Prognose Teil 2

13. April 2020

Christoph Hofer

Originaltext aus dem DP Skript : Zeitreihenanalyse von Dr. Marcel Dettling für den Studiengang Verkehrssysteme an der ZHAW

Email: christoph.hofer@zhaw.ch

Inhaltsverzeichnis

1 Einführung

1.1 Sinn & Zweck

Zeitreihen sind allgegenwärtig! Und sie entstehen ganz natürlich: vielerorts werden nämlich Daten *in einem* bzw. *über einen* fixen Messrhythmus erhoben, z.B. die Höchsttemperatur eines jeden Tages, oder die totale Anzahl Passagiere des jeweiligen Monats. Zeitreihen treten im Verkehrswesen prominent auf, in vielen anderen Gebieten ebenso, zum Beispiel:

Business Verkaufszahlen, Produktionsdaten, Kundenfrequenzen, ...

Wirtschaft Aktienkurse, Wechselkurse, Zinsen, ...

Offizielle Statistik Volkszählungen, Konsumausgaben, Unfallstatistiken, ...

Naturwissenschaften Populationsgrössen, Sonnenflecken-Aktivität, Chemische Prozessdaten, ...

Umweltwissenschaften Niederschlags-, Temperatur- oder Luftschadstoff-Messungen, ...

Bei der statistischen Auswertung solcher Daten spricht man von *Zeitreihenanalyse*. Deren Hauptziele sind das Verstehen des vergangenen Verlaufs, und die Vorhersage von zukünftigen Werten. Während einfache (grafische) Methoden aus der beschreibenden Statistik das Verständnis der Daten oft schon entscheidend verbessern können, liefert meist erst eine modellbasierte Analyse den vollständigen Einblick in die Zusammenhänge.

In den (zahlreichen) Fällen, wo ein gutes Modell für die Daten gefunden werden kann, ist es möglich, Vorhersagen für zukünftige Werte zusammen mit einem Prognoseintervall zu erzeugen. Diese dienen z.B. als Grundlage, um Budgets festzulegen, über Beschaffungen zu entscheiden, etc. Die Zeitreihen-Modelle können auch benutzt werden, um Schwankungen in den Daten zu beurteilen: z.B. ob eine beobachtete Abweichung noch im Rahmen der zufälligen Variabilität liegt, oder ob sich möglicherweise substanzielle Änderungen ergeben haben.

Bei vielen Zeitreihen aus der Praxis beobachtet man Trends und/oder saisonale Effekte. Diese werden meist als deterministisch angenommen, d.h. man geht davon aus, dass sie mit mathematischen Funktionen vollständig beschrieben werden können. Doch auch das Element des Zufalls hat bei Zeitreihen seinen Platz. Es gibt in der Regel eine sogenannt stationäre Komponente, wo die Beobachtungen als zufällig, aber seriell korreliert angenommen werden. D.h. dass bis zu einem gewissen Ausmass die nächste Beobachtung durch die vorangehenden vorhersehbar ist. Viele der Zeitreihen-Modelle versuchen diese Abhängigkeiten auszunutzen.

Dieses Zeitreihen-Skript fokussiert auf die Praxis: von den mathematischen Grundlagen wird nur das nötigste Besprochen, die Betonung liegt auf dem Durchführen von Zeitreihenanalysen mit der Software **R**. Die Hauptelemente sind das Beschreiben und Visualisieren von Zeitreihen, das Anpassen von einfachen Modellen, das Erzeugen von Vorhersagen und das Ziehen von adäquaten Schlussfolgerungen aus den Outputs.

1.2 Beispiele

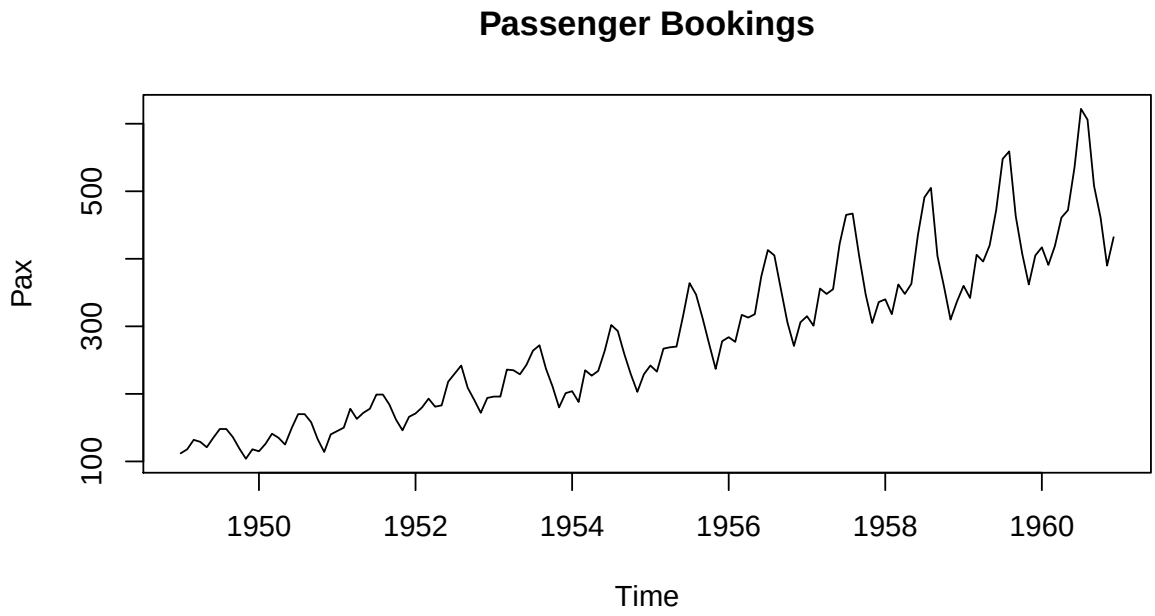
1.2.1 Air Passenger Bookings

Von der *United States Federal Aviation Administration* ist die Anzahl Buchungen pro Monat (in 1'000) für die Fluggesellschaft *PanAm* während der Periode 1949 bis 1960 erhältlich. Die Airline nutzte diese Daten, um zukünftige Passagiervolumina vorherzusagen, die als Basis für Entscheidungen über die Beschaffung von neuen Flugzeugen und das Rekrutieren bzw. Ausbilden von Personal dienten. Die *Air Passenger Bookings* sind als Beispiel-Datensatz in R vorhanden. Als erstes zeigen wir, wie die Daten geladen werden, und wie man sie in der Konsole anzeigt.

```
> data(AirPassengers)
> AirPassengers

     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

Weitere Informationen zu den Air Passenger Bookings erhält man direkt in **R** mit dem Befehl `?AirPassengers`. Der Datensatz liegt als Objekt der Klasse`ts` vor: diejenige Struktur, welche für den Umgang mit Zeitreihen gedacht ist. Weiteres Wissenswertes zum Umgang mit Zeitreihen in **R** finden sich im Abschnitt 3. Der grundlegende Schritt in der Zeitreihenanalyse ist stets das Visualisieren der Daten, d.h. die Darstellung in einem *Zeitreihenplot*. Hierbei werden die Zahlenwerte auf der y-Achse gegen die Zeit der Beobachtung auf der x-Achse aufgetragen. Falls die Daten in **R** als Zeitreihenobjekt vorliegen, so ist dies sehr einfach, bzw. geschieht beim Aufruf der generischen `plot()`-Funktion automatisch. Befehl und Output sind wie folgt:

```
> plot(AirPassengers, ylab='Pax', main='Passenger Bookings')
```



Der Plot zeigt einige Eigenheiten, welchen vielen Zeitreihen gemein sind. So nimmt die Anzahl Passagiere über die Zeit deutlich zu. Eine solche, systematische Änderung des Erwartungswerts nennt man einen *Trend*. Das einfachste Modell für einen Trend ist eine lineare Funktion, d.h. das Anpassen einer (Regressions-)Gerade. Häufig ist dies in der Praxis bereits hinreichend genau, doch es gibt auch verfeinerte Methoden. Details dazu finden sich in Kapitel 6

Neben dem Trend zeigen die Air Passenger Bookings auch ein saisonales Muster, d.h. wir beobachten im Sommer stets höhere Passagierzahlen als im Winter. Ein zyklisches Verhalten ist typisch für Zeitreihen mit Monatsdaten. Der Begriff der Saisonalität wird jedoch auch bei anderem Messrhythmus verwendet, z.B. für Anzahl Reservationen pro Tag in einem Restaurant, die oftmals einen Wochenrhythmus aufweisen.

Sowohl für den Trend wie auch für die Saisonalität gibt es bei den Air Passenger Bookings klar nachvollziehbare, exogene Faktoren. Die generelle Zunahme der Buchungen ist auf Faktoren wie den wirtschaftlichen Aufschwung in der Zeit nach dem 2. Weltkrieg zurückzuführen, ebenso auf das Bevölkerungswachstum und die durch technischen Fortschritte erzielte Verbilligung von Luftreisen. Der Saisoneffekt deckt sich mit den üblichen Ferienperioden, d.h. die Sommerferien bilden die Hauptreisezeit, einen kleineren Nebengipfel beobachten wir im Frühling, während um den Jahreswechsel vergleichsweise seltener mit dem Flugzeug verreist wird.

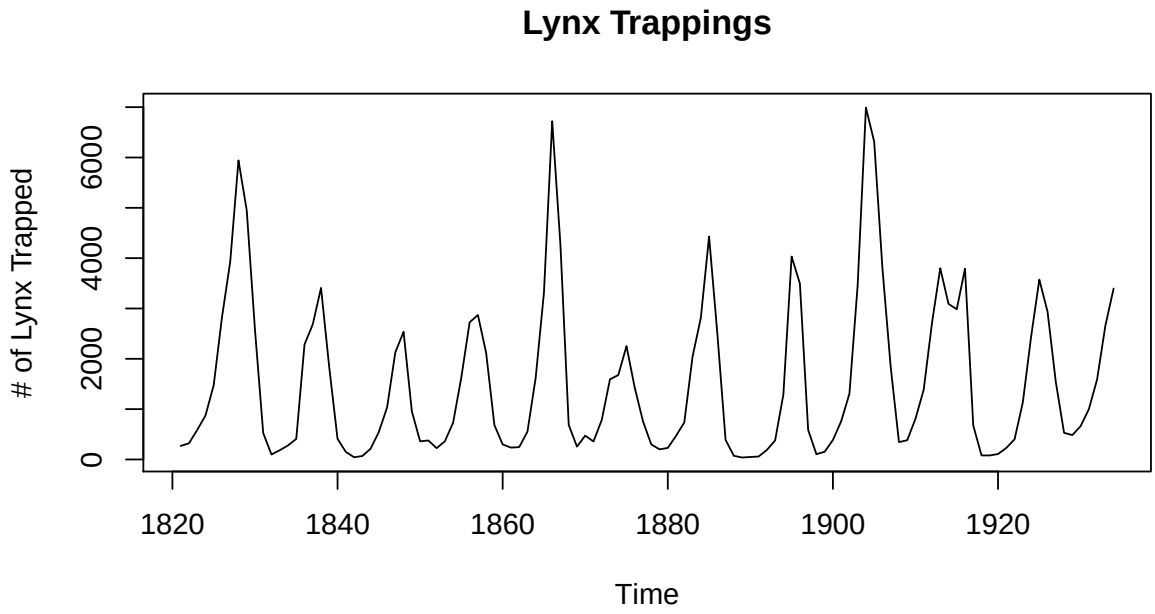
1.2.2 Lynx Trappings

Eine etwas exotisch anmutende Zeitreihe mit jedoch instruktiven Eigenschaften ist die Anzahl gefangener Luchse im Mackenzie River District in Kanada, im Zeitraum von 1824 bis 1934. Auch sie ist in **R** als Beispiel integriert. Wir laden die Reihe und stellen sie

grafisch dar:

```
data(lynx)
plot(lynx, ylab='# of Lynx Trapped', main='Lynx Trappings')
```

Wie der Plot zeigt, erreicht die Anzahl gefangener Luchse alle rund 10 Jahre ein Maximum, um danach wieder beinahe auf null abzufallen. Ebenso beobachten wir, dass die Spitze in einem Intervall von zirka 40 Jahren nochmals beinahe doppelt so hoch ist wie die anderen Maxima. Diese Phänomene können nicht wie zuvor bei den Flugpassagieren auf den (dort monatlichen) Messrhythmus zurückgeführt werden. Die Ursachen bleiben vorerst im Dunkeln, und wir benennen die Zyklen in den Daten nicht als deterministisch, sondern als zufällig, bzw. stochastisch.



Während das Verstehen von deterministischen Komponenten wie Trend und Saisonalität durchaus wichtig ist, bildet die Analyse von stochastischen Periodizitäten, wie sie in den *Lynx Trappings* in sehr offensichtlicher Weise vorliegen, das Herzstück der Zeitreihenanalyse. Der grösste Teil von Literatur und Methodik zu Zeitreihen dreht sich um die Analyse solcher, *stationärer Reihen*.

Eine wesentliche Eigenheit der Lynx-Daten ist deren Abhängigkeit: es besteht eine Korrelation zwischen nacheinander folgenden Beobachtungen. Falls der vorangehende Wert nahe bei null war, d.h. wir uns „im Tal“ befinden, so ist der aktuelle Wert mit grosser Wahrscheinlichkeit ebenfalls wieder klein. Und natürlich auch umgekehrt, ein hoher Vorgänger-Wert impliziert für das aktuelle Jahr einen überdurchschnittlichen Wert. Das Aufzeigen, Verstehen und Beschreiben solcher Abhängigkeiten ist die Aufgabe der mathematischen Zeitreihenanalyse.

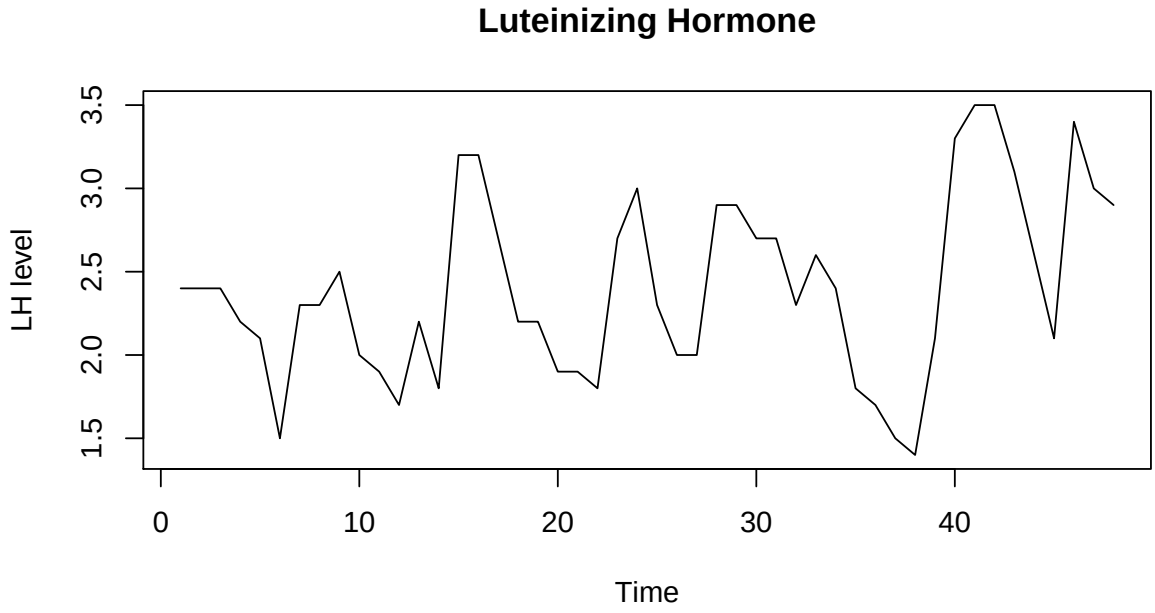
1.2.3 Luteinizing Hormone Measurements

Um die Abhängigkeiten von zeitlich aufeinander folgenden Beobachtungen intuitiv noch besser verstehen zu können, betrachten wir hier eine weitere Beispielreihe, welche den Wert des den Eisprung steuernden Geschlechtshormons Lutropin beschreibt. Bei einer Frau wurden in 10-Minuten-Intervallen Messungen vorgenommen, insgesamt 48 Stück. Die Daten sind in **R** verfügbar.

```
data(lh)
lh
```

Wie immer sind die exakten Zahlenwerte (wie oben dargestellt) unverzichtbar, doch bieten sie wenig Übersicht. Ein Zeitreihenplot schafft Abhilfe:

```
> plot(lh, ylab='LH level', main='Luteinizing Hormone')
```

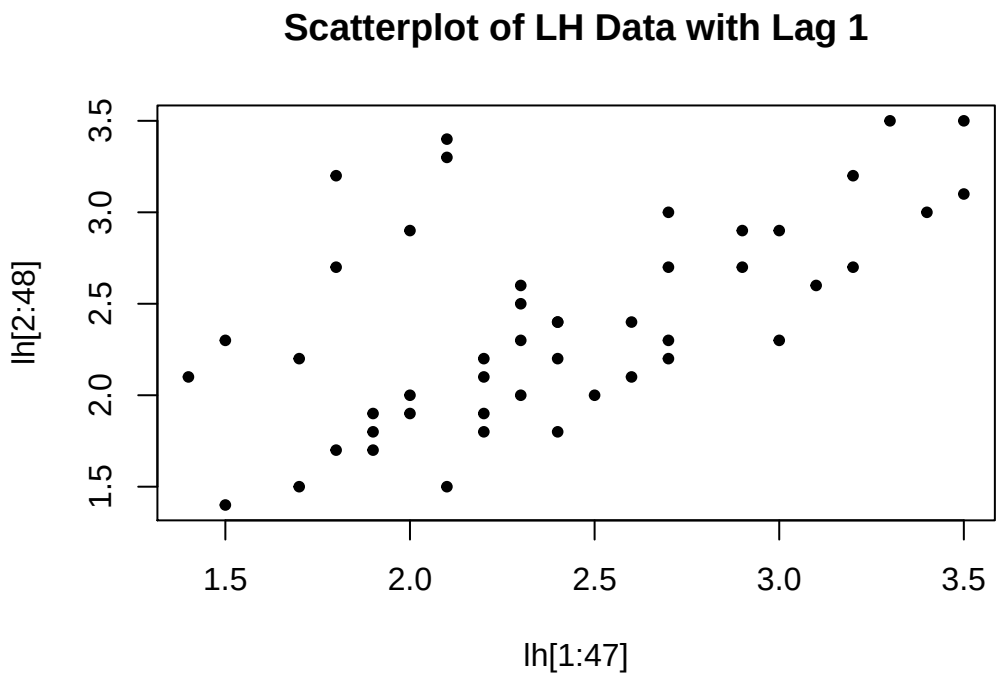


Aufgrund des Messrhythmus, d.h. den 48 Intervallen à 10 Minuten, ist ein deterministischer Saisoneffekt bei diesen Daten unplausibel. Es bleibt aber die Frage, ob es stochastische Zyklen gibt. Der Plot zeigt Berge und Täler, doch weisen diese ein regelmässiges Muster auf? Anstatt uns aufs Spekulieren oder rein visuelle Folgerungen zu verlassen, sollten wir besser die Mathematik zu Hilfe nehmen.

Regelmässige Muster sind gleichbedeutend mit Abhängigkeiten, welche sich quantitativ mit Korrelationen beschreiben lassen. Als erstes Beispiel betrachten wir zeitlich unmittelbar aufeinander folgende Beobachtungen. Es geht also um die Abhängigkeit zum *Zeitabstand* oder *Lag* 1. Weil es um Abhängigkeiten innerhalb der Reihe selbst geht, spricht man auch

von der *Autokorrelation*. Wir können diese mit einem *Lagged Scatterplot* wie folgt grafisch darstellen.

```
> plot(lh[1:47], lh[2:48],pch=20, main  ='Scatterplot of LH Data with Lag 1')
```



Die Tatsache, dass die im zweiten Quadranten gelegenen 8 Beobachtungen auch eine Inhomogenität darstellen könnten, ignorieren wir der Einfachheit halber und mangels Fachwissen nun einmal beflissentlich. Auch dann (und sonst erst recht) ist eine positive Korrelation zwischen zeitlich aufeinander folgenden Beobachtungen erkennbar. Das bedeutet konkret, dass ein Zeitpunkt mit hohem Lutropinwert eher von einer weiteren Beobachtung mit hohem Wert gefolgt wird, als von einer mit tiefem Wert. Und natürlich auch umgekehrt: ein tiefer Wert impliziert einen eher tiefen, und keinen hohen Nachfolgewert. Um dies in einer Zahl festzuhalten, können wir die Pearson-Korrelation berechnen:

```
> cor(lh[1:47], lh[2:48])
```

```
[1] 0.5807322
```

Diesen Wert kann man als Schätzung für die Autokorrelation zum Lag 1 benützen. Wie in Kapitel 4.3 dann ausführlicher erklärt wird, liefert er eine zwar nützliche Approximation, doch gibt es mathematisch geeignetere Schätzmethoden. Wir beschliessen dieses Kapitel indem wir festhalten, dass die Autokorrelationen ein Mittel sind, um die Abhängigkeiten in einer Zeitreihe, bzw. um die „Berge und Täler“ eines Zeitreihenplots quantitativ festzumachen.

1.2.4 Swiss Market Index

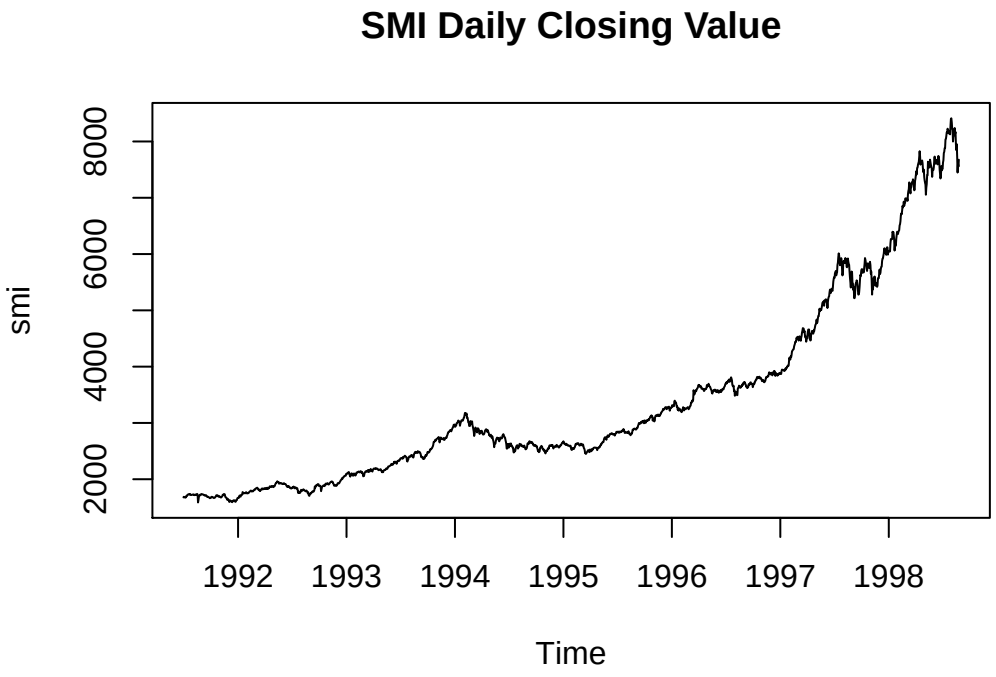
Der SMI ist der Blue Chip Index an der Schweizer Börse. Er repräsentiert die Kapitalisierung der 20 grössten Unternehmungen, die zusammen rund 85% des gesamten Marktes ausmachen. In **R** sind die täglichen Schlusskurse von 1860 aufeinander folgenden Börsen-Arbeitstagen während einer Periode von 1991 bis 1998 enthalten.

```
> data(EuStockMarkets)
> head( EuStockMarkets )
```

```
          DAX    SMI    CAC   FTSE
[1,] 1628.75 1678.1 1772.8 2443.6
[2,] 1613.63 1688.5 1750.5 2460.2
[3,] 1606.51 1678.6 1718.0 2448.2
[4,] 1621.04 1684.1 1708.1 2470.4
[5,] 1618.16 1686.6 1723.1 2484.7
[6,] 1610.61 1671.6 1714.3 2466.8
```

`EuStockMarkets` ist ein multiples Zeitreihenobjekt. Das heisst, es enthält für einen gegebenen Beobachtungszeitpunkt nicht nur einen einzigen, d.h. den SMI-Wert, sondern auch gleich noch Daten für den deutschen DAX, den französischen CAC und den FTSE der Londoner Börse. Wir wollen hier aber auf die Zeitreihe des SMI fokussieren, extrahieren sie deshalb, und stellen sie grafisch dar.

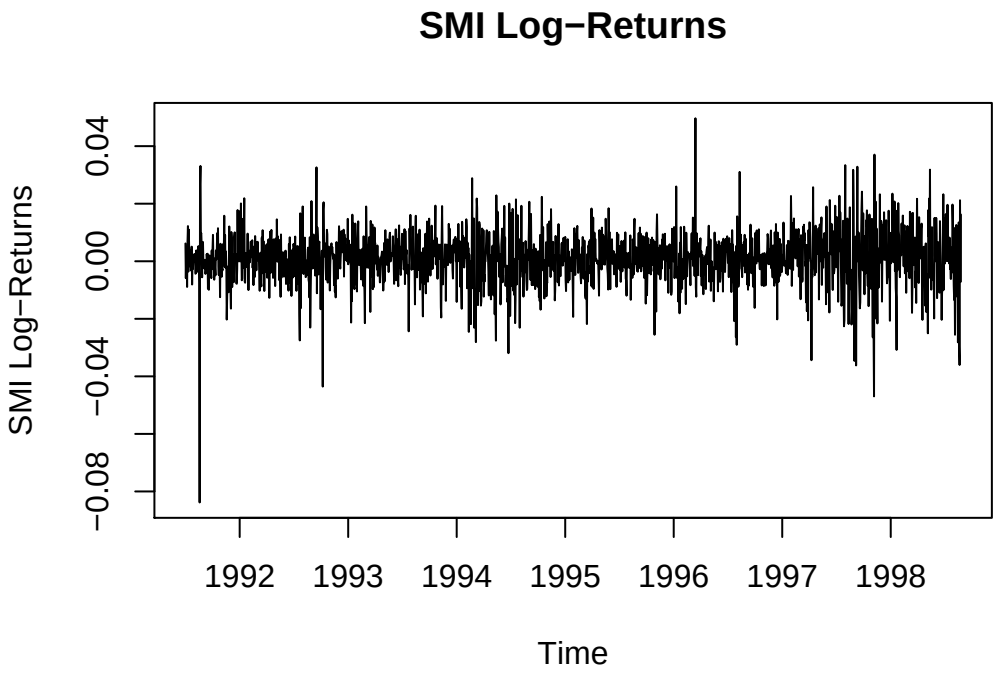
```
> esm <- EuStockMarkets
> tmp <- EuStockMarkets[,2]
> smi <- ts(tmp, start=start(esm), freq=frequency(esm))
> plot(smi, main='SMI Daily Closing Value')
```



Um den Code im Detail verstehen zu können, muss man wissen, dass sogenanntes *Subsetting von multiplen Zeitreihen-Objekten einen Vektor*, und nicht ein Zeitreihen-Objekt, *ausgibt*. Jenes muss erst wieder erzeugt werden, wobei wir bequem die Startzeitpunkt und Frequenz des Originals übernehmen können. Detaillierte Erklärungen hierzu finden sich später in Kapitel XX

Die SMI-Reihe zeigt einen deutlichen Trend, d.h. der Erwartungswert ändert im Laufe der Zeit. Dies ist typisch für Finanzzeitreihen, und ebenso ist es für diese üblich, dass der Trend deterministisch kaum modellierbar, und damit auch nicht vorhersagbar ist. Um dieses heisse Eisen kümmern wir uns hier nicht. Wir betrachten aber die sogenannten Log-Returns, d.h. wir logarithmieren die Kurse und betrachten die Differenz zwischen aufeinander folgenden Werten.

```
> tmp <- log(smi[2:1860])-log(smi[1:1859])
> lret.smi <- ts(tmp, start=c(1991,131), freq=frequency(esm))
> plot(lret.smi, main='SMI Log-Returns', ylab = "SMI Log-Returns")
```



Weil die Differenz von zwei Logarithmen dem Logarithmus des Quotienten entspricht, sind die SMI Log-Returns eine Approximation für die prozentuale Veränderung des Aktienmarktes im Vergleich zum Vortrag. Diese schwanken um Null herum und weisen keinen Trend mehr auf. Wohl scheint es stürmische Perioden zu geben, wo grosse Ausschläge häufiger vorkommen als in ruhigen Phasen. Betrachtet man hingegen Lagged Scatterplots, so findet man keine direkte Abhängigkeit in den Log-Returns. Dies bedeutet, dass man nicht vorhersagen kann, ob die Börse am nächsten Tag steigt oder fällt.

Hingegen sind instutionelle Anleger durchaus in der Lage, aus der Volatilität in der Reihe, d.h. aus den stürmischen und ruhigen Zeiten, Profit zu schlagen. Darum werden wir uns im Rahmen dieses Kurses aber nicht kümmern. Einen Hinweis auf vorhandene Abhängigkeiten findet man, wenn man die Log-Returns quadriert und dann die Autokorrelation, z.B. mit Lagged Scatterplots, studiert.

2 Mathematische Grundlagen

Ohne mathematische Grundlagen ist nicht viel mehr an Zeitreihenanalyse möglich als einige grafische Darstellungen. Auch wenn man ein praxisorientierter Anwender ist, kommt man nicht um etwas Mathematik herum. Insbesondere sollte man das Konzept der Stationarität verinnerlicht haben.

2.1 Mathematische Definition einer Zeitreihe

Eine Zeitreihe ist eine Menge von n beobachteten Datenpunkten, wir verwenden die Notation $x = (x_1, x_2, \ldots, x_n)$. Die Reihenfolge der Messungen ist relevant, und es ist wichtig, dass die Beobachtungen äquidistant sind. Eine Reihe darf z.B. Tageswerte oder monatliche Summen enthalten, jedoch nicht einen Mix von täglichen Beobachtungen und mehrtägigen Aggregaten. Ein zentrales Merkmal von Zeitreihen ist weiter, dass aufeinander folgende Messungen in der Regel stochastisch abhängig sind.

Der Stochastiker setzt hinter jede Zeitreihe einen Vektor von Zufallsvariablen, nämlich $X = (X_1, X_2, \ldots, X_n)$, und interpretiert die Beobachtungen als eine (einzige!) Realisation davon. Dieses theoretische Konstrukt nennt man Zeitreihen-Prozess. Man kann ihm wie jeder Zufallsvariablen mathematische Eigenschaften und eine Verteilung zuschreiben. Diese Dinge sind jedoch nicht direkt beobachtbar, sondern man kann höchstens aus den Daten Rückschlüsse darüber ziehen.

Dieses Vorgehen passt in die Welt der Statistik, denn sie ist die Kunst, aus vielen gleichartigen Beobachtungen das Grundlegende, Gemeinsame herauszuschälen. Nun haben wir bei Zeitreihen zwar n Beobachtungen, die aber nur eine einzige Realisation des Prozesses sind. Damit wir aus den Daten lernen können, müssen wir annehmen, dass die einzelnen Elemente der Zeitreihe auf gewisse Weise „ähnlich“ sind. Dies wird durch das Konzept der Stationarität garantiert.

2.2 Stationarität

Stationarität ist eine Eigenschaft des Zeitreihen-Prozesses und wird mit Bedingungen an die Verteilungen der Zufallsvariablen formuliert. Wir lassen uns hier nicht auf technische Details ein, sondern definieren anschaulich: Stationarität heisst, dass jeder Abschnitt der Reihe von beliebiger Länge zu jedem anderen beliebigen Zeitpunkt eingesetzt werden könnte, und passen würde! In der Praxis überprüft man diesen Sachverhalt, und zusätzlich ob:

- der Erwartungswert μ_t über die Zeit konstant ist, d.h. $\mu_t = \mu$.
- die Streuung σ_t^2 sich im Verlauf der Reihe nicht ändert, d.h. $\sigma_t^2 = \sigma^2$.
- die (Auto)korrelation nur vom Lag h abhängt, d.h. $Cor(X_t, X_{t+h}) = \gamma(h)$.

Ist mindestens eine dieser Bedingungen offensichtlich verletzt, so kann der Prozess, welcher die Reihe generiert hat, nicht stationär sein. Weil stationäre und nichtstationäre Zeitreihen

mit anderen Methoden analysiert werden, ist das Konzept auch für den Praktiker zentral. In der Datenanalyse zeigt sich auch rasch, dass viele Zeitreihen nicht stationär sind, z.B.:

- Die Air Passenger Bookings sind nicht stationär, denn sie weisen einen Trend und einen Saisoneffekt auf. Die Beobachtungen der Jahre 1950-1952 könnte man nicht zur Zeit $1958-1960$ einfügen, sie wären dort nicht passend, weil das Passagierniveau deutlich angestiegen ist. Mathematisch bedeutet dies, dass der Erwartungswert μ_t der Reihe von der Zeit t abhängt.

- Die SMI Log-Returns sind ebenfalls nicht stationär. Hier ist die Bedingung der konstanten Varianz verletzt. Es gibt Perioden wie die zweite Jahreshälfte von 1997, wo der SMI massiv stärker schwankt wie zu ruhigen Zeiten, z.B. im Jahr 1995. Die Originalreihe des SMI ist ebenfalls nicht stationär: sie weist einen Trend auf, d.h. der Erwartungswert ist über die Zeit nicht konstant.

- Bei der Reihe mit den Luteinizing Hormone Data können wir aufgrund der vorliegenden Daten keinen Widerspruch zur Stationarität erkennen. Erwartungswert und Varianz scheinen konstant, man könnte jeden Abschnitt der Reihe zu einem anderen Zeitpunkt einsetzen und würde dies nicht als „komisch“ erkennen.

- Die Lynx Trappings sind von unseren Beispielen eindeutig der schwierigste Fall. Sie zeigen, dass die Entscheidung zwischen stationär oder nichtstationär längst nicht immer trivial ist. Man kann hier durchaus Zweifel hegen, ob der Erwartungswert der Reihe konstant ist, und man jeden Abschnitt der Reihe anderswo einsetzen könnte. Dennoch zeigt die Praxis, dass man diese Reihe gut als stationär modellieren kann.

Eine Zeitreihenanalyse beginnt mit dem Sammeln und Einlesen der Daten, der nächste Schritt ist in der Regel die Darstellung mit dem Zeitreihenplot. Daraufhin gilt es zu entscheiden, ob die Reihe stationär ist oder nicht. Nach den obigen Grundlagen hier nun ein wichtiger Hinweis:

Die Stationarität einer Zeitreihe ist als eine Hypothese zu betrachten. Diese kann mit grosser Sicherheit widerlegt werden, wenn die Daten, wie z.B. die Air Passenger Bookings, deutlich Trends und/oder Saisoneffekte zeigen. Andererseits ist es nicht möglich, Stationarität zu beweisen: wenn die Reihe keine Verletzung der Annahmen zeigt, so ist es wohl plausibel, dass der zugehörige Prozess stationär ist, bewiesen ist dies aber nicht.

3 Zeitreihenanalyse mit R

3.1 Zeitreihenobjekte

R ist so organisiert, dass es *Objekte* gibt, die alle zu einer bestimmten *Klasse* gehören, von denen es eine Vielzahl gibt. Beispiele von Klassen, die wir früher bereits kennen gelernt haben sind *Vektoren*, *Faktoren*, *Data Frames*, *Modelloutput* und viele mehr. Somit ist es keine Überraschung, dass es auch eine Klasse für Zeitreihen gibt - genau genommen nicht nur eine einzige, sondern mehrere. Wir beschränken uns hier aber auf die Vorstellung und Erklärung von `ts`, die Standard-Klasse zur Zeitreihenanalyse in **R**. Sie kann nur Reihen mit äquidistanten Beobachtungen repräsentieren, aber für uns reicht dies aus.

3.1.1 Die Klasse `ts`

Um ein Objekt der Klasse `ts` zu definieren müssen einerseits die Daten zur Verfügung gestellt werden, jedoch mit dem Argument `start` auch der Zeitpunkt der ersten Beobachtung sowie die Frequenz für welche das Argument `frequency` zur Verfügung steht. Falls der Startzeitpunkt fehlt, so verwendet **R** den Default-Wert von 1, d.h. die Zeitpunkte werden mit $1, \ldots, n$ nummeriert. Die Frequenz ist die Anzahl Messungen pro Zeiteinheit, z.B. 1 für Jahresdaten, 4 für Quartalsdaten oder 12 für Monatsdaten. Das folgende Beispiel illustriert, wie es geht:

Beispiel: Wir betrachten hier eine kurze und einfache Zeitreihe, welche die Anzahl Tage mit Staus vor dem Nordportal des Gotthard-Strassentunnels enthält. Die Daten sind auf der Webseite des ASTRA zu beziehen.

Tabelle 1: Anzahl Tage mit Staus vor dem Nordportal des Gotthard-Strassentunels.

2004	2005	2006	2007	2008	2009	2010
88	76	112	109	91	98	139

Der Zeitpunkt der ersten Beobachtung ist 2004. Die Zeiteinheit beträgt 1 Jahr, und weil nur ein einziger Record pro Jahr vorliegt, ist auch die Frequenz gleich eins. Daraus können wir gleich schon schliessen, dass diese Reihe vielleicht einen Trend, sicher aber keinen Saisoneffekt haben wird, denn dies ist nur für periodische Reihen mit Frequenz > 1 möglich. Wir definieren das Zeitreihenobjekt:

```
> rawdat <- c(88, 76, 112, 109, 91, 98, 139)
> ts.dat <- ts(rawdat, start=2004, freq=1)
> ts.dat

Time Series:
Start = 2004
End = 2010
Frequency = 1
[1]  88  76 112 109  91  98 139
```

Es gibt in **R** mehrere einfache, aber nützliche Funktionen, mit welchen sich die Eigenschaften und Gegebenheit der Reihe eruieren lassen:

```
> start(ts.dat)

[1] 2004    1

> end(ts.dat)

[1] 2010    1

> frequency(ts.dat)

[1] 1

> deltat(ts.dat)

[1] 1
```

Während `start()`, `end()` und `frequency()` selbst erklärend sind oder bereits erläutert wurden, ist `deltat()` der Zeitabstand zwischen zwei Messungen. Weiter kann man sich auch die Zeiten angeben lassen, zu welchen die Zeitreihe beobachtet wurde:

```
> time(ts.dat)

Time Series:
Start = 2004
End = 2010
Frequency = 1
[1] 2004 2005 2006 2007 2008 2009 2010
```

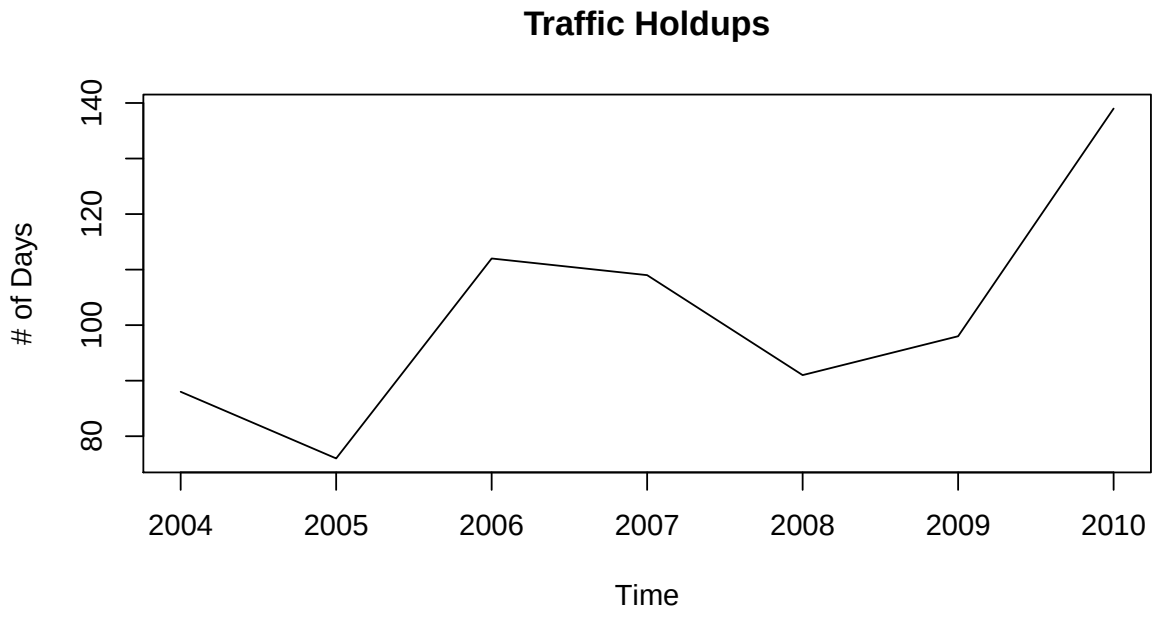
Eine nächste, relative simple, aber sehr nützliche Funktion ist `window()`. Sie ist dazu da, um Subset aus Zeitreihen zu extrahieren. Reguläres R-Subsetting im Stile von `ts.dat[2:5]` funktioniert zwar auch mit Zeitreihen. *Auf diese Weise erhält man aber ein Objekt der Klasse numeric und nicht eine Zeitreihe!* Mit `window()` ist dies anders, man erhält ein `ts`-Objekt, darum wird dieses Vorgehen meistens bevorzugt:

```
> window(ts.dat, start=2006, end=2008)

Time Series:
Start = 2006
End = 2008
Frequency = 1
[1] 112 109  91
```

Natürlich gibt es neben diesen Grundfunktionalitäten noch weitere nützliche Funktionen für die Klasse `ts`. Mit dazu gehört die Funktion `plot()`, sowie viele weitere zur Schätzung von Trend, Saisoneffekt und Abhängigkeitsstruktur, ebenso solche zum Anpassen von Zeitreihenmodellen und zum Erzeugen von Vorhersagen. Diese werden in den nächsten Kapiteln präsentiert.

Hier schliessen wir mit einem Zeitreihenplot für die Anzahl Stautage ab, siehe nächste Seite. Weil die Reihe nur 7 Beobachtungen aufweist, ist es schwierig, eine Aussage bzgl. Trend oder stochastische Abhängigkeit zu machen.

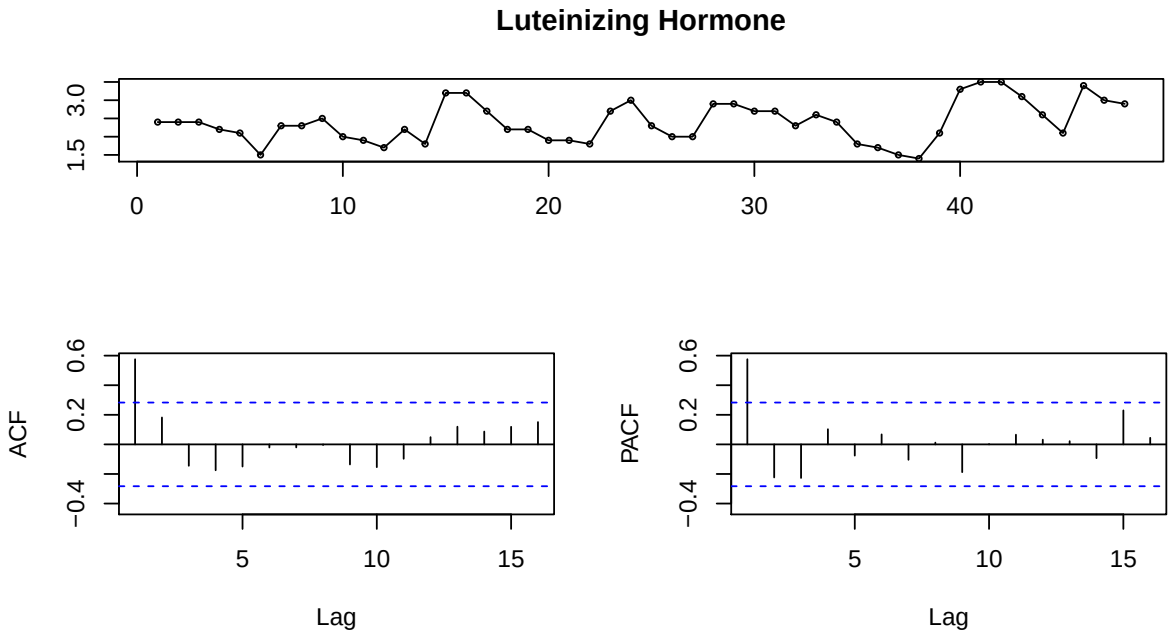
```
> plot(ts.dat, ylab='# of Days', main='Traffic Holdups')
```



3.2 Das Package `forecast`

Das **R**-Package `forecast` bietet zusätzliche Methoden und Werkzeuge um univariate Zeitreihen darzustellen und zu analysieren. Dazu gehören im Besonderen die *automatische* Generierung von Zeitreihenmodellen die auf exponentieller Glättung via Zustandsraum-Modellen basieren und eine sehr anwenderfreundliche Möglichkeit bieten, Prognosen inklusive einer Angabe zur Prognoseunsicherheit zu berechnen. Das Paket ermöglicht zudem eine automatischen Generierung von ARIMA-Modellen. Ein Teil der Funktionen wird in den entsprechenden Abschnitten im Skript eingeführt. Als Beispiel wird hier die Funktion `tsdisplay()` erwähnt, mit der eine Zeitreihe direkt mit dem emprischen Autokorrelogramm (Darstellung der ACF = auto correlation function) und dem empirischen partielle Autokorrelogramm (PACF = partial auto correlation function) dargestellt wird.

```
> library(forecast)
> data(lh)
> tsdisplay(lh, main='Luteinizing Hormone')
```



Beide Plots (ACF und PACF) sind zentral für die klassische Modellierung von stationären Zeitreihen. Im Abschnitt 4.3 finden Sie die mathematischen Definitionen dieser Funktionen sowie eine ausführliche Beschreibung zu ihrer Anwendung.

3.3 Zeit und Datum in R

Im Rahmen dieses Modules werden wir nicht explizit mit Zeit und Datum hantieren müssen. Dies auch, weil das Problem in der Klasse `ts` elegant umschifft wird: wegen den als äquidistant vorausgesetzten Beobachtungen ist es möglich, die Zeitpunkte zu nummerieren, und die Angabe vom Zeitpunkt der ersten Beobachtung ist ausreichend. Wer jedoch praxisnah mit Zeitreihen arbeitet, wird kaum darum herum kommen, Zeit und Datum explizit in **R** darzustellen. Daher folgt an dieser Stelle eine ausführliche Erklärung, wie man das am besten macht.

Es gibt in **R** mehrere verschiedene Möglichkeiten, Zeitpunkte zu repräsentieren. Das in der Grundfunktionalität vorhandene `as.Date()` kann ein Datum, nicht aber Zeiten darstellen. Weiter gibt es das Add-on-Package `chron`. Mit diesem kann man das Datum und die Zeit darstellen, verschiedene Zeitzonen werden aber nicht unterstützt. Doch auch hierfür gibt es eine Lösung, nämlich die Klassen `POSIXct` und `POSIXlt`. Was soll man also tun? Als Grundregel gilt: man wähle die einfachste Methode, die das bietet, was man

braucht.

3.3.1 Die Klasse `Date`

Wie oben erwähnt, ist die Funktion `as.Date()` die einfachste Möglichkeit, Kalendertage in **R** darzustellen. Dank dem Argument format kann jedes Datum eingelesen werden. Der Default ist Jahr-Monat-Tag, mit Bindestrich oder Slash:

```
> as.Date('2012-02-14')
[1] "2012-02-14"
> as.Date('2012/02/07')
[1] "2012-02-07"
```

Doch wie erwähnt, auch andere Formate sind lesbar, selbst ein Datum in Textform. Für das Argument `format` gibt es eine Formelsprache. Wir präsentieren hier die wichtigsten Kürzel, weitere Informationen dazu findet man in der **R**-Hilfe zur Funktion `strptime()`.

```
> as.Date('2012-02-14')
[1] "2012-02-14"
> as.Date('2012/02/07')
[1] "2012-02-07"
```

Code	Bedeutung
`%d`	Day of the month (decimalnumber)
`%m`	Month (decimal number)
`%b`	Month (character, abbreviated)
`%B`	Month (character, full name)
`%y`	Year (decimal, two digit)
`%Y`	Year (decimal, four digit)

Die folgenden Beispiele zeigen die praktische Anwendung:

```
> as.Date('27.01.12',format='%d.%m.%y')
[1] "2012-01-27"
> as.Date('14. Februar, 2012',format='%d. %B, %Y')
[1] NA
```

German?!

Objekte der Klasse `Date` werden intern numerisch gespeichert, nämlich als Anzahl Tage, die seit dem 1. Januar 1970 vergangen ist, frühere Zeitpunkte erhalten einfach eine negative Zahl. Mit dem Befehl `as.numeric()` kann man auf die Zahl zugreifen. Ebenso kann jede Zahl in das entsprechende Datum verwandelt werden, wie das Beispiel unten zeigt:

```
> mydat <-as.Date('2012-02-14')
> ndays <-as.numeric(mydat)
> ndays
[1] 15384
> tdays <- 10000
> class( tdays ) <- "Date"
> tdays
[1] "1997-05-19"
```

10'000 Tage seit dem 01.01.1970

Eine weitere, sehr nützliche Funktionalität ist die Möglichkeit, sich den Wochentag, Monat und das Quartal eines Datums angeben zu lassen. Die erhaltenen Werte kann man zu einer Faktorvariable verarbeiten, was nützlich für grafische Darstellung, Zerlegung von Zeitreihen oder auch für Regression ist.

```
> weekdays(mydat)
[1] "Tuesday"
> months(mydat)
[1] "February"
> quarters(mydat)
[1] "Q1"
```

Auf den ersten Blick mag es vielleicht erstaunlich anmuten: man kann mit Objekten der Klasse `Date` sogar rechnen. Nun gut, so erstaunlich ist das nicht, handelt es sich doch um die Anzahl Tage seit dem 1. Januar 1970.

```
> dat <- as.Date(c('2000-01-01','2004-04-04','2007-08-09'))
> dat
[1] "2000-01-01" "2004-04-04" "2007-08-09"
> min(dat)
[1] "2000-01-01"
> max(dat)
[1] "2007-08-09"
> median(dat)
[1] "2004-04-04"
> dat[3]-dat[1]
Time difference of 2777 days
```

Ebenfalls nützlich ist die Möglichkeit, sich Zeitsequenzen generieren zu können. Als erstes Beispiel erzeugen wir einen Vektor, wo 12 aufeinander folgende Tage ab dem 3. August 1985 präsent sein sollen:

```
> seq(as.Date('1985-08-03'), by='days', length=12)
 [1] "1985-08-03" "1985-08-04" "1985-08-05" "1985-08-06" "1985-08-07"
 [6] "1985-08-08" "1985-08-09" "1985-08-10" "1985-08-11" "1985-08-12"
[11] "1985-08-13" "1985-08-14"
```

Die Funktion `seq()` kennt auch das Argument by. Hier kann man verschiedene Zeiteinheiten angeben, und sogar Zahlen davor platzieren. Wir erzeugen einen Vektor mit 12 Zeitpunkten im 2-Wochen-Abstand:

```
> seq(as.Date('1992-04-17'), by='2 weeks', length=12)
 [1] "1992-04-17" "1992-05-01" "1992-05-15" "1992-05-29" "1992-06-12"
 [6] "1992-06-26" "1992-07-10" "1992-07-24" "1992-08-07" "1992-08-21"
[11] "1992-09-04" "1992-09-18"
```

3.3.2 Das Package `chron`

Das Package wird mit `library(chron)` geladen. Seine Hauptfunktion ist `chron()`. Sie wird benützt, um in **R** einen Zeitpunkt zu definieren. Achtung: Datum und Uhrzeit müssen separat eingegeben werden. Falls sie in der ursprünglichen Quelle in einem einzigen String vorliegen, müssen sie erst getrennt werden, wofür die Funktionen `substr()` und `strsplit()` hilfreich sind.

Verschiedene Zeitzonen und/oder Sommerzeit gibt es in der Repräsentation von `chron()` nicht. Intern wird jeder Zeitpunkt als numerischer Wert gespeichert, und zwar auch wieder als die vergangene Anzahl Tage seit dem 1. Januar 1970. Im Unterschied zur Klasse `Date` sind hier aber nun Kommastellen erlaubt. Die numerischen Werte sind wiederum via `as.numeric()` zugänglich. Das folgende Beispiel illustriert den Gebrauch der Funktion:

```
> library(chron)
> dat <-c('2007-06-09 16:43:20','2007-08-29 07:22:40',
+         '2007-10-21 16:48:40', '2007-12-17 11:18:50')
> dts <- substr(dat, 1, 10)
> tme <- substr(dat, 12,19)
> fmt <- c('y-m-d','h:m:s')
> cdt <- chron(dates=dts, time=tme, format = fmt)
> cdt
[1] (07-06-09 16:43:20) (07-08-29 07:22:40) (07-10-21 16:48:40)
[4] (07-12-17 11:18:50)
```

extract Dates and Times

Weil die Zeitpunkte ja auch hier als numerische Werte repräsentiert werden, stehen auch hier Summary-Statistiken und Rechenoperationen zur Verfügung. Speziell ist hier, dass Zeitdifferenzen nun in Wochen, Tagen, Stunden, Minuten, Sekunden etc. dargestellt werden können.

```
> library(chron)
> cdt[2]-cdt[1]
Time in days:
[1] 80.61065
> difftime(cdt[2], cdt[1], units = "secs")
Time difference of 6964760 secs
```

3.3.3 Die `POSIX`-Klassen

Die beiden Klassen `POSIXct` und `POSIXlt` implementieren ebenfalls Zeit und Datum, wobei hier auch unterschiedliche Zeitzonen und die Sommerzeit korrekt gehandelt werden können. Dies ist zwar schön, bringt aber eine gewisse Komplexität mit sich, so dass es sich empfiehlt, diese Funktionalität nur einzusetzen, wenn man sie auch tatsächlich braucht. Wir verzichten an dieser Stelle auf weitere Details und verweisen auf die **R**-Hilfe.

4 Deskriptive Zeitreihenanalyse

Auch Zeitreihen sind vorerst einmal nichts anderes als ein Zahlenhaufen. Ohne geeignete Darstellung und/oder Zusammenfassung ist es schwierig, die Übersicht zu behalten und etwas aus den Daten zu lernen. Deshalb erklären wir in diesem Kapitel, wie Zeitreihen grafisch dargestellt werden können, erläutern dann die Zerlegung in Trend, Saisoneffekt und stationären Teil, und schliessen mit den Methoden, welche das Aufzeigen der Abhängigkeitsstruktur erlauben.

4.1 Visualisierung

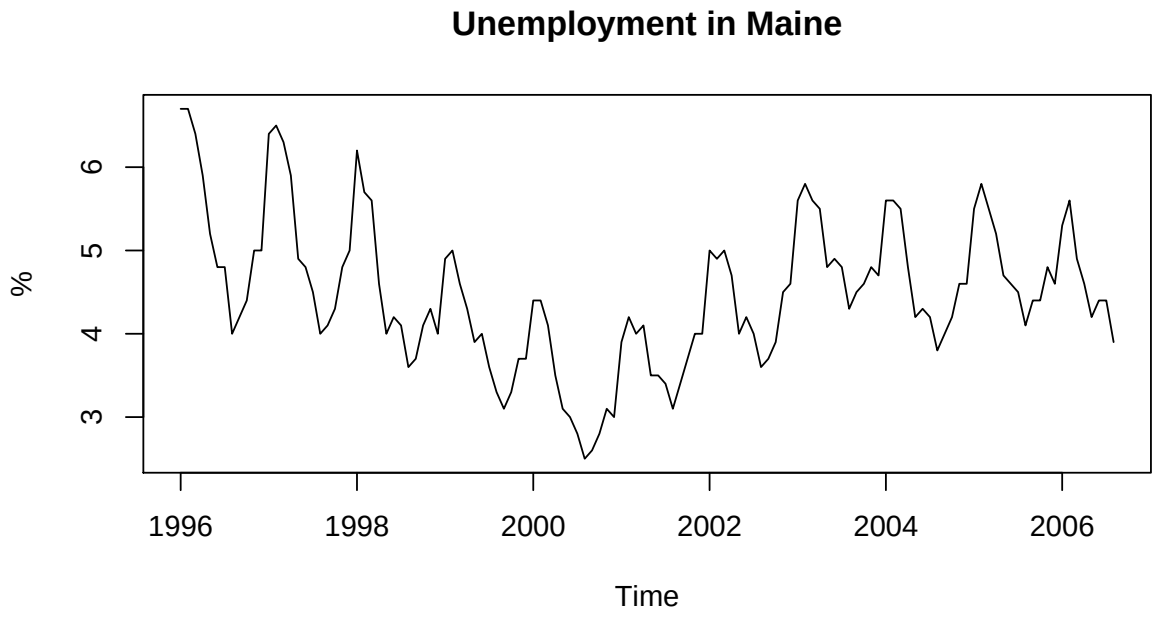
4.1.1 Zeitreihenplot

Dem Zeitreihenplot sind wir bereits in Kapitel 1.2 begegnet. Er ist das wichtigste Mittel zur Visualisierung von Zeitreihen. Mit der generischen Funktion `plot()`, angewandt auf ein Zeitreihen-Objekt, ist er auch rasch erzeugt. Als Beispiel verwenden wir die monatliche Arbeitslosenrate im US-Bundesstaat Maine, von Januar 1996 bis August 2006. Die Daten stehen auf dem Web als Textfile zur Verfügung und können direkt eingelesen werden:

```
> dat <- read.table(paste(data.dir, 'Maine.dat', sep = ''), header=TRUE)
> tsd <- ts(dat, start=c(1996,1), freq=12)
```

Die Reihe zeigt sowohl einen Trend wie auch einen Saisoneffekt und ist nicht stationär. Weil die Arbeitslosenrate ein wichtiger ökonomischer Indikator ist, interessiert speziell auch die Vorhersage zukünftiger Werte.

```
> plot(tsd, ylab='%', main='Unemployment in Maine')
```

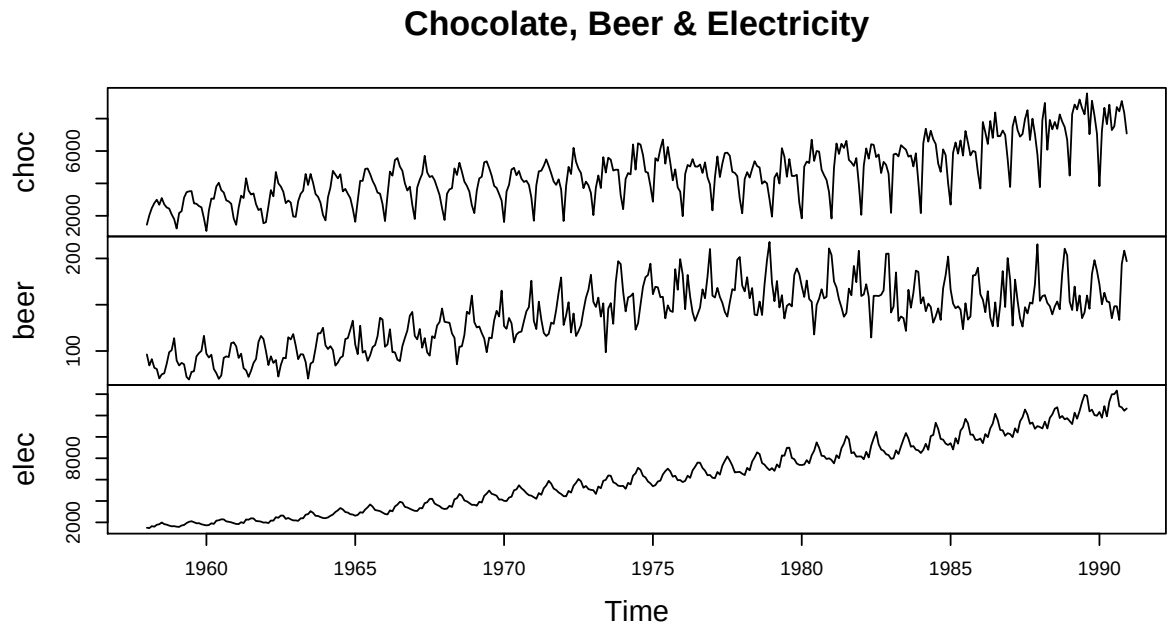


Ein wichtiger Aspekt ist die Wahl von Breite und Höhe des Frames: wenn man die Zeitachse zu stark komprimiert, so kann es schwierig werden, die Eigenschaften einer Zeitreihe herauszulesen. Wie macht man es denn richtig? Als Faustregel gilt das „banking to 45 degrees“, d.h. die durchschnittliche Steigung (im Betrag) sollte ca. 45 Grad betragen. Für sehr lange Zeitreihen kann dies schwierig sein. In jenem Fall gibt es die Möglichkeit, die Reihe mit dem Befehl `window()` aufzutrennen und in verschiedenen Frames darzustellen.

4.1.2 Darstellung mehrerer Zeitreihen

Ziemlich häufig hat man über einen bestimmten Zeitraum nicht nur eine Datenreihe beobachtet, sondern gleich mehrere. An dieser Stelle geht es darum, wie man diese in R korrekt definiert und übersichtlich darstellt. Das verwendete Beispiel enthält monatliche Werte für die Elektrizitäts-, Bier- und Schokolade-Produktion in Australien für den Zeitraum 1958-1990. Die Daten sind vom Australian Bureau of Statistics, und leider nicht mehr online als Textfile greifbar.

```
> load(paste(data.dir, 'cbe.rda', sep = ''))
> dat <- cbe
> tsd <- ts(dat, start=1958, freq=12)
> plot(tsd, main='Chocolate, Beer & Electricity')
```

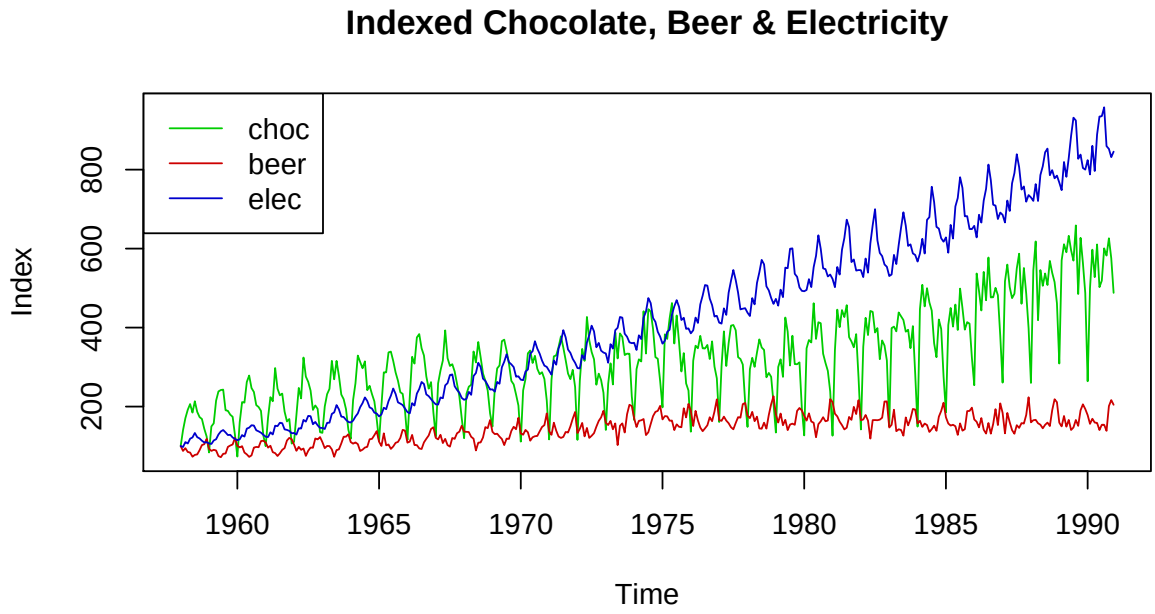


Jede der drei Zeitreihen weist sowohl einen Trend wie auch einen Saisoneffekt auf, daher sind sie nicht stationär. Ebenso ist es wissenswert, dass die australische Bevölkerung in der Beobachtungsperiode um den Faktor 1.8 zugenommen hat – die Reihen zum Teil massiv stärker, wie wir später sehen.

Wie dem Code auf der vorangehenden Seite zu entnehmen ist, können die in **R** als multiples Zeitreihen-Objekt verfügbaren Daten mit dem `plot()`-Befehl visualisiert werden. Man erhält für jede Reihe ein eigenes Frame, was generell als die empfehlenswerteste Darstellung gilt. Doch es gibt auch andere Fälle, wo es hilfreich ist, wenn die Reihen in einem einzigen Frame gezeigt werden. Dies bedingt aber, dass in derselben Einheit/Skala gemessen wird, oder, dass eine Indexierung stattgefunden hat. Letzteres wollen wir an diesem Beispiel illustrieren.

```
> # Indexierung der Reihen
> tsd[,1] <- tsd[,1]/tsd[1,1]*100
> tsd[,2] <- tsd[,2]/tsd[1,2]*100
> tsd[,3] <- tsd[,3]/tsd[1,3]*100
>
> # Darstellung in einem einzigen Frame
> clr <- c('green3','red3', 'blue3')
> plot.ts(tsd[,1], ylim=range(tsd), ylab='Index', col=clr[1])
> title('Indexed Chocolate, Beer & Electricity')
> lines(tsd[,2], col=clr[2])
> lines(tsd[,3], col=clr[3])
```

```
>
> # Legende
> legend('topleft',lty=1, col=clr, legend=names(dat))
```



Die indexierten Reihen können nun problemlos in einem einzigen Frame geplottet werden. Dies erlaubt einen sehr guten Vergleich der relativen Entwicklung, was mit multiplen Frames kaum möglich ist. Wir erkennen so, dass die Elektrizitätsproduktion von 1958 bis 1990 um den Faktor 8 zugenommen hat, dieser bei Schokolade ca. 4 beträgt, und er für Bier kleiner als 2 ist. Ebenso zeigt sich, dass der Saisoneffekt bei Schokolade am stärksten ist, gefolgt von Schokolade, und am kleinsten beim Bier.

4.2 Deskriptive Zerlegung

4.2.1 Grundlagen

Wie zuvor erklärt wurde, ist die Stationarität einer Zeitreihe eine wichtige Grundlage, um statistische Analysen, wie z.B. das Studium der seriellen Korrelation, durchführen zu können. In der Praxis weisen aber viele Zeitreihen Trends und/oder Saisoneffekte auf und sind deshalb nicht stationär. Dann bietet es sich an, eine deskriptive Zerlegung durchzuführen. Im additiven Fall lautet das Modell wie folgt:

$$X_t = m_t + s_t + E_t \tag{1}$$

wo X_t die Zeitreihe zum Zeitpunkt t ist, m_t den Trend, s_t den Saisoneffekt und E_t den Restterm darstellt, d.h. eine Sequenz von in der Regel korrelierten Zufallsgrössen. Unsere

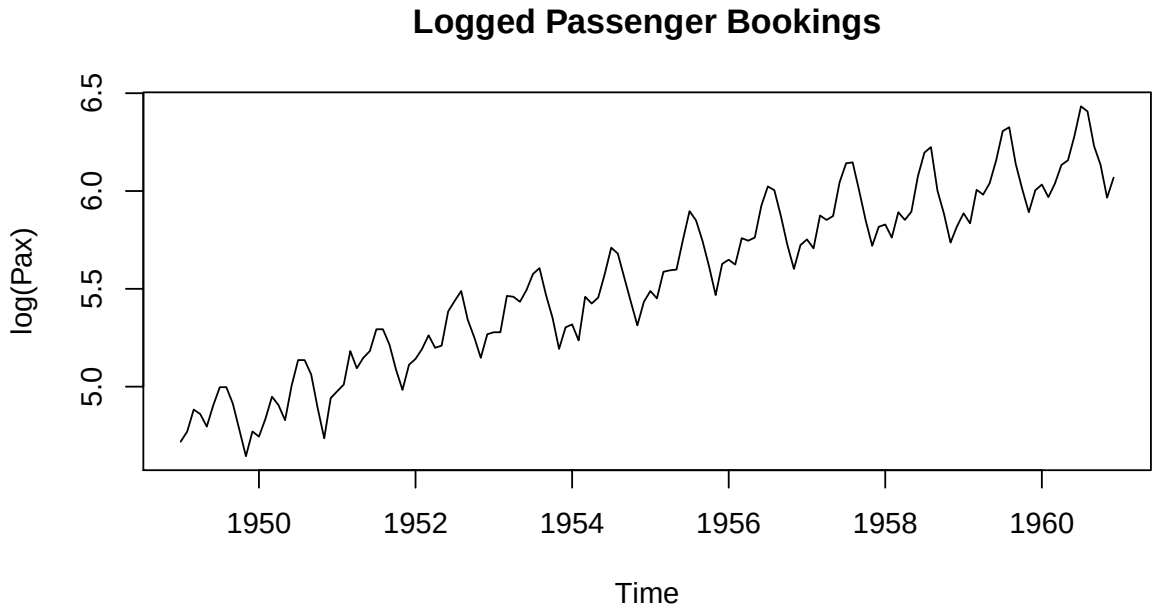
Absicht ist, m_t und s_t zu schätzen, und zwar so, dass E_t eine stationäre Zeitreihe ist.

Nicht selten beobachtet man Zeitreihen, wo der Saisoneffekt und auch die zufällige Streuung grosser werden, wenn das Niveau der Reihe ansteigt. Die Air Passenger Bookings sind ein Beispiel dafür. In diesem Fall kann man ein multiplikatives Zerlegungsmodell ansetzen:

$$X_t = m_t \cdot s_t \cdot E_t \tag{2}$$

Führt man eine Logarithmus-Transformation auf das multiplikatives Zerlegungsmodell (2) durch, so sind wir zurück im additiven Fall:

$$log(X_t) = log(m_t) + log(s_t) + log(E_t) = m'_t + s'_t + E'_t \tag{3}$$



In der vorangehenden Abbildung sind die logarithmierten Air Passenger Bookings dargestellt. Im Gegensatz zum Original sind hier nun Saisoneffekt und Streuung nicht mehr vom Niveau abhängig. Das heisst, dass hier eine multiplikative Zerlegung passend ist. Wer genau hinschaut, erkennt in den transformierten Daten einen sich über die Zeit verändernden Saisoneffekt: der Nebengipfel im Frühling wird immer kleiner, der Hauptgipfel im Sommer immer grösser.

4.2.2 Glätten und Filtern

Um die deskriptive Zerlegung in der Praxis durchführen zu können, brauchen wir eine Methode, welche Schätzungen $\widehat{m}_t$, $\widehat{s}_t$ und $\widehat{E}_t$ für Trend, Saisoneffekt und Restterm liefert.

Eine intuitive Idee zur Trendschätzung ist das gleitende Mittel. Im Zeitreihen-Kontext spricht man auch von einem *additiven linearen Filter*.

$$\widehat{m}_t = \frac{1}{2p+1} \sum_{i=-p}^{p} X_{i+1} \tag{4}$$

Diese Methode haben wir bereits früher (Modul: Lineare Regression) als *Running Mean* kennen gelernt. Sie wird normalerweise zu jedem Beobachtungszeitpunkt ausgewertet. Weil diese äquidistant sind, befindet sich stets dieselbe Anzahl Datenpunkte im Fenster.

Beispiel: Trendschätzung mit Running Mean

Als Beispiel betrachten wir die Zeitreihe des Verkehrsindex Schweiz. Sie wird vom ASTRA erhoben und zeigt die Zunahme der Mobilität auf den Strassen auf, hier im Zeitraum zwischen 1990-2010. Das Startjahr wurde mit dem Wert 100 indexiert, so dass man danach die relative Zunahme ablesen kann. Die Daten sind wie folgt:

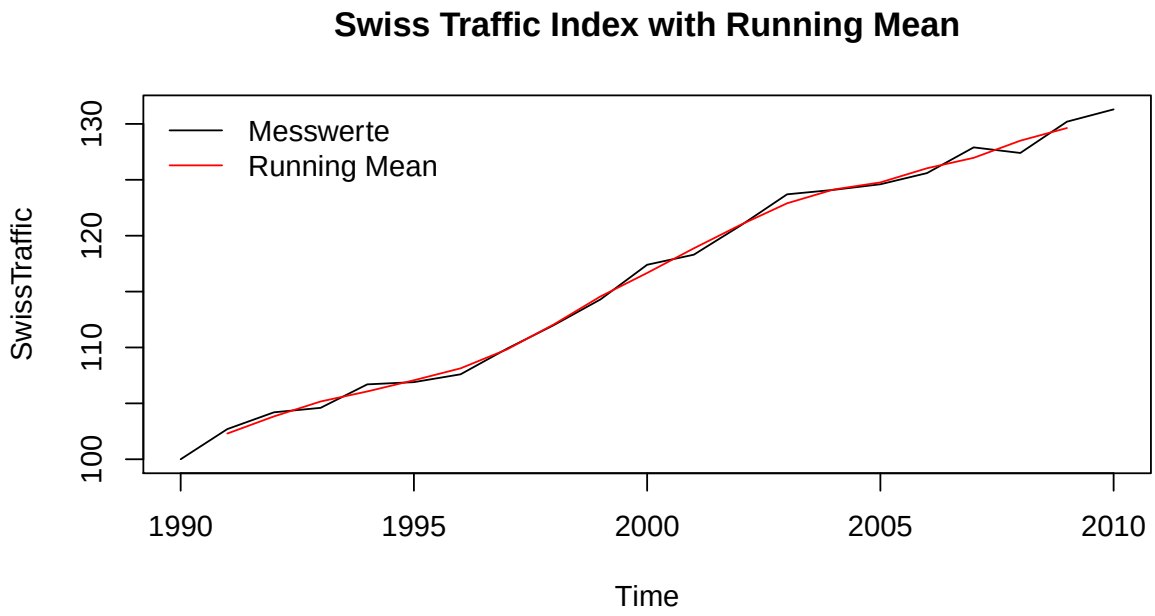
```
>  SwissTraffic <- ts(c(100.0, 102.7, 104.2, 104.6, 106.7,
+                      106.9, 107.6, 109.9, 112.0, 114.3,
+                      117.4, 118.3, 120.9, 123.7, 124.1,
+                      124.6, 125.6, 127.9, 127.4, 130.2,
+                      131.3), start=1990, freq=1)
```

Die Reihe zeigt deutlich einen ansteigenden Trend. Ein Saisoneffekt ist nicht sichtbar, und ist auch nicht zu erwarten. Es liegt ja nur 1 Messung pro Zeiteinheit (d.h. pro Jahr) vor. Wir schätzen nun den Trend mit einem Running Mean, welches die aktuelle und die beiden benachbarten Beobachtungen enthalten soll. In **R** kann man dies bequem mit der Funktion `filter()` ausführen.

```
> trend.est <- filter(SwissTraffic, filter=c(1,1,1)/3)
> trend.est

Time Series:
Start = 1990
End = 2010
Frequency = 1
 [1]       NA 102.3000 103.8333 105.1667 106.0667 107.0667 108.1333 109.8333
 [9] 112.0667 114.5667 116.6667 118.8667 120.9667 122.9000 124.1333 124.7667
[17] 126.0333 126.9667 128.5000 129.6333       NA
```

Für die erste und die letzte Beobachtung gibt es keine Trendschätzung, weil dort nicht ein volles Fenster beobachtet wurde. Generell gilt: je grösser man das Fenster wählt, desto glatter verläuft der Trend, umso mehr Beobachtungen am Anfang und am Ende der Reihe erhalten aber keine Schätzung. Grafisch dargestellt ist das Ergebnis wie folgt:

```
> plot(SwissTraffic)
> title('Swiss Traffic Index with Running Mean')
> lines( trend.est, type ="l", col = "red")
> legend( "topleft", legend = c("Messwerte", "Running Mean"),
+         col= c("black", "red"), lty = 1, bty = "n")
```



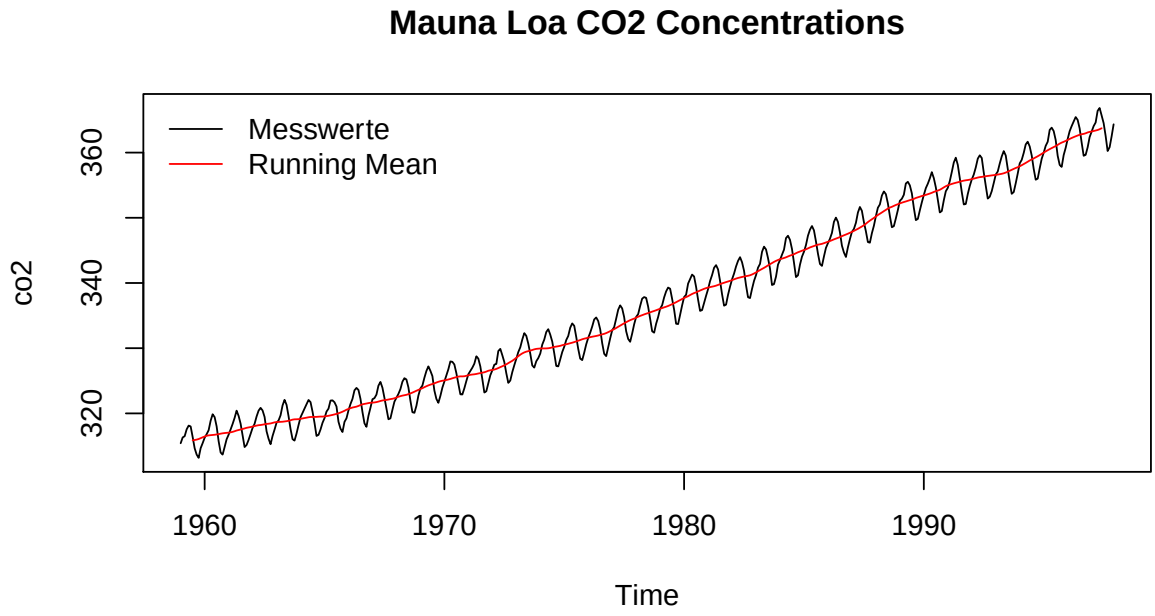
Trendschätzung für saisonale Daten

Bei Zeitreihen, die einen Saisoneffekt aufweisen, kann der Trend ebenfalls mit einem linearen Filter geschätzt werden. Man muss aber sicherstellen, dass im Fenster gerade eine komplette Periode enthalten ist. Für Monatsdaten sieht das wie folgt aus:

$$\widehat{m}_t = \frac{1}{12}\big(\frac{1}{2}X_{t-6} + X_{t-5} + X_{t+5} + \frac{1}{2}X_{t+6} \cdots + \big) \text{for } t = 7, \ldots, n-6 \tag{5}$$

Weil es 12 Monate gibt, sprich eine gerade Zahl, zählen die beiden Beobachtungen am Rand des Fensters nur halb. Konkret: wenn wir den Trend für Dezember schätzen wollen, so verwenden wir alle Beobachtungen von Juli desselben Jahres bis Mai nächsten Jahres. Beim Juni verwenden wir die Beobachtung vom aktuellen und vom nächsten Jahr, je mit der Hälfte des Gewichts. Durchführen kann man das erneut mit der **R**-Funktion `filter()`. Als Beispiel verwenden wir hier die von Januar 1959 bis Dezember 1997 gemessenen, durchschnittlichen

```
> data(co2)
> wghts <- c(.5,rep(1,11),.5)/12
> trend.est <- filter(co2, filter=wghts, sides=2)
> plot(co2, main='Mauna Loa CO2 Concentrations')
> lines(trend.est, col='red')
> legend( "topleft", legend = c("Messwerte", "Running Mean"),
+         col= c("black", "red"), lty = 1, bty = "n")
```



Wie man in der Abbildung oben sieht, folgt der Trend den Daten sehr gut, aber ohne die saisonalen Schwankungen mitzumachen. Er ist nicht linear, sondern steigt progressiv an und weist einige Dellen auf.

Natürlich gibt es neben dem hier vorgestellten Running Mean noch etliche weitere Methoden, mit welchen wir den Trend schätzen können. Dazu gehören z.B. Running Median, Loess sowie viele weitere. Wir gehen aber nicht ins Detail.

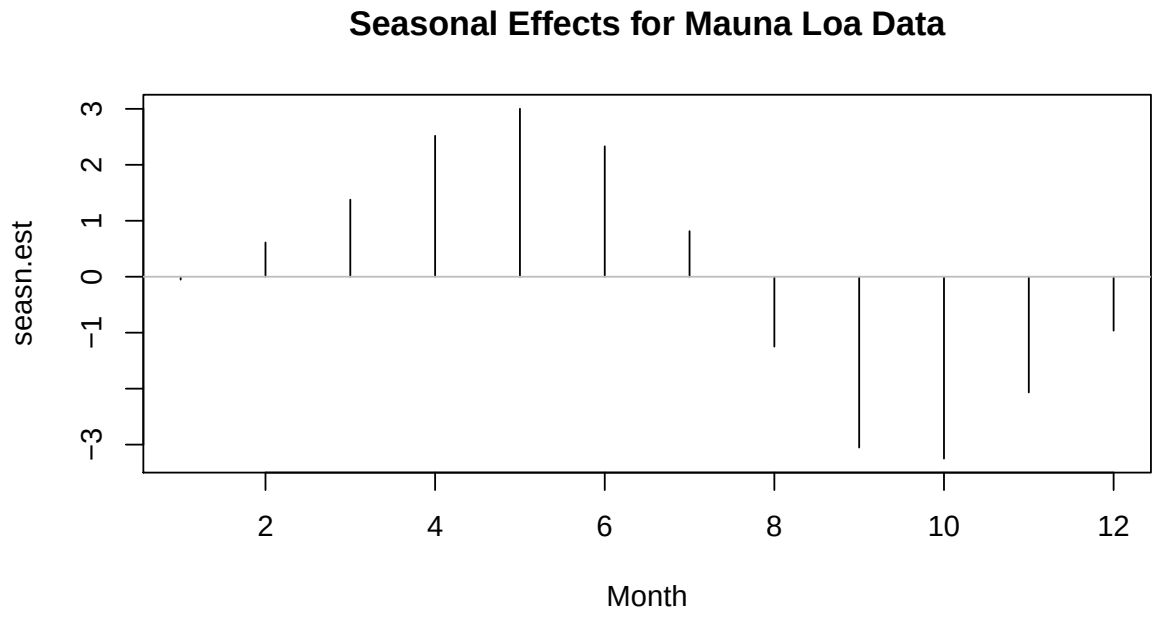
Schätzung des Saisoneffekts

Wir betrachten hier weiterhin die CO_2 -Konzentrationen. Den Trend haben wir oben erfolgreich mit einem linearen Filter geschätzt. Das Ziel ist jetzt, auch für den Saisoneffekt eine Schätzung anzugeben. Dies macht man, indem man monatsweise über die trendbereinigten Daten mittelt, d.h. über alle Januar-, Februar-, März-, ... Werte mittelt.

$$\widehat{s}_{\text{Jan.}} = \widehat{s}_1 = \widehat{s}_{13} = \cdots = \frac{1}{39}\sum_{j=0}^{38}(x_{12j+1} - \widehat{m}_{12j+1}) \tag{6}$$

Die Mittelung umfasst jeweils 39 Terme, weil wir so viele Jahre an Beobachtungen haben. In **R** implementiert man das am einfachsten, indem man eine Faktorvariable für den Monat bildet und dann die Funktion `tapply()` verwendet. Die `NA`s zu Beginn und Ende werden in der Mittelung nicht berücksichtigt.

```
> trend.adj <-  co2-trend.est
> month <- factor(rep(1:12,39))
> seasn.est <- tapply(trend.adj, month, mean, na.rm=TRUE)
> plot(seasn.est, type='h', xlab='Month')
> title('Seasonal Effects for Mauna Loa Data')
> abline(h=0, col='grey')
```

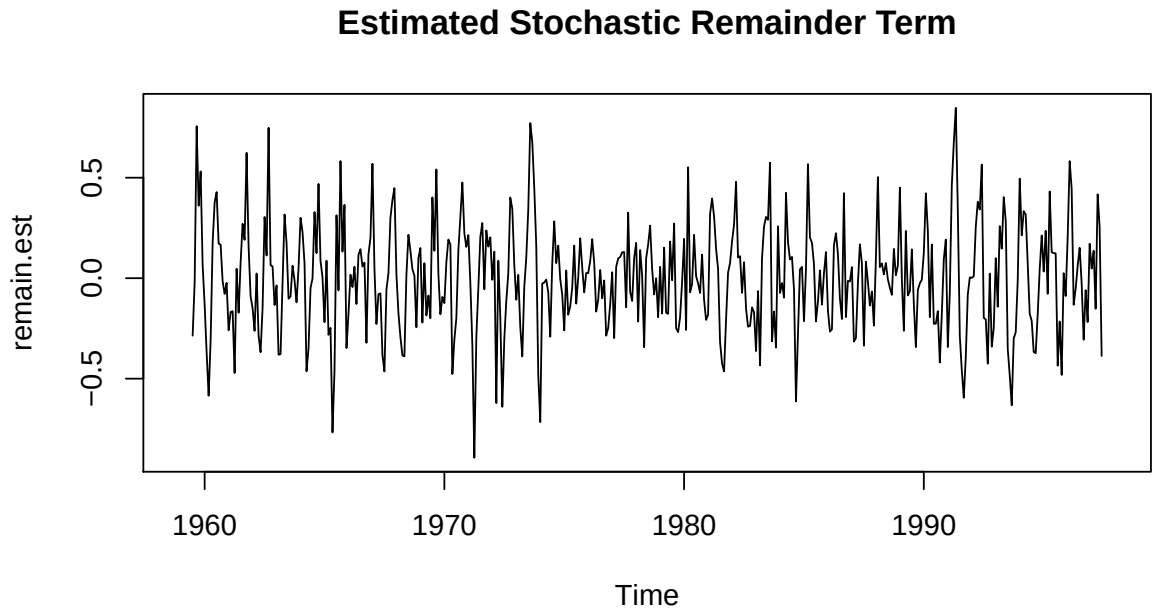


Im obigen Plot sind die Saisoneffekte dargestellt. Die höchsten CO_2-Werte treten jeweils im Mai auf, und das Minimum ist im Oktober. Die nächste Aufgabe ist nun das Bestimmen des Restterms. Dazu werden einfach Trend- und Saisonschätzung von der Reihe subtrahiert.

$$\widehat{E}_t = x_t - \widehat{m}_t - \widehat{s}_t \tag{7}$$

Das Resultat ist unterhalb dargestellt. Es scheint, als ob in den Daten immer noch eine gewisse Periodizität vorhanden ist. Dies könnte ein Hinweis darauf sein, dass die Zerlegung nicht geglückt ist. Eine vertiefte Analyse zeigt, dass der Saisoneffekt hier nicht konstant ist. Am Anfang wird er unter-, am Schluss dann überschätzt. Es gibt jedoch Methoden, die dies beheben können.

```
> remain.est <-  co2- trend.est - rep(seasn.est, 39)
> plot( remain.est)
> title('Estimated Stochastic Remainder Term')
```

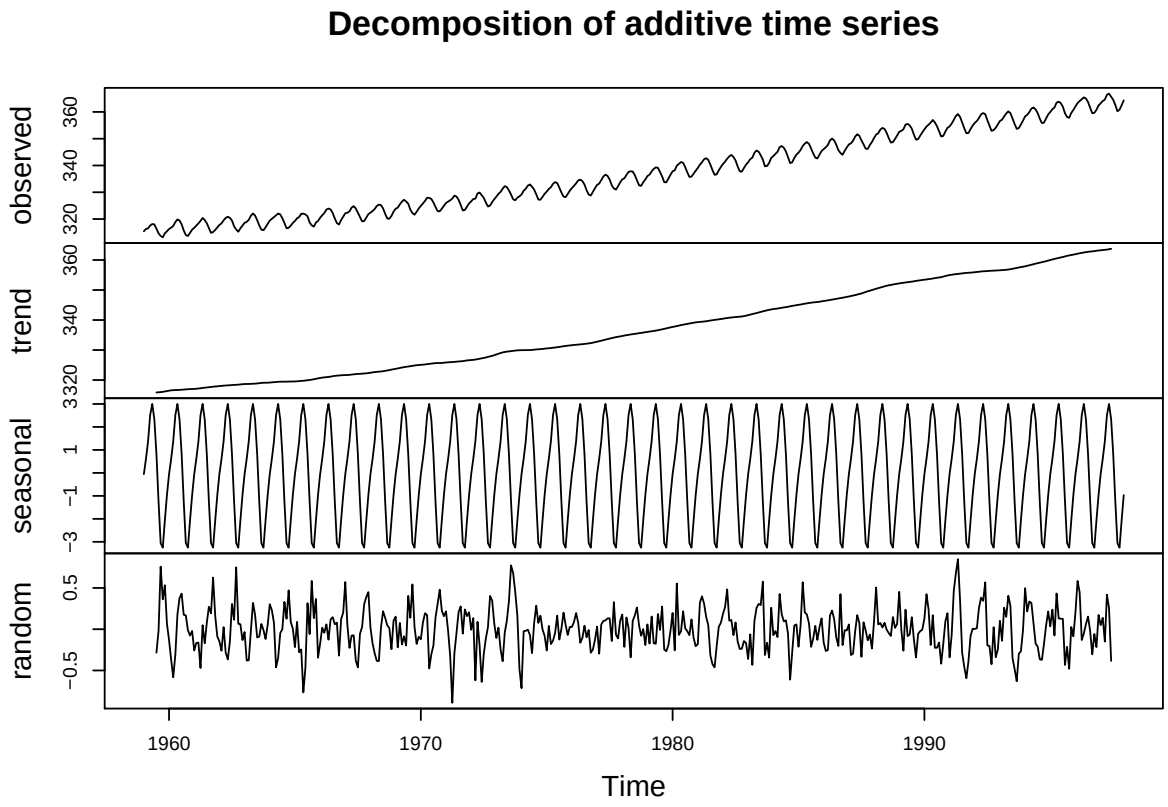


Nach diesen Erklärungen zur deskriptiven Zerlegung mag die Frage aufkommen, ob es nicht auch ein bisschen einfacher geht. Die Antwort darauf ist „ja“. In **R** gibt es die Funktion `decompose()`, welche das komplette, oben diskutierte Vorgehen quasi auf Knopfdruck implementiert. *Ein Hinweis*: `decompose()` kann nur für zyklische Reihen verwendet werden, wo mindestens zwei komplette Perioden an Beobachtungen vorliegen!

```
> co2.dec <- decompose(co2)
```

Zur Funktion `decompose()` gibt es auch eine bequeme Plot-Funktion, welche Originalreihe sowie die Schätzungen für Trend, Saison und Rest grafisch darstellt. Bis auf die unterschiedliche Darstellung sind die Resultate aber identisch zu dem, was wir zuvor „von Hand“ produziert hatten.

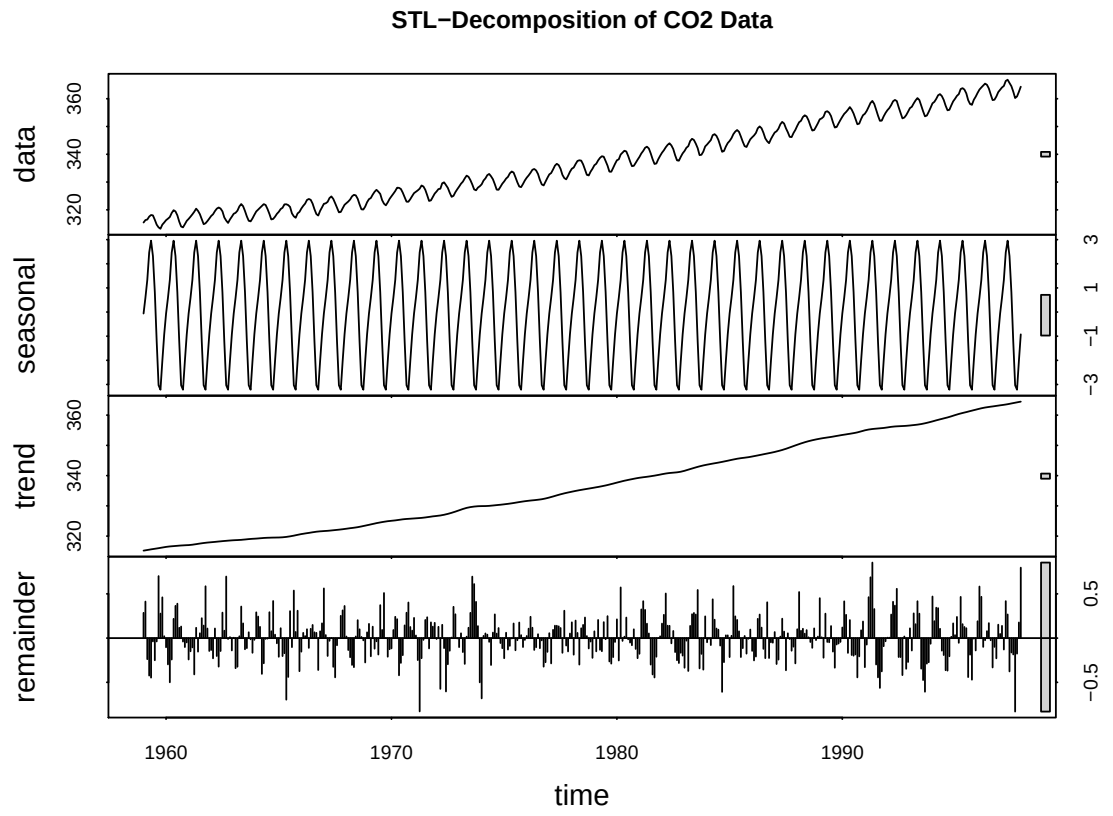
```
> plot(co2.dec)
```



4.2.3 Die STL-Prozedur

Wir hatten bereits beim Thema Regression gelernt, dass es Glätter mit besseren Eigenschaften wie das Running Mean gibt. Wegen den äquidistanten Beobachtungen sind dessen Probleme in der Zeitreihenanalyse nicht ganz so akzentuiert, aber die Verwendung eines LOESS-Glätters ist auch hier zu bevorzugen. Kommt hinzu, dass dies auch sehr bequem möglich ist, nämlich mit der Prozedur `stl()`. Deren Output ist (bis auf Details) äquivalent zu demjenigen von `decompose()`. Hingegen sind die algorithmischen Details, um die wir uns hier aber nicht kümmern wollen, deutlich komplexer. *Auf jeden Fall ist die* ***`stl`****-Methode robuster gegen Ausreisser und funktioniert damit in der Praxis meist besser.*

```
> co2.stl <- stl(co2, s.window='periodic')
> plot(co2.stl, main='STL-Decomposition of CO2 Data')
```



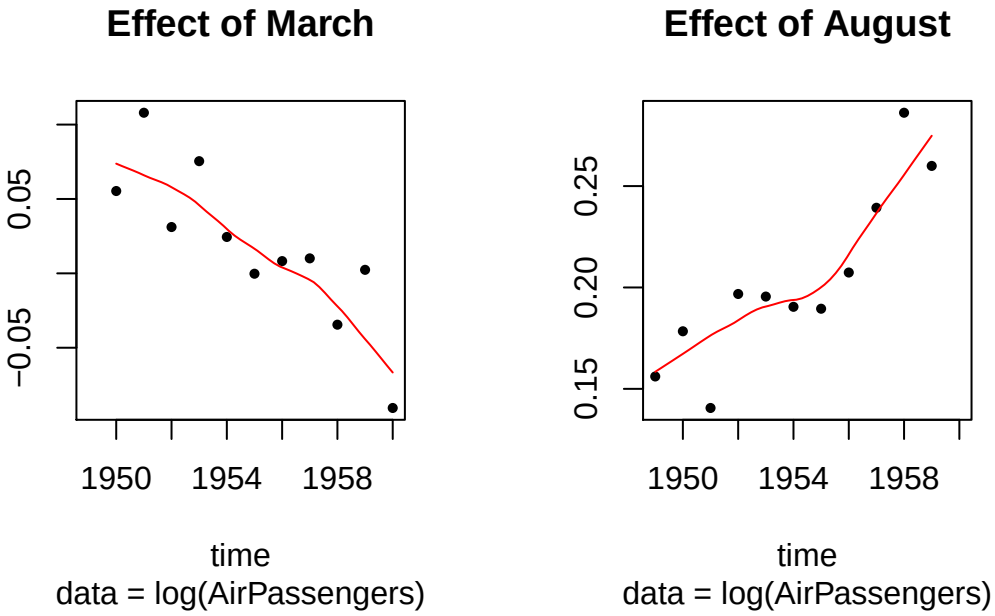
Wie erwähnt ist die Visualisierung fast deckungsgleich mit jener auf von `decompose()`. Zusätzlich wird rechts mit den grauen Balken der relative Anteil der Komponenten gezeigt. Je kleiner er ist, desto wichtiger ist eine Komponente in der Zerlegung, denn die Balken decken in jedem Frame dieselbe y-Spanne ab.

Die Funktion `stl()` hat diverse Argumente, von denen wir die meisten aber ausser Acht lassen können, bzw. deren Default-Einstellungen verwenden. Wichtig sind hingegen `t.window` und `s.window`. Ersteres, `t.window`, steuert die Fensterbreite bei der Trend-Glättung. Es hat einen Default-Wert, der zumeist gut funktioniert. Hin und wieder kommt es aber vor, dass man von Hand korrigieren muss. Der erste Schritt ist das Ablesen des Default-Werts. Dies geht so:

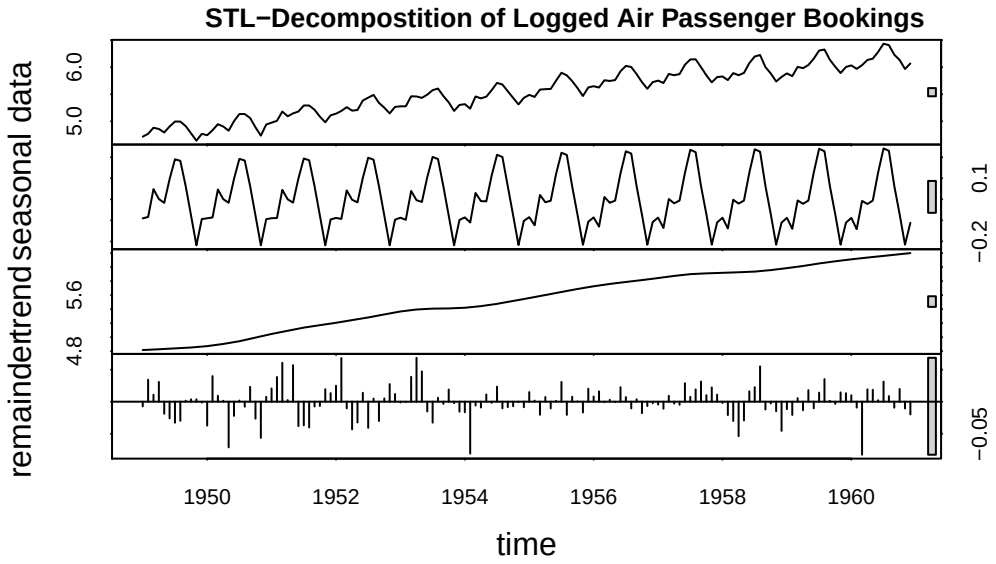
```
> co2.stl$win[2]

 t
19
```

Wenn wir nun z.B. `t.window=31` setzen, so wird der Trend stärker geglättet. Wählen wir hingegen einen Wert kleiner als 19, so erhalten wir einen raueren Trend, der dafür der Reihe besser folgt. Ziel ist es natürlich, den Wert so einzustellen, dass der Trend weder zu rau noch zu glatt ist.

Das zweite Argument, `s.window`, steuert die Glättung beim Schätzen des Saisoneffekts. Es hat *keinen* Default und muss in jedem Fall manuell gesetzt werden. Üblicherweise verwendet man `s.window='periodic'`. Der Saisoneffekt wird dann als über die Zeit gleich bleibend geschätzt. Es gibt jedoch auch Zeitreihen, wo sich der Saisoneffekt über die Zeit deutlich verändert. Die Air Passenger Bookings sind ein gutes Beispiel dafür. Dann sollte man eine adaptive Schätzung verwenden. Mit `stl()` kann man dies tun, indem man s.window auf einen numerischen Wert setzt, z.B. s.window=13. Doch was passiert dann?

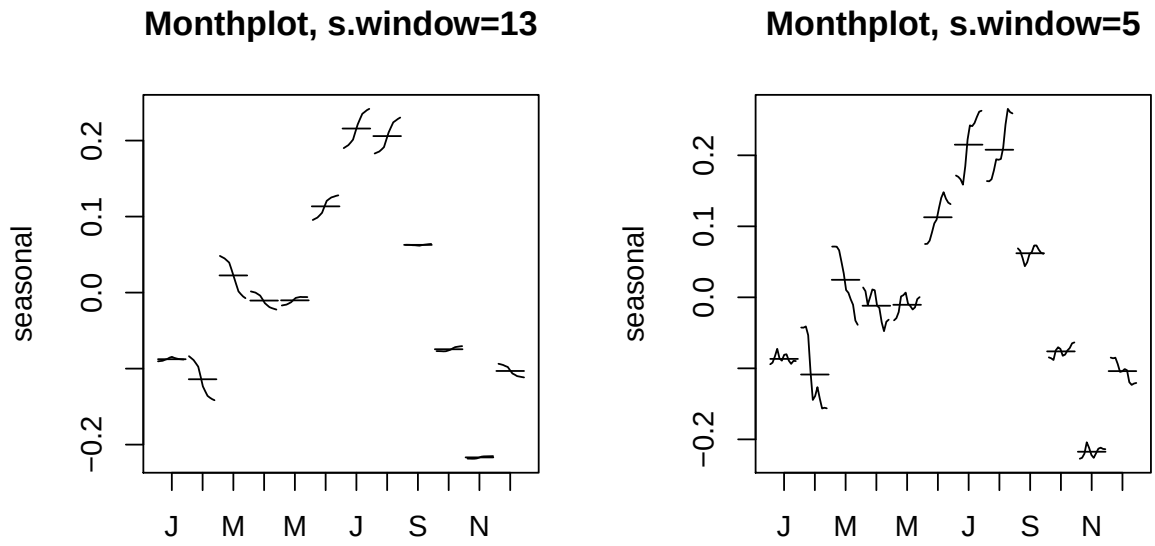


Für diese adaptive Schätzung geht man davon aus, dass die Veränderung des Saisoneffekts langsam und glatt verläuft. Die obigen Plots illustrieren dies für die Air Passenger Bookings: dargestellt sind die trendbereinigten März- und August-Effekte über die 11 Beobachtungsjahre hinweg. Man sieht, dass der Märzeffekt immer schwächer, der Augusteffekt dagegen immer stärker wird. Manifestiert wird dies durch den rot eingezeichneten Glätter.



Der Plot auf der vorangehenden Seite zeigt die Zerlegung der Air Passenger Bookings bei Verwendung eines adaptiven Saisoneffekts. Diese Schätzung ist als gelungen zu bewerten, `s.window=13` war also eine gute Wahl. Wie kommt man darauf? Ganz ohne zu experimentieren läuft das hier nicht ab, und dazu ist die Funktion `monthplot()` sehr nützlich. Sie zeigt, wie die Saisoneffekte geschätzt wurden.

```
> par(mfrow = c(1,2))
> monthplot(stl(log(AirPassengers), s.window = 13))
> title('Monthplot, s.window=13')
> monthplot(stl(log(AirPassengers), s.window = 5))
> title('Monthplot, s.window=5')
```



Der linke Plot, erzeugt mit dem Argument `s.window=13`, erfasst die Veränderungen beim Saisoneffekt (Zunahme im Sommer, Abnahme im Frühling), zeigt aber keine Anzeichen von Overfitting. Anders bei `s.window=5` wie im Plot rechts. Die geschätzten Änderungen des Saisoneffekts sind hier sehr rau und darum nicht vertrauenswürdig.

4.3 Autokorrelation

Wir kümmern uns nun um die Schätzung und Darstellung der stochastischen Abhängigkeit zwischen aufeinander folgenden Beobachtungen in Zeitreihen. Diese ist unter dem Namen Autokorrelation bekannt, und wie folgt definiert:

$$Cor(X_{t+k}, X_t) = \frac{Cov(X_{t+k}, X_t)}{\sqrt{Var(X_{t+k})Var(X_t)}} \tag{8}$$

Die Autokorrelation ist ein dimensionsloses Mass für den linearen Zusammenhang zwischen zwei Zufallsvariablen. Für stationäre Zeitreihen hängt diese nur vom Abstand zwischen den Beobachtungen, d.h. dem Lag k ab, nicht aber vom Zeitpunkt t. Wir können daher die Autokorrelationsfunktion oder ACF (engl. Abkürzung für auto correlation function) einführen:

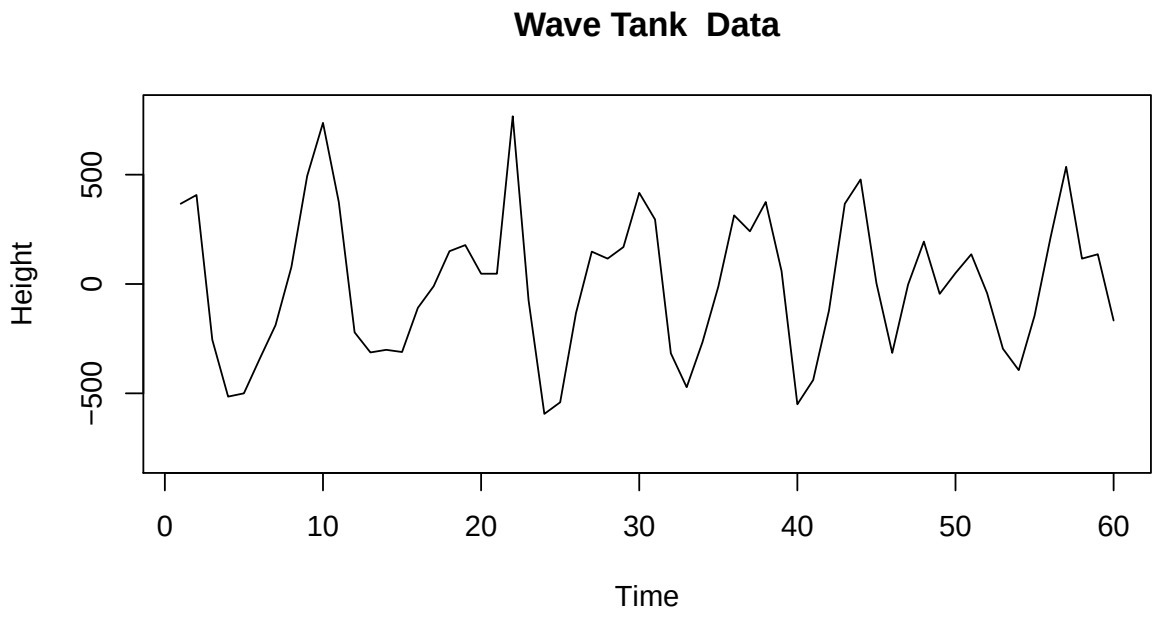
$$\rho(k) = Cor(X_{t+k}, X_t) \tag{9}$$

Im Folgenden kümmern wir uns um Schätzung und Bedeutung der ACF.

Zum Studium der Autokorrelation verwenden wir hier das Beispiel eines Wassertanks. Dessen Inhalt ist durch Wellengang in ständiger Bewegung. Gemessen wird die Wasserhöhe

an einem bestimmten Punkt, und zwar als Abweichung in Millimeter gegenüber dem Ruhezustand. Der Zeitabstand zwischen zwei Messungen beträgt 0.1 Sekunden, total liegen 396 Beobachtungen vor. Wir stellen hier die ersten 60 davon dar.

```
> wave <- ts( read.table(paste(data.dir,'wave.dat', sep = ''), header=TRUE)$waveht)
> plot(window(wave, 1, 60), ylim=c(-800,800), ylab='Height')
> title('Wave Tank  Data')
```

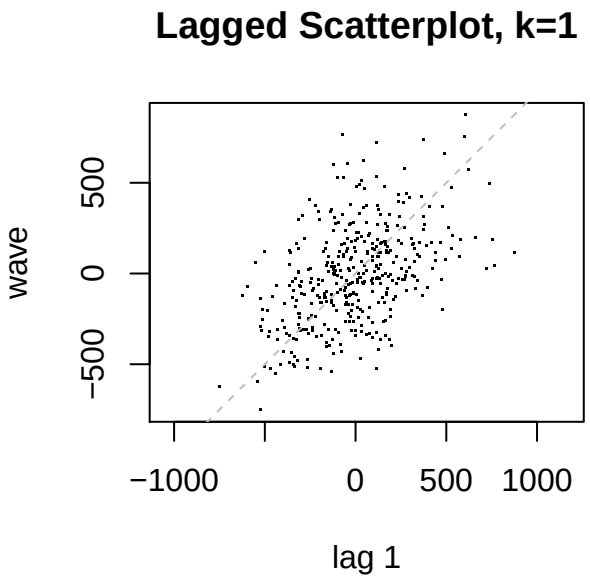


Die Daten zeigen deutlich ein zyklisches Verhalten. Wie wir aus eigener Erfahrung wissen, „kommen und gehen“ die Wellen. D.h. der Wasserstand zum aktuellen Zeitpunkt ist mit demjenigen zum Vorzeitpunkt verknüpft, oder eben mathematisch: korreliert.

4.3.1 Lagged Scatterplot

Um die Korrelation zwischen Beobachtungen mit Zeitabstand k zu untersuchen bietet sich ein Scatterplot der Datenpaare $(x_t, x_{t+k}$ für alle $t = 1, \ldots, n-k$ an. In einem ersten Schritt setzen wir $k = 1$. Während sich das auch selber einfach programmieren liesse, hält **R** die Funktion `lag.plot()` dafür bereit.

```
> lag.plot(wave, do.lines=FALSE, pch='.')
> title('Lagged Scatterplot, k=1')
```



Der Zusammenhang ist linear und positiv. Die Pearson-Korrelation beträgt 0.47, ist also mässig stark. Der Wert bedeutet übrigens, dass der durch x_t erklärte Anteil der Variabilität in x_{t+1} gerade $0.47^2 = 0.22$ ist, d.h. dass 22% der Variabilität im Nachfolge-Wert durch den Vorgänger erklärt werden.

Natürlich kann die Idee des Lagged-Scatterplot auf $k > 1$ erweitert werden, siehe die grafische Darstellung auf der nächsten Seite. Wiederum lässt sich die Pearson-Korrelation dieser Punktwolken berechnen, und zwar mit der hier angeführten Formel.

$$\widetilde{\rho}(k) = \frac{\sum_{s=1}^{n-k}(x_{s+k} - \overline{x}_{(k)})(x_s - \overline{x}_{(l)})}{\sqrt{\sum_{s=k+1}^{n}(x_s - \overline{x}_{(k)})^2 \sum_{t=1}^{n-k}(x_t - \overline{x}_{(l)})^2}} \tag{10}$$

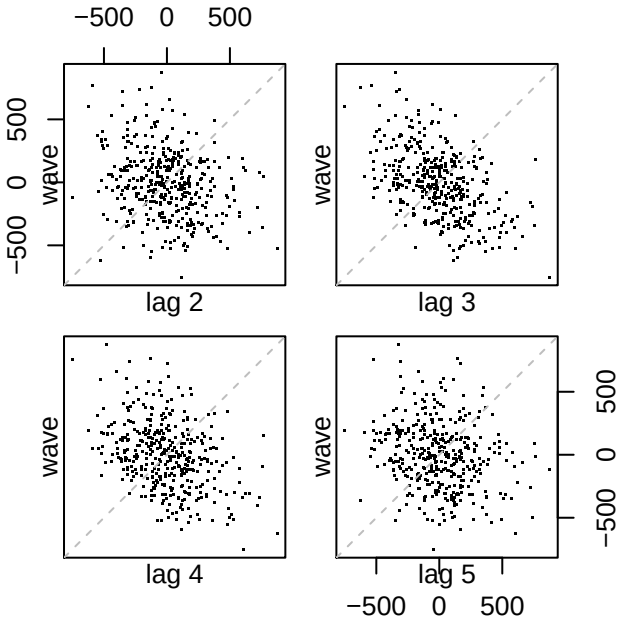
$$\text{wobei } \overline{x}_{(l)} = \frac{1}{n-k}\sum_{i=1}^{n-k} x_i \text{ und } \overline{x}_{(k)} = \frac{1}{n-k}\sum_{i=k+1}^{n} x_i$$

Für die Lags $k = 2, \ldots, 5$ ergeben sich als $-0.27, -0.50, -0.39$ und -0.22. Zu berücksichtigen ist, dass für die Berechnung von $\widetilde{\rho}(1)$ noch $n-1$ Datenpunkte zur Verfügung stehen, jedoch nur noch $n-2$ für $\widetilde{\rho}(2)$, usw.. Allgemein sind es für den Lag k nur noch $n-k$ Paare und je höher wir gehen, desto eher betrachten wir rein aus Zufall grosse Werte. Für $\widetilde{\rho}(n-2)$ stehen nur noch 2 Datenpunkte zur Verfügung, die man immer durch eine Gerade verbinden kann. Somit ergibt sich in jedem Fall $\widetilde{\rho}(n-2) = \pm 1$. Dabei handelt es sich aber um einen Schätzeffekt und nicht um eine tatsächlich existierende, perfekte Korrelation.

Aus den eben dargelegten Gründen, lässt sich mathematisch nachweisen, dass die Schätzung der Autokorrelation durch Berechnung der Pearson-Korrelation in den Lag-

ged Scatterplots suboptimal ist. Während sie als „Bild vor den Augen“ zur Interpretation nützlich ist, liefert eine relativ kleine Modifikation in der Berechnungsweise viel zuverlässigere Resultate.

```
> lag.plot(wave, lags = 4, set.lags = 2:5,do.lines=F, pch='.')
```



4.3.2 Plug-In-Schätzung

Um das oben angesprochene Problem bei der Schätzung der Autokorrelation zu entschärfen hat sich folgende Berechnungsvorschrift etabliert:

$$\widehat{\rho}(k) = \frac{\widehat{\gamma}(k)}{\widehat{\gamma}(0)}, \text{ für } k = 1, \ldots, n-1 \tag{11}$$

$$\text{wobei } \widehat{\gamma}(k) = \frac{1}{n}\sum_{s=1}^{n-k}(x_{s+k} - \bar{x})(x_s - \bar{x}), \text{mit } \bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t$$

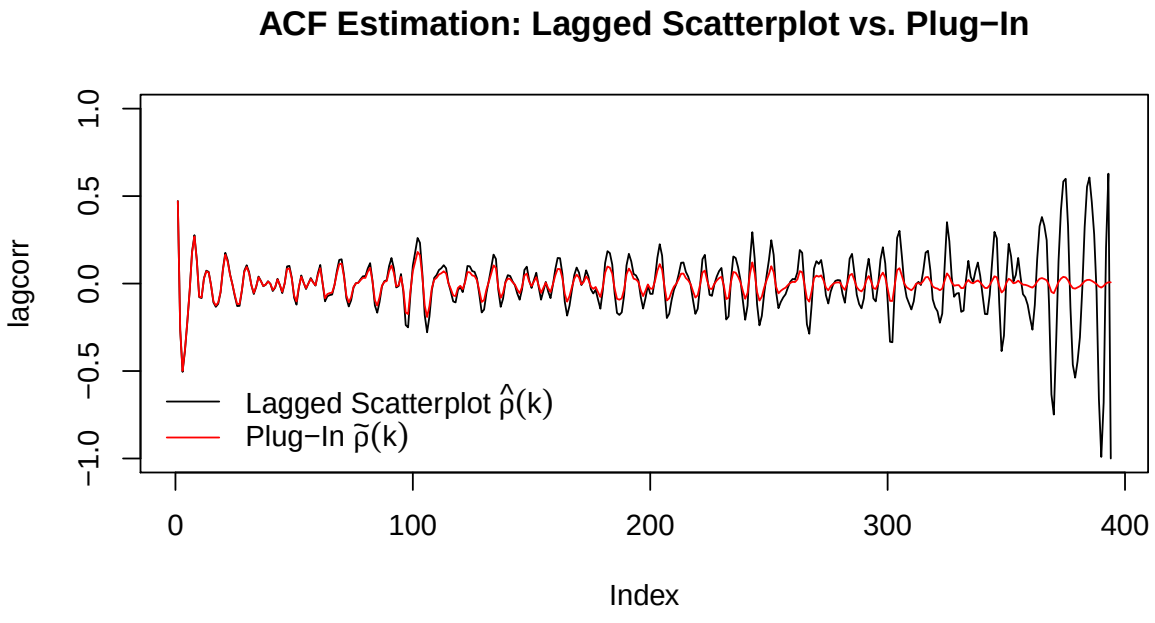
Sämtliche Terme sind $\widehat{\gamma}(k)$ Summen über $n-k$ Terme. Dennoch wird beim Mitteln aber durch n geteilt. Dies mag auf den ersten Blick unlogisch erscheinen, führt jedoch genau dazu, dass die geschätzten Autokorrelationen für grosse Lags gedämpft werden. Dieser Mechanismus ist sinnvoll, weil die Autokorrelation durch die wenigen Datenpunkte sonst oft zu gross angegeben wird. Die „richtige“ Schätzung der ACF ist also definiert durch:

$$\widehat{\rho}(k) = \frac{\sum_{s=1}^{n-k}(x_{s+k} - \bar{x})(x_s - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2} k = 1, \ldots, n-1. \tag{12}$$

Diese Formel ist in **R** bereits implementiert, und zwar in der Funktion `acf()`. Per Default werden die Resultate grafisch dargestellt. .Wenn wir an den Zahlenwerten interessiert sind, so müssen wir das Argument `plot=FALSE` setzen. Man beachte auch, dass **R** die Autkorrelationen nur für die Lags $k = 1, \ldots, 25$ ausgibt.

```
> acf(wave, plot=FALSE)


Autocorrelations of series 'wave', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.470 -0.263 -0.499 -0.379 -0.215 -0.038  0.178  0.269  0.130 -0.074
    11     12     13     14     15     16     17     18     19     20     21
-0.079  0.029  0.070  0.063 -0.010 -0.102 -0.125 -0.109 -0.048  0.077  0.165
    22     23     24     25
 0.124  0.049 -0.005 -0.066
```

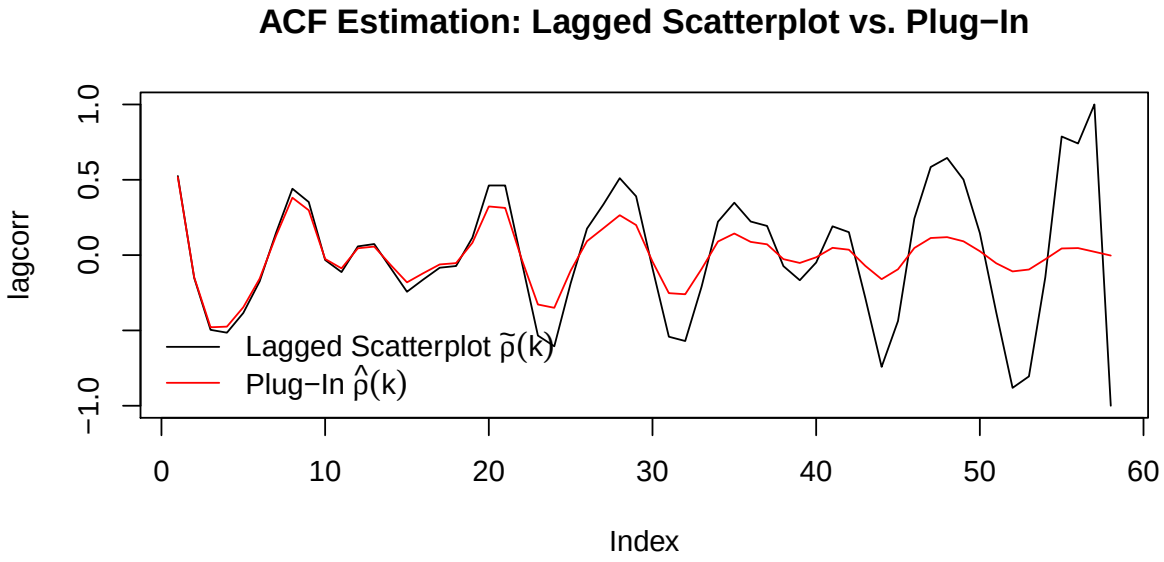
Für die ersten paar Lags stimmen die $\widehat{\rho}(k)$ beinahe mit den $\widetilde{\rho}(k)$ überein, was auch nicht weiter erstaunlich ist, wenn man die Formeln studiert. Die Unterschiede treten erst für die höheren Lags zu Tage. Wir führen darum einen grafischen Vergleich der beiden Berechnungsweisen durch:



Während die Unterschiede auf den ersten 50 Lags klein sind, werden sie danach immer grösser. Die in Rot dargestellten $\widehat{\rho}(k)$ sind gedämpft und bleiben betragsmässig klein, die in Schwarz dargestellten $\widetilde{\rho}(k)$ oszillieren und nehmen grosse Werte an. Natürlich kann man

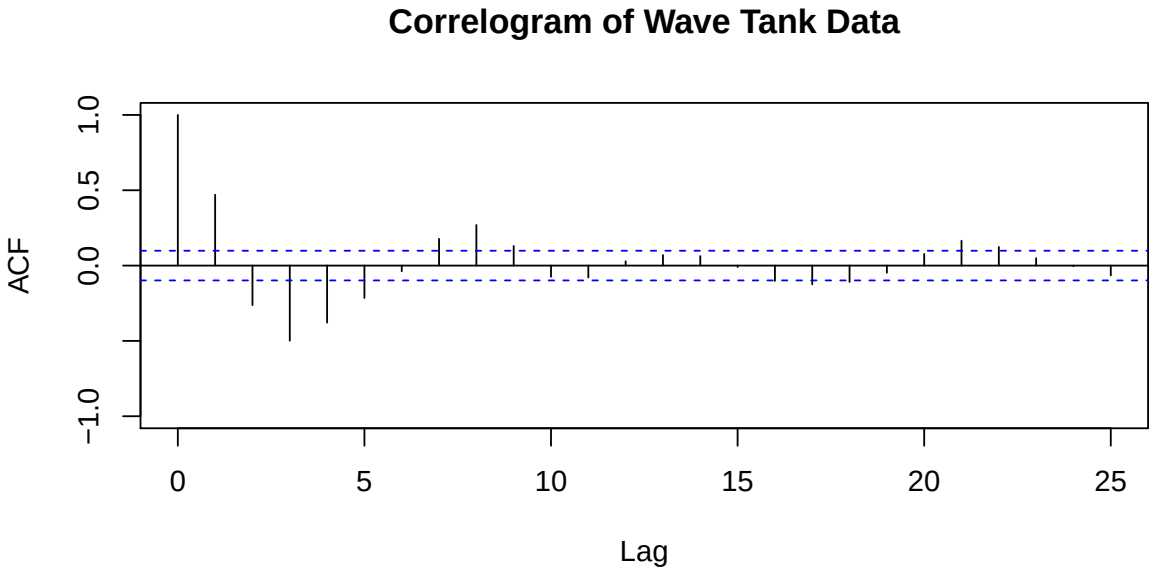
nun die Frage stellen, welche der beiden Methoden genauer ist. Dafür kann man entweder die Mathematik bemühen, oder es gibt ein Argument, das auch den Praktiker überzeugt.

Wir beschränken die Zeitreihe willkürlich auf nur noch 60 Beobachtungen. Weil sie stationär ist, soll sich die Abhängigkeit der Beobachtungen nicht ändern. Daher stellen wir wiederum die Schätzungen für $\widehat{\rho}(k)$ und $\widetilde{\rho}(k)$ dar. Weil die Reihe nun kürzer geworden ist, ist dies nur noch für die Lags $k = 1, \ldots, 59$ möglich. Und wir haben nun den empirischen Beweis, dass die Schätzmethode $\widetilde{\rho}(k)$ versagt: bei den hohen Lags zeigen sich wiederum erratische Schätzungen nahe bei $\pm + 1$.



4.3.3 Das Korrelogramm

Wir haben bisher bereits gelernt, warum und wie bei stationären Zeitreihen die Autokorrelationen geschätzt werden können. Wir besprechen hier nun auch noch die Visualisierung der Resultate, und noch wichtiger, deren Interpretation. Wenn wir die Funktion `acf()` im Default-Modus ausführen, so ergibt sich stets eine grafische Darstellung mit einem Stabdiagramm:



Die ACF beginnt mit Lag 0. Dies ist ein Anachronismus, denn es handelt sich ja um die Korrelation einer jeden Beobachtung mit sich selbst, und diese ist stets 1. Damit die Stärke der Autokorrelation besser eingeschätzt werden kann, empfiehlt es sich auch, die y-Achse mit dem Intervall $[-1, 1]$ darzustellen. Doch nun zur Interpretation: für die Lags 3 und 4 haben wir negative Korrelation. Wenn wir uns im Wellental befinden, entspricht sie gerade dem nächsten Wellenberg. Bei den Lags 7 und 8 ist die Korrelation wieder positiv, hier kommt das nächste Wellental.

Konfidenzbänder

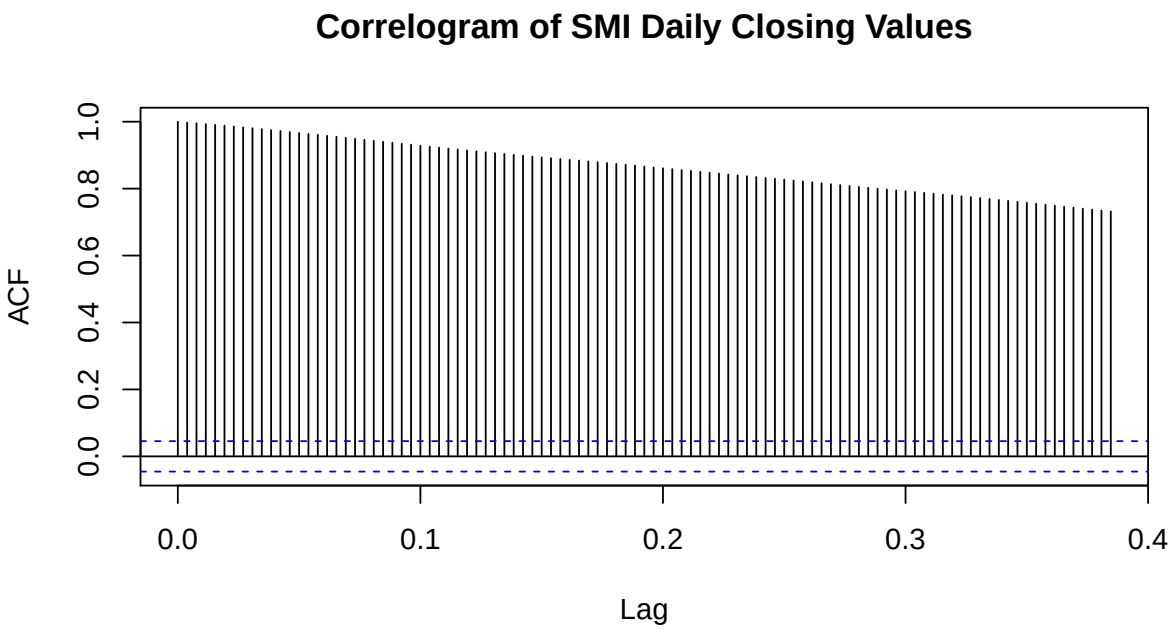
Im obigen Korrelogramm fallen die blau punktierten Linien auf, dies sind die sogenannten Konfidenzbänder. Sie sind da, weil selbst im Fall von unabhängigen (d.h. nicht korrelierten, $\rho(k) = 0$ für alle k) Beobachtungen die Schätzungen typischerweise von null verschieden sind, d.h. $\widehat{\rho}(k) \neg 0$. Falls die Schätzungen qualitativ gut sind, so sollten die Ergebnisse natürlich nahe bei null liegen, aber exakt dort werden sie nie sein. Natürlich stellt sich die Frage, was denn „nahe bei null“ heisst. Sie wird durch die blau punktierten Konfidenzbänder beantwortet:

Für jede stationäre Zeitreihe gehen wir davon aus, dass sämtliche geschätzten Autokorrelationen $\widehat{\rho}(k)$, die innerhalb der durch $\pm 1.96/\sqrt{(n)}$ definierten Konfidenzbänder liegen, nur durch Zufall von null verschieden sind. Somit besteht nur bei jenen Lags tatsächlich eine Korrelation, wo die Schätzung das Konfidenzband übertrifft.

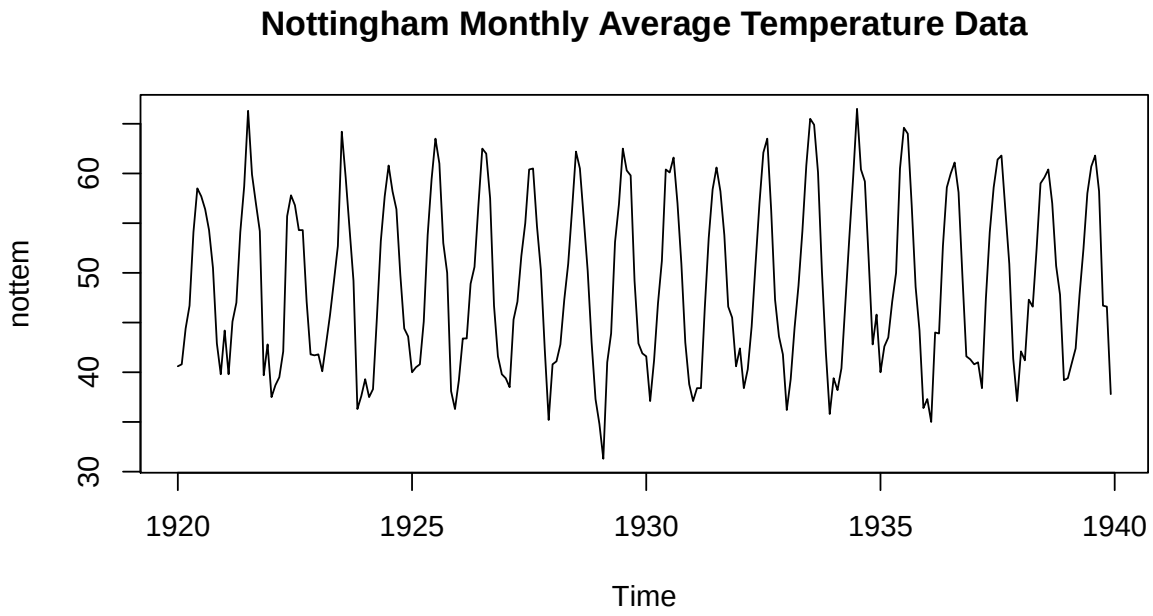
Nun gilt es noch zu beachten, dass diese Beurteilung einem statistischen Hypothesentest entspricht. Und hier besteht immer eine Irrtumswahrscheinlichkeit, im konkreten Fall von 5%. Selbst wenn das Konfidenzband übertroffen wird, hat man also keine Gewähr, dass

tatsächlich eine Abhängigkeit besteht. textbfEigenschaften der ACF von nicht stationären Zeitreihen Damit man die ACF überhaupt aus den Datenpaaren schätzen kann, muss man davon ausgehen, dass die Abhängigkeitsstruktur über die Zeit erhalten bleibt. Oder mit anderen Worten: die Schätzung macht nur für stationäre Reihen Sinn. *Berechnen*, im Sinne einer Durchführung des Algorithmus, kann man die ACF aber selbstverständlich für alle, auch nicht stationäre Reihen. Dies ist zentral, weil nicht stationäre Reihen oftmals sehr charakteristische Korrelogramme zeigen, welche zur Entscheidung stationär *ja/nein* herangezogen werden können.

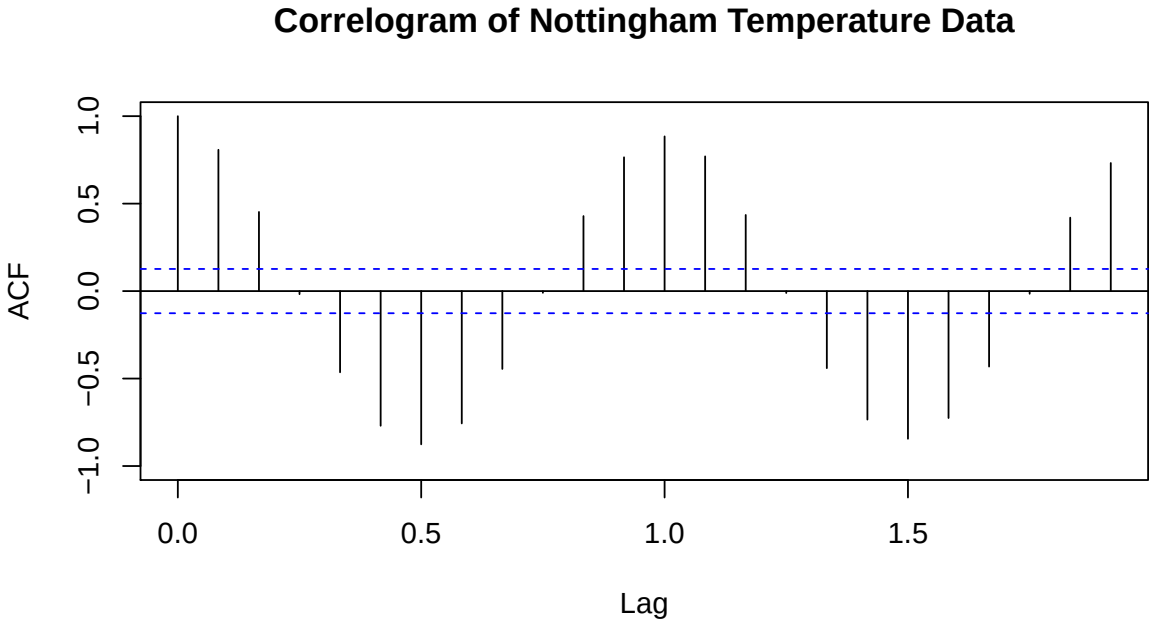
Als Beispiel betrachten wir die Reihe der SMI-Schlusskurse aus Kapitel 1.2.4. Weil ein Trend vorliegt, ist diese ganz eindeutig nicht stationär. Einen Saisoneffekt scheint es hingegen nicht zu geben. Im Plot auf der nächsten Seite zeigt sich, dass die ACF nur sehr langsam abfällt. Das ist sehr typisch für nicht stationäre Reihen die einen Trend enthalten. Es ist darin begründet, dass aufeinander folgende Beobachtungen typischerweise auf derselben Seite des globalen Mittels $\bar{x}$ liegen. Dann sind für kleine bis mittlere Lags k fast alle Terme $(x_{s+k} - \bar{x})(x_s - \bar{x})$ positiv. Daher kommt ein positives Gesamtresultat zustande, und je stärker der Trend ist, desto näher liegen die Resultate beim Wert 1.



Als nächstes betrachten wir eine Zeitreihe mit starkem Saisoneffekt, aber ohne Trend. Ein Beispiel sind die in **R** verfügbaren, monatlichen Durchschnittstemperaturen aus Nottingham Castle in England aus dem Zeitraum 1920-1939.

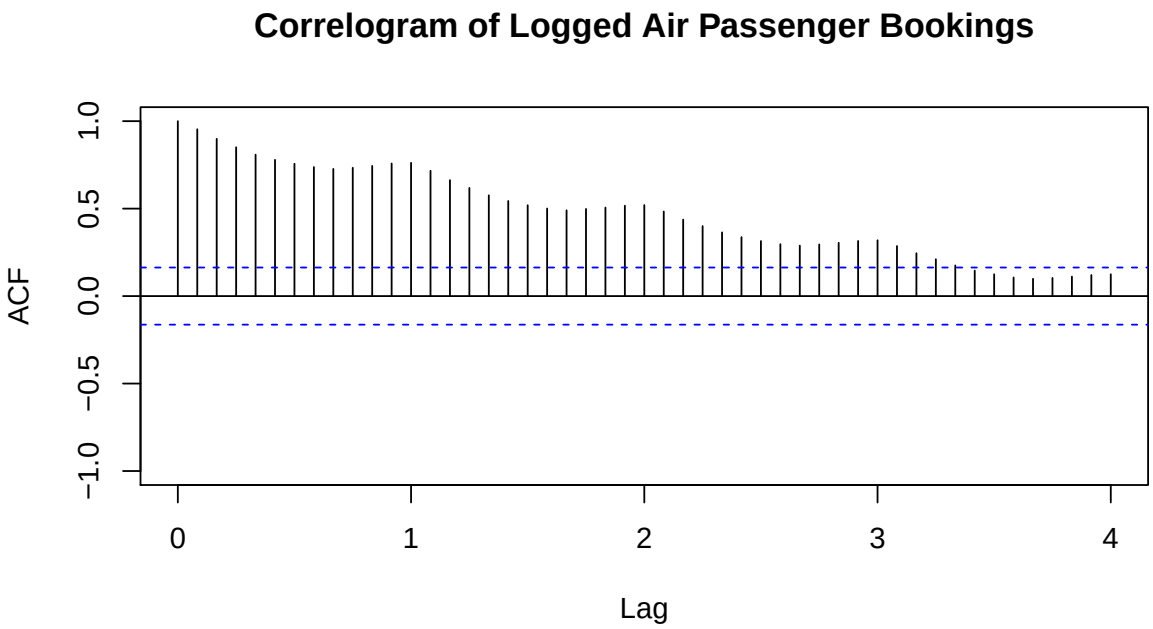


Die ACF (siehe nächste Seite) zeigt nun ebenfalls ein zyklisches Verhalten. Zum Lag 1, d.h. zum Folgemonat, ist die Korrelation positiv, weil die Temperatur da in der Regel ähnlich ist. Ebenso ist in demselben Monat des Folgejahres eine ähnliche Temperatur zu erwarten, und daher die Korrelation zum Lag 12 stark positiv. Andersrum hingegen beim Lag 6, hier haben wir negative Korrelation.



Ein starkes zyklisches Verhalten im Korrelogramm kann auf einen Saisoneffekt hin-

weisen. Schliesslich ist es so, dass sich die Auswirkung von Trend und Saison überlagern können. Dies wird durch das Korrelogramm der Air Passenger Bookings aufgezeigt.

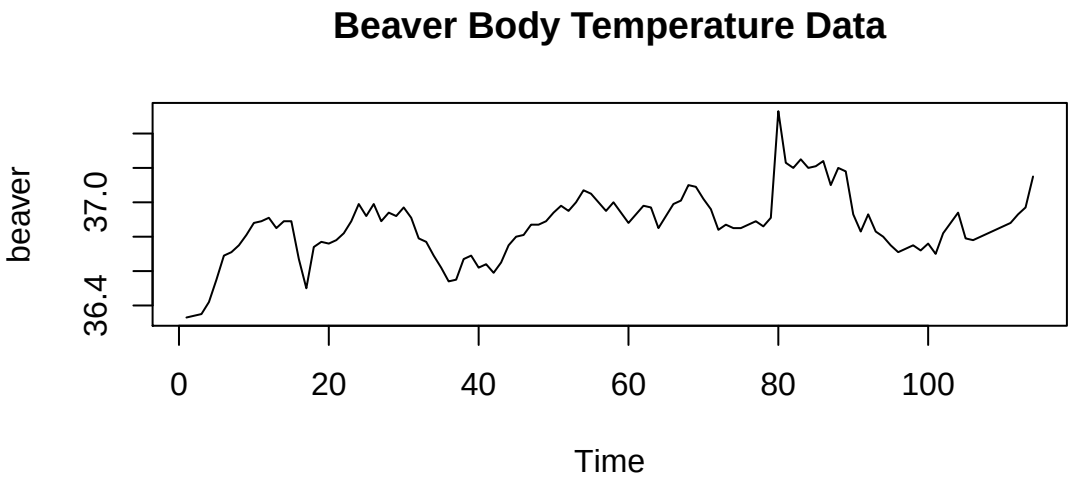


Die Folgerung aus einem derartigen Korrelogramm ist, dass die Reihe zuerst in ihre Komponenten (Trend, Saison und Rest) zerlegt werden muss.

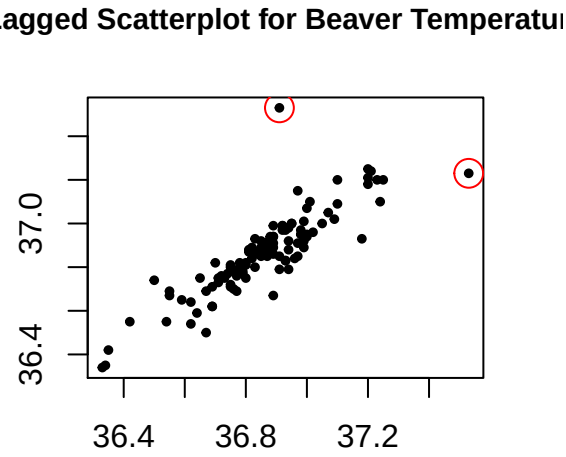
Auswirkung von Ausreissern in der ACF

Geht es um statistische Analysen, sind Ausreisser nie willkommen. Im Korrelogramm verstärkt sich das Problem sogar, weil sie, im Sinne der Bildung von Punktepaaren, sogar einen doppelten Einfluss haben können. Wir studieren dies mit der Zeitreihe `temp` aus dem Data Frame `beaver1`, welches im **R**-Datensatz `data(beavers)` vorhanden ist. Gemessen wurde dabei die Körpertemperatur eines Biber-Weibchens in 10-Minuten-Intervallen.

```
> data(beavers)
> beaver <- ts(beaver1$temp, start=1, freq=1)
> plot(beaver, main='Beaver Body Temperature Data')
```

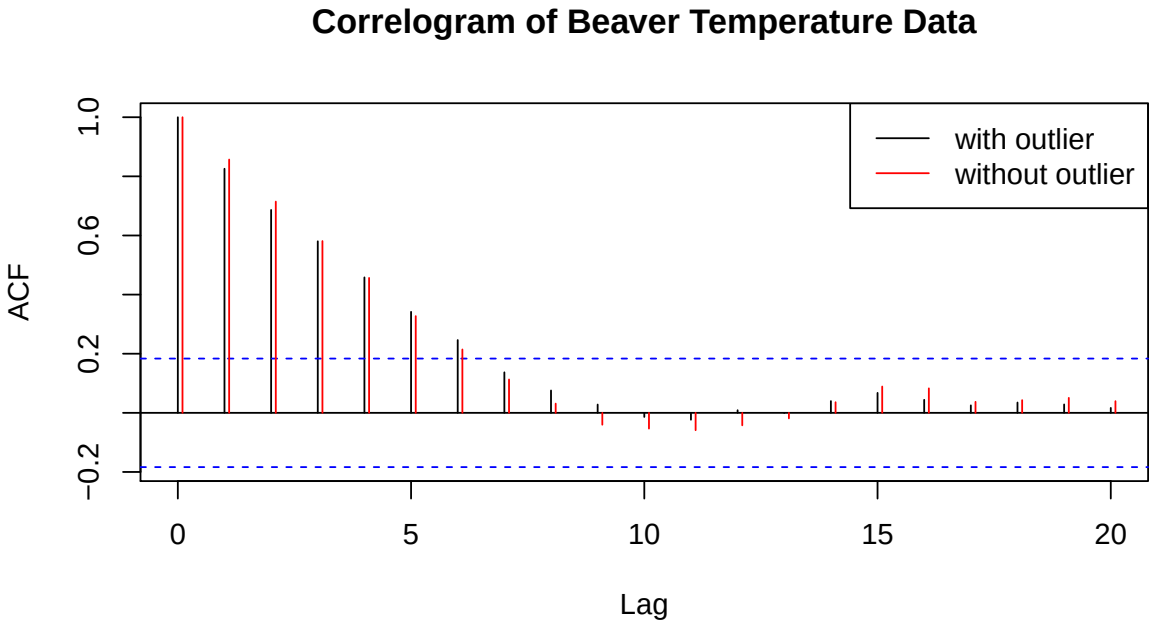


Der Datenpunkt 80 ist ein (moderater) Ausreisser. Ob es sich um einen Datenfehler oder um einen korrekten Wert handelt, ist dem Autor unbekannt. Dies ist jedoch auch nicht so wichtig, weil es hier vor allem um die potentielle Schädlichkeit geht. Man bedenke, dass die ACF-Schätzungen nicht robust sind, d.h. bereits durch einen einzelnen Ausreisser verfälscht werden können. Für eine Visualisierung stellen wir einen Lagged Scatterplot dar (siehe nächste Seite).



Die beiden Paare, wo der Ausreisser beteiligt ist, sind klar identifizierbar (rot eingekreiste Pukte). Hier haben wir ein Stück weit das Glück, dass sie sich gegenseitig neutralisieren, was aber längst nicht immer der Fall sein muss. So ergibt sich die Pearson-Korrelation zu 0.91 ohne die Ausreisser, und zu 0.86 mit. Ich denke es erstaunt nicht, dass der Unterschied auch viel grösser ausfallen kann.

Wir betrachten hier auch noch das gesamte Korrelogramm, berechnet mit (in Schwarz) und ohne (in Rot) Ausreisser. Die Unterschiede sind hier relativ klein und mögen bloss als von akademischem Interesse erscheinen. Man sei sich aber in der Zeitreihenanalyse stets bewusst, dass durch Ausreisser grosse Verfälschungen eintreten können.



Natürlich fragt sich auch, was man denn tun soll, falls es tatsächlich einen oder mehrere Ausreisser gibt. Beobachtungen einfach wegzulassen ist gar nicht so einfach, weil so das Paradigma der äquidistanten Beobachtungen verletzt wird. Somit werden fehlende/gelöschte Beobachtungen oft ersetzt. Und zwar mit:

1. dem globalen Mittelwert
2. einem lokalen Mittelwert, z.B. ± 3 Beobachtungen
3. einer modellbasierten Imputation mit einem Vorhersagemodell

Welche Strategie die beste ist, hängt von der Zeitreihe ab und kann kaum allgemein beantwortet werden. Und **R** sei Dank existiert auch eine vierte Alternative: während die Funktion `acf()` per Default fehlende Werte nicht zulässt, kann das Argument na.action=na.pass, gesetzt werden. Die Autokorrelation wird dann aus den kompletten Datenpaaren geschätzt.

4.4 Partielle Autokorrelation

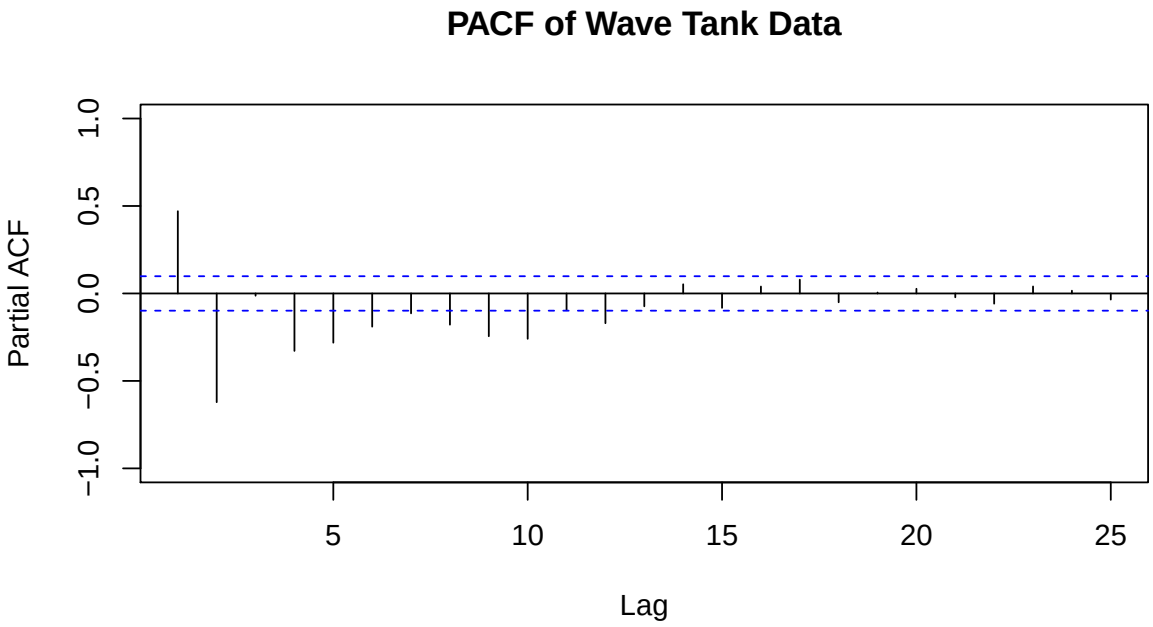
Die gewöhnlichen Autokorrelationen aus Kapitel 4.3 stellen die Abhängigkeit von zwei Beobachtungen mit Zeitabstand k dar und lassen dabei ausser Acht, was zwischen den beiden Beobachtungen passiert. Diese Information ist durchaus nützlich, es hat sich jedoch gezeigt, noch tiefer zu gehen. Die partielle Autokorrelation von zwei Beobachtungen im

Lag k untersucht also, wie der Zusammenhang ist, wenn man die Zwischenwerte kennt. Mathematisch handelt es sich um eine bedingte Korrelation:

$$\pi(k) = Cor(X_{t+k}, X_t | X_{t+1} = x_{t+1}, \ldots, X_{t+k+-1} = x_{t+k-1}) \tag{13}$$

Am einfachsten verständlich ist es, wenn man eine Analogie zur linearen Regression zieht: die gewöhnliche Autokorrelation $\rho(k)$ entspricht dem Resultat einer einfachen Regression, wo X_t die Zielgrösse und X_{t-k} der Prädiktor ist. Die partielle Autokorrelation $\pi(k)$ erhält man hingegen als Koeffizient von X_{t-k}, wenn man eine multiple Regression der Zielgrösse X_t auf die Prädiktoren $X_{t-1}, \ldots, X_{t-k}$ rechnet. Das ist ein wesentlicher Unterschied, wie wir bereits früher gelernt haben! Die Schätzung in **R** findet tatsächlich mit einem Regressionsansatz statt. Wir verzichten hier jedoch auf die Details, und weisen nur darauf hin, dass die Funktion `pacf()` dies implementiert.

```
> pacf(wave, ylim=c(-1,1),main='PACF of Wave Tank Data')
```



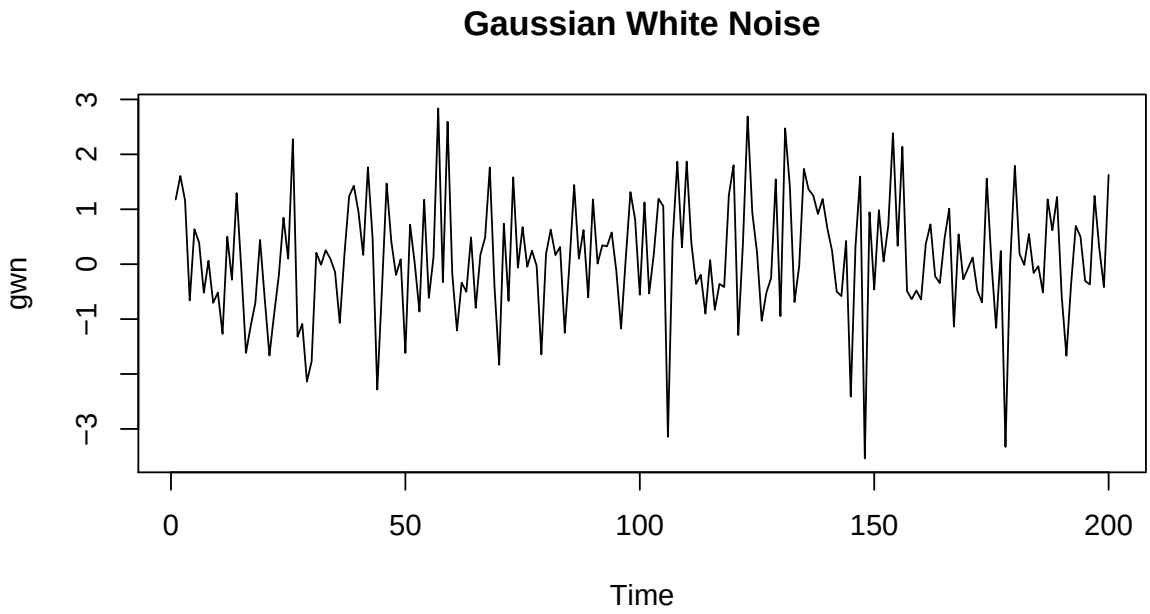
5 Modelle für stationäre Zeitreihen

Während wir bisher die Visualisierung von Zeitreihen und ihre Eigenschaften studierten, geht es hier um das Anpassen von Modellen. Diese sollen in erster Linie die Abhängigkeiten in stationären Zeitreihen aufzeigen.

5.1 WeissesRauschen

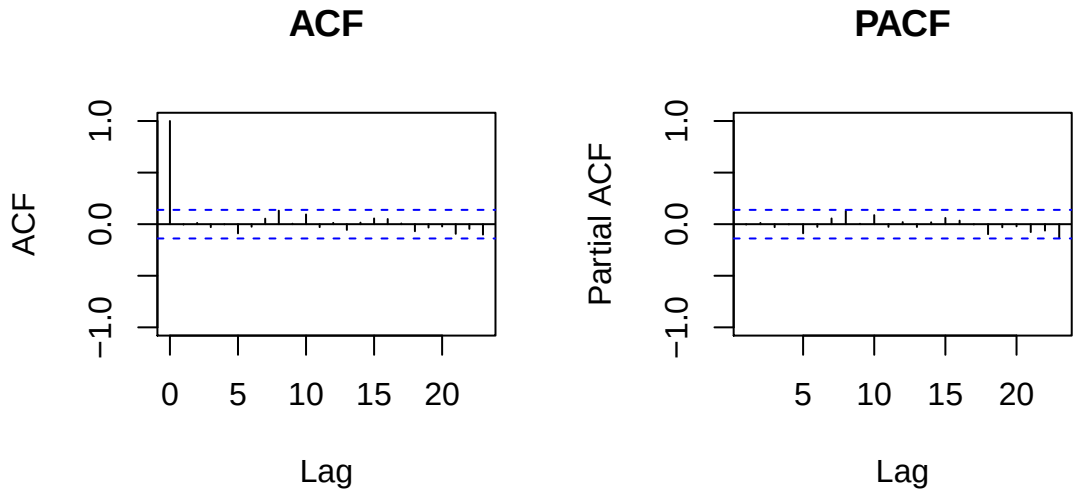
Ein einfaches und grundlegendes Zeitreihen-Modell ist das Weisse Rauschen. Eine Reihe $(W_1, W_2, \ldots, W_n)$ wird so benannt, falls die Zufallsvariablen $W_1, W_2, \ldots$ stochastisch unabhängig sind, und alle dieselbe Wahrscheinlichkeitsverteilung haben. Dies impliziert, dass die W_t gleichen Erwartungswert, gleiche Varianz und keine Autokorrelation haben: $\rho(k) = 0$ und $\pi(k) = 0$ für alle Lags k. Falls sie zusätzlich einer Normalverteilung folgen, d.h. $W_t \sim N(0, \sigma_W^2)$, so spricht man von Gauss'schem Weissem Rauschen. Der Begriff wurde übrigens durch eine Analogie zur Physik geprägt. In **R**, lässt sich Gauss'sches Weisses Rauschen sehr einfach generieren:

```
> gwn <- ts(rnorm(200, mean=0, sd=1))
> plot(gwn, main = "Gaussian White Noise")
```



Die obige Reihe ist 200 Beobachtungen lang und stationär, überdies sollte keine Abhängigkeit bestehen. Falls dem so ist, so sollten auch die geschätzten Autokorrelationen nahe bei null, sprich innerhalb der Konfidenzbänder liegen. Ein kurzer Check (siehe nächste Seite) belegt

dies. Man kann sich nun fragen, was das Studium von weissem Rauschen bringt. Die Antwort ist „viel“, denn solche Reihen und ihre Eigenschaften spielen beim Modellieren von komplexeren Reihen mit Abhängigkeiten eine wesentliche Rolle.



5.2 Autoregressive Modelle

5.2.1 Definition und Eigenschaften

Wie wir gelernt haben, zeigen auch stationäre Zeitreihen Abhängigkeiten. Das heisst, der Wert X_t ist mit den zeitlich davor liegenden Instanzen verknüpft. Die einfachste Art, eine solche Abhängigkeit zu modellieren ist mit einem linearen Modell auf die letzten p Beobachtungen:

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + \cdots + \alpha_p X_{t-p} + E_t \tag{14}$$

Es handelt sich um eine Regression der Reihe auf sich selbst, daher spricht man von einem *autoregressiven Modell der Ordnung p* , oder in Kurzform, $AR(p)$. Der Fehlerterm E_t soll die Eigenschaften von weissem Rauschen zeigen, d.h. in sich selbst unabhängig sein. Er ist das stochastische Element in der Modellgleichung, man nennt ihn auch *Innovation*, weil er der Reihe eine komplett neue Richtung geben kann. Die wichtigsten Fragen, die uns beschäftigen werden sind natürlich die Wahl von p sowie die Schätzung der Koeffizienten $\widehat{\alpha}_1, \ldots, \widehat{\alpha}_p$. Zuerst aber noch der entscheidende Punkt:

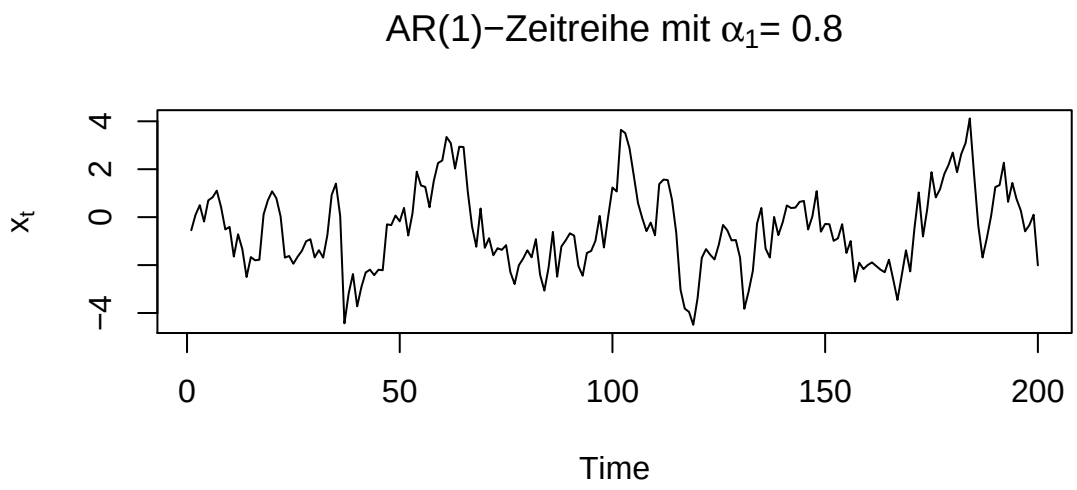
- *Die Anpassung von $AR(p)$-Modellen soll nur an stationäre Zeitreihen erfolgen, insbesondere müssen Trend und Saisoneffekt vorgängig entfernt worden sein.*

Wir betrachten nun als erstes einige einfache, aber dafür umso instruktivere Beispiele. Wir beginnen mit einem $AR(1)$-Modell mit $\alpha_1 = 0.8$. Die Modellgleichung lautet also:

$$X_t = 0.8 \cdot X_{t-1} + E_t$$

Eine solche Reihe können wir simulieren, indem wir für E_t Gauss'sches weisses Rauschen verwenden, und als Startwert $x_1 = E_1$ setzen.

```
> set.seed(24)
> E <- rnorm(200, 0, 1)
> x <- numeric()
> x[1] <- E[1]
> for (i in 2:200) x[i] <- 0.8*x[i-1] + E[i]
> plot(ts(x), ylab = expression(x[t]),
+      main =expression(paste("AR(1)-Zeitreihe mit ", alpha[1], "= 0.8", sep ="")))
```

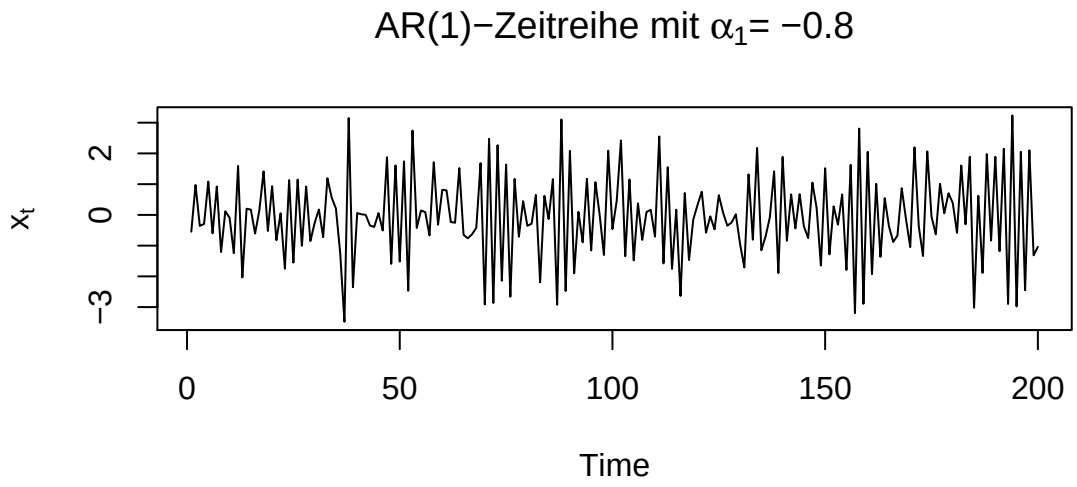


Wir beobachten, dass in dieser Reihe einige Zyklen mit ausschliesslich positiven Werten, und andere mit nur negativen Werten vorkommen. Dies ist nicht weiter erstaunlich: nimmt die Zeitreihe einmal einen betragsmässig grossen Wert an, so bestimmt sich die folgende Beobachtung als 0.8-mal diesen Wert, plus die Innovation. Befindet man sich auf einer Seite von null, so hat der nächste Wert eher wieder dasselbe Vorzeichen als nicht. Der Innovationsterm ist aber so stark, dass er dies *überspielen* kann, d.h. ein Wechsel zum anderen Vorzeichen ist dennoch jederzeit möglich. Dies gilt jedoch nur für $AR(1)$-Reihen, wo $|\alpha_1| < 1$, ansonsten ist die Innovation zu schwach, und die Zeitreihe kann in Richtung $+\infty$ oder $-\infty$ abdriften. In diesen Fällen ist sie dann auch nicht mehr stationär. Für die Autokorrelation der obigen Reihe gilt:

$$\rho(k) = 0.8^k \tag{15}$$

Benachbarte Instanzen weisen eine Korrelation von 0.8 auf. Ebenso sind alle weiteren Autokorrelationen positiv, sie fallen jedoch exponentiell schnell gegen null ab. Ein negatives Vorzeichen für den Koeffizienten α_1 ist auch erlaubt. Dies führt zu einer Reihe mit alternierendem Charakter: befinden wir uns auf einer Seite von null, so liegt die nächste

Beobachtung eher auf der anderen Seite als auf derselben. Der Plot mit einer Simulation der Länge 200 zeigt dies eindeutig. Und natürlich ist auch diese Reihe stationär.



Während das Simulieren von Daten aufgrund von Modellen stets instruktive Einblicke liefert, hilft es für die praktische Anwendung wenig. Wir besprechen im nächsten Abschnitt, wie man $AR(p)$-Modelle an beobachtete Zeitreihen anpasst.

5.2.2 Anpassung an Daten

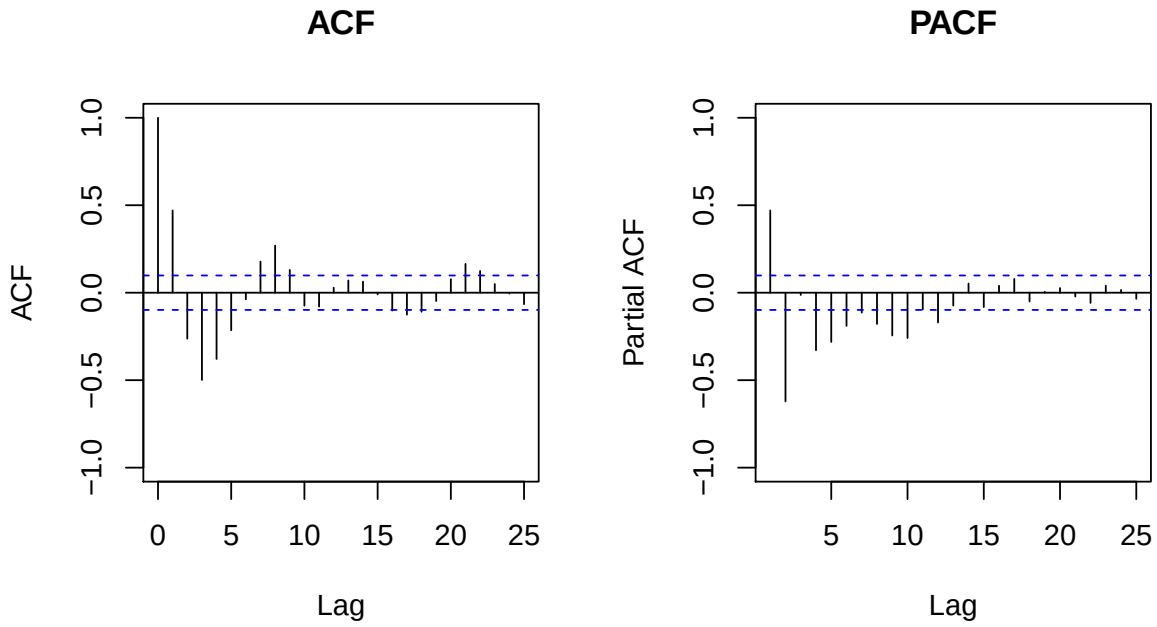
Die Anpassung eines $AR(p)$-Modelles umfasst mehrere Schritte. Wir müssen uns nämlich einerseits für eine Ordnung p entscheiden, dann müssen die Koeffizienten $\alpha_1, \ldots, \alpha_p$ geschätzt werden, und es gilt sicherzustellen, dass das Modell auch gut passt.

Wahl der Ordnung p

Hierfür bestehen im Prinzip zwei Möglichkeiten: der Experte liest sie aus den Korrelogrammen von ACF und PACF ab, was den Vorteil hat, dass man gleichzeitig noch weitere Eigenschaften der Reihe zu Gesicht bekommt und so schon eine Vorstellung entwickeln kann, wie gut ein $AR(p)$ zur Reihe passen wird. Für weniger versierte Anwender besteht in **R** die Option, die optimale Ordnung numerisch mit dem AIC-Kriterium zu bestimmen. In manchen Fällen kommt man mit beiden Methoden zu demselben Resultat, aber natürlich längst nicht immer.

Die Modellwahl mit ACF und PACF beruht auf der Tatsache, dass $AR(p)$-Zeitreihen in den Korrelogrammen ein sehr typisches Bild zeigen. Die ACF fällt exponentiell schnell ab, bzw. ist eine Schwingung mit exponentiell gedämpfter Amplitude. Die PACF kann bis zum Lag p ein beliebiges Verhalten zeigen, ist aber danach gleich null. Wir illustrieren dies mit den Wellenhöhen-Daten aus Kapitel 4.3, die beiden Plots befinden sich auf der nächsten Seite. Die ACF zeigt tatsächlich eine exponentiell schnell abklingende Schwingung, die man als musterhaft für eine $AR(p)$-Zeitreihe bezeichnen könnte. In der PACF ist das Bild nicht

ganz so schön: $\pi(1)$ und $\pi(2)$ sind beide betragsmässig gross, alle folgenden Koeffizienten deutlich kleiner. Der letzte signifikante Koeffizient ist hingegen beim Lag 12 zu finden. Dies spricht dafür, entweder $p = 2$ oder $p = 12$ zu verwenden. Als Grundregel gilt, immer zuerst das einfachere, sprich das kleinere Modell zu versuchen. Die Resultate und eine Modellwahl via AIC werden weiter unten gezeigt.



Schätzung der Koeffizienten

Ist die Ordnung p festgelegt, so ist die nächste Aufgabe, die Koeffizienten $\alpha_1, \ldots, \alpha_p$ zu schätzen. Da es sich bei einem $AR(p)$ im Prinzip um ein lineares Regressionsproblem handelt, könnten wir den Kleinste-Quadrate-Algorithmus verwenden. Aus mathematisch motivierten Gründen hat es sich aber etabliert, nicht einfach die Summe der Fehlerquadrate zu minimieren, sondern diesen Term:

$$\sum_{t=p+1}^{n} \left\{ \left(X_t - \sum_{k=1}^{p} \alpha_k X_{t-k} \right)^2 + \left(X_{t-p} - \sum_{k=1}^{p} \alpha_k X_{t-p+k} \right)^2 \right\} \tag{16}$$

Es handelt sich um eine Doppelsumme mit Fehlerquadraten. Hierfür gibt es keine explizite Lösung, sondern diese muss numerisch bestimmt werden. In **R** ist dies implementiert, und zwar mit der Funktion `ar.burg()`. Als Beispiel passen wir ein $AR(p)$ an die Wellenhöhen an:

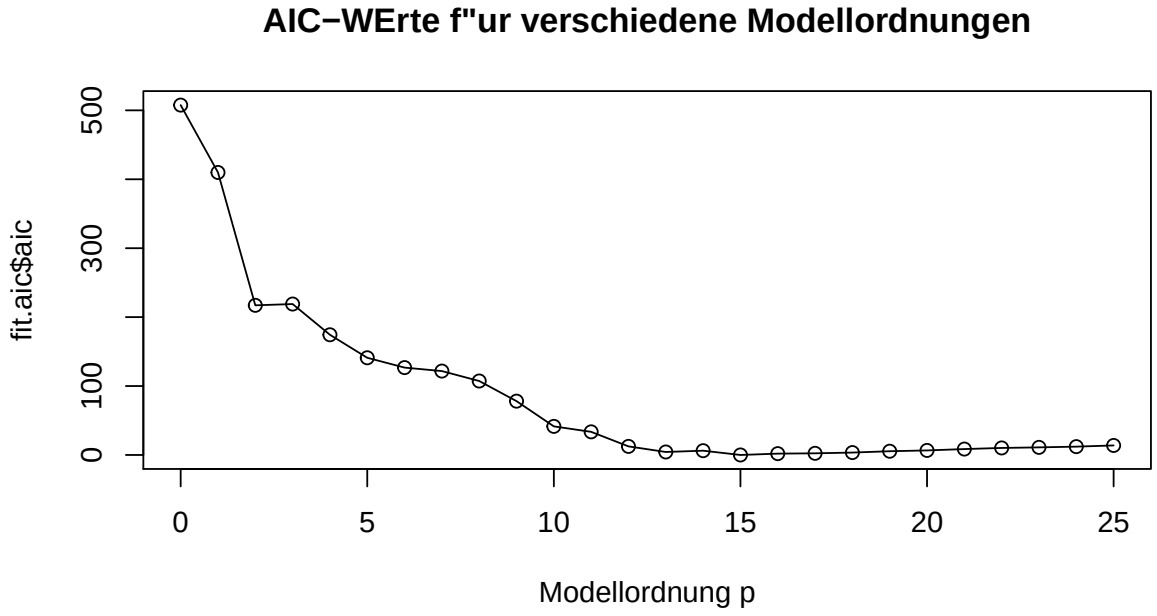
Die beiden Koeffizienten ergeben sich als $\alpha_1 = 0.7660026$ und $\alpha_2 = -0.6233047$. Es wird auch noch die geschätzte Varianz der Innovation angegeben, $\widehat{\sigma}_E^2 = 3.3688033 \times 10^4$. Zu beachten sind auch noch die Argumente der Funktion `ar.burg()`. Mit `aic=FALSE` teilen

wir mit, dass die Modellordnung bereits festgelegt ist, und **R** nicht noch mit Hilfe des AIC-Kriteriums nach dem optimalen p suchen soll. Mit `order.max=2` schreibt man dieses gewählte $p = 2$ fest.

Wie erwähnt besteht eine Alternative darin, die Ordnung automatisch via AIC zu wählen. Hierfür setzen wir `aic=TRUE`, bzw. lassen dieses Argument einfach weg, denn es ist per Default so gesetzt. Mit `order.max` spezifizieren wir dann, bis zu welchem p gesucht werden soll. Im Hintergrund werden dann sämtliche Modelle evaluiert. Falls man keine maximale Modellordnung angibt, so trifft **R** automatisch eine sinnvolle Wahl, nämlich $max(p) = 10 \cdot \log_{10}(n)$, wobei n der Länge der Zeitreihe entspricht.

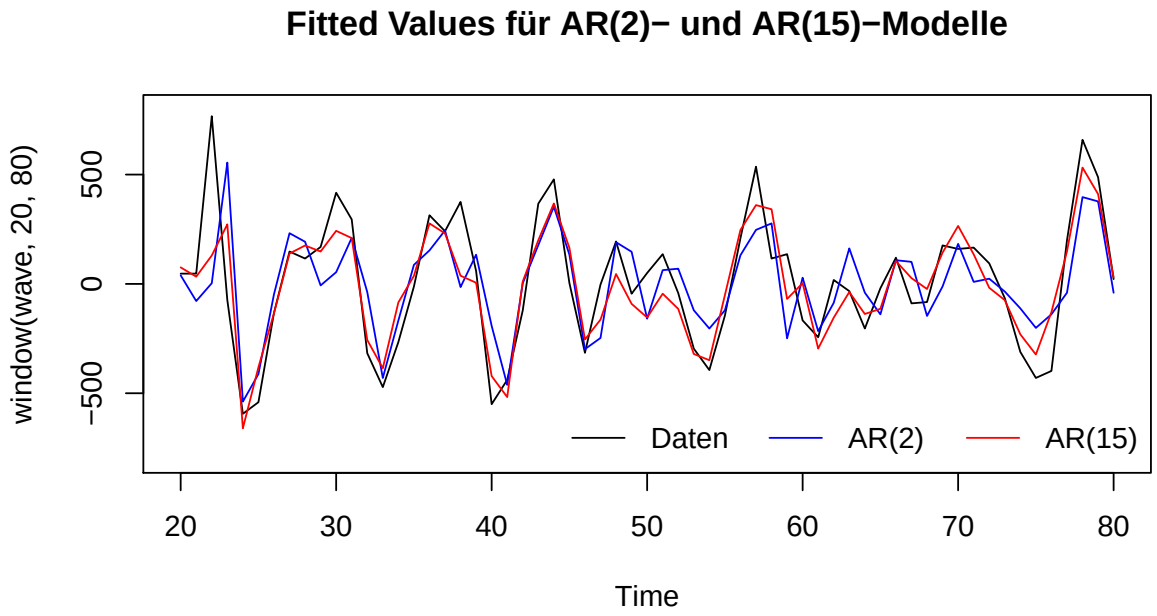
```
Call:
ar.burg.default(x = wave)

Coefficients:
      1        2        3        4        5        6        7        8
 0.2722  -0.9497  -0.4564  -0.7059  -0.7787  -0.7700  -0.6899  -0.6656
      9       10       11       12       13       14       15
-0.5398  -0.6144  -0.3225  -0.2666  -0.2862   0.0207  -0.1433

Order selected 15  sigma^2 estimated as  18233
```

Wie wir sehen, wurde hier ein $AR(15)$ als optimal befunden. Wir können die errechneten AIC-Werte grafisch darstellen, um die Performance verschieden grosser Modelle zu beurteilen.



Wir beobachten, dass ein $AR(2)$-Modell gegenüber einem $AR(1)$ oder gar nichts schon eine substanzielle Verbesserung ist. Andererseits zeigt sich von $p = 3$ bis $p = 15$ eine kontinuierliche Verbesserung des AIC-Kriteriums. Dies deutet darauf hin, dass das grössere Modell vermutlich die bessere Wahl ist. Wir versuchen dem weiter auf die Spur zu gehen, indem wir die Fitted Values und die Residuen betrachten. Erstere stellen wir wie folgt dar:

```
>  fv <- wave-fit$resid
>  fv.aic <-wave-fit.aic$resid
>  plot(window(wave,20,80),ylim=c(-800,800))
>  title('Fitted Values für AR(2)- und AR(15)-Modelle')
>  lines(20:80, fv[20:80],col='blue')
>  lines(20:80, fv.aic[20:80],col='red')
>  legend("bottomright", bty ="n", horiz= T,
+          legend = c('Daten', 'AR(2)', 'AR(15)'),
+          col= c('black', 'blue', 'red'), lty = 1)
```



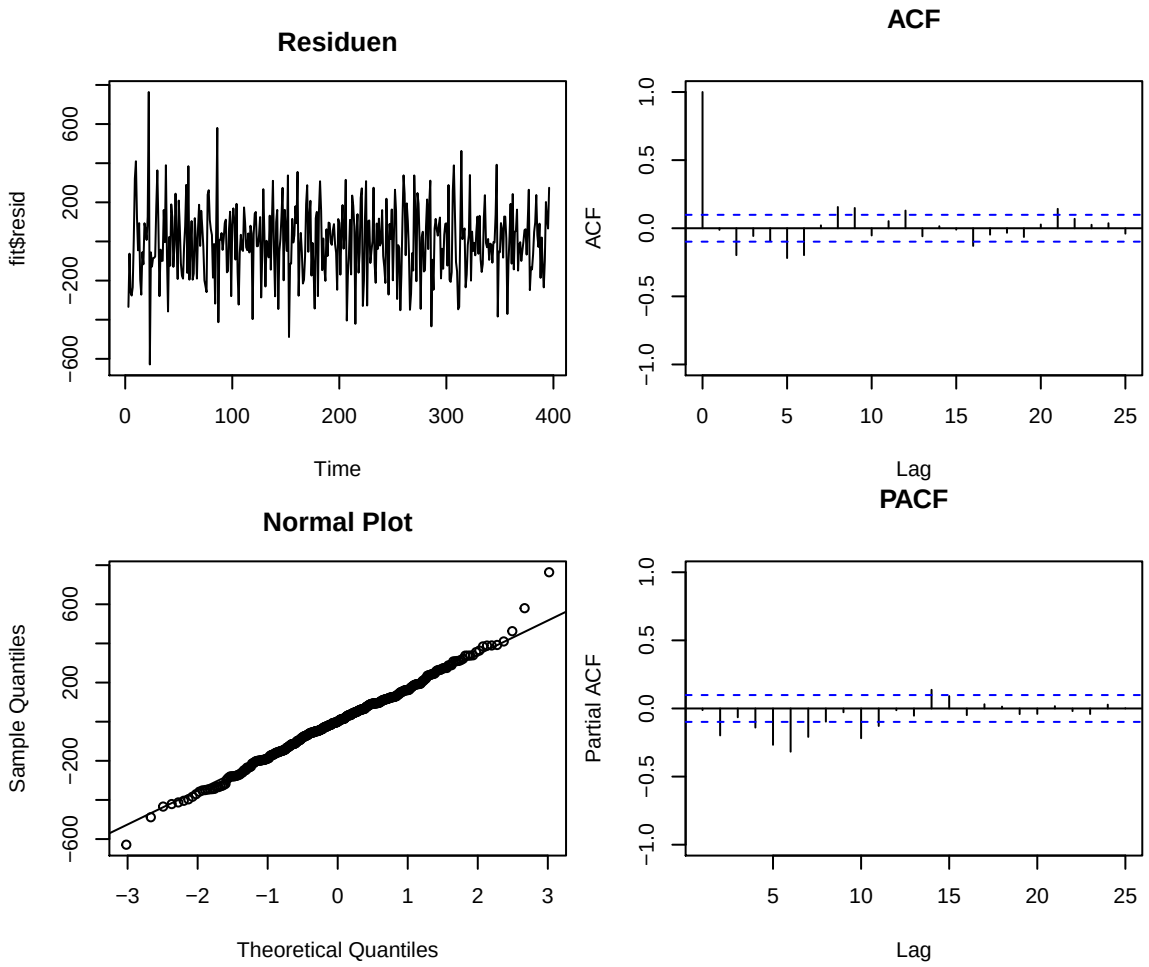
Wir erkennen, dass die rote Linie des $AR(15)$ dem Verlauf der Daten besser zu folgen vermag als das in blau gehaltene $AR(2)$. Dies ist nicht weiter erstaunlich, handelt es sich doch um das komplexere Modell mit mehr Parametern. Entscheidend ist nun, ob das kleinere Modell bereits genügend gut ist oder nicht. Das AIC-Kriterium meint nein, eine zweite Antwort folgt im nächsten Abschnitt.

Residuenanalyse

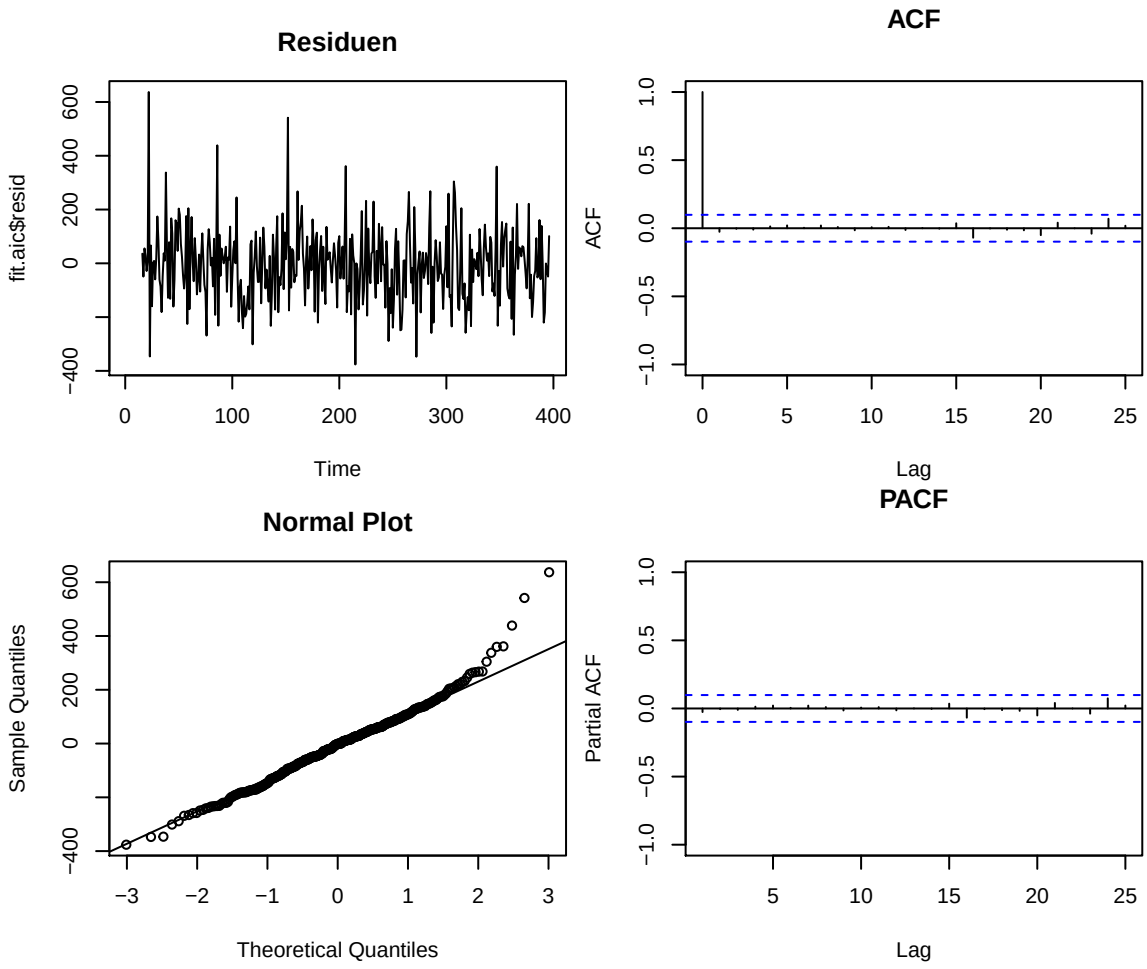
Die Residuen aus den AR-Fits sind eine Schätzung der Innovationen E_t . Damit wir einem Modell eine gute Anpassung attestieren können, sollten die Residuen Eigenschaften aufweisen, welche die Annahmen für E_t widerspiegeln. D.h. konkret, dass die Residuen wie weisses Rauschen auszusehen haben. Wir können dies überprüfen, indem wir ACF und PACF betrachten.

Weiter wird für autoregressive Modelle zwar nicht zwingend vorausgesetzt, dass die Innovationen normalverteilt zu sein brauchen. Dennoch sollten die Abweichungen davon nicht zu stark sein, weil sonst der Algorithmus von Burg zur Schätzung der Koeffizienten rasch ineffizient wird. Es empfiehlt sich daher, auch noch einen Normalplot der Residuen anzufertigen, und bei allzu groben Verletzungen bei Interpretation und Vorhersage vorsichtig zu sein. Wir führen diese Schritte für das angepasste $AR(2)$ aus.

```
> par( mfrow= c(2,2), mar = c(4, 4, 3, 0.5))
> plot(fit$resid, main = 'Residuen')
> acf(fit$resid, ylim= c(-1,1), na.action=na.pass, main ="ACF")
> qqnorm(fit$resid, main = 'Normal Plot')
> qqline(fit$resid)
> pacf(fit$resid, ylim=c(-1,1), na.action=na.pass, main = 'PACF')
```



Rein visuell sind in den Residuen keine grossen Strukturen mehr enthalten, und der Normal Plot zeigt keine Auffälligkeiten. Die beiden Korrelogramme zeigen aber eindeutig, dass in den Residuen noch Abhängigkeiten vorhanden sind. Dies ist so nicht erlaubt! Es ist als Hinweis zu taxieren, dass ein $AR(2)$ nicht adäquat ist, und ein Modell von grösserer Ordnung p zu wählen ist. Nahe liegend ist es, das vom AIC gewählte $AR(15)$ zu evaluieren. Dessen Residuenplots sind auf der nächsten Seite abgebildet und zeigen keine unerwünschten Abhängigkeiten mehr. Wäre dies so, d.h. zeigt auch das AIC-optimale Modell noch Strukturen in den Residuen, so bleibt nur festzustellen, dass die in den Daten vorhandene Abhängigkeit mit einem AR-Modell nicht zu beschreiben ist. Dies kann durchaus der Fall sein. Es besteht dann die Möglichkeit, andere, komplexere Modelle einzusetzen. Diese sind als ARMA-Modelle bekannt (siehe Abschnitt **??**).

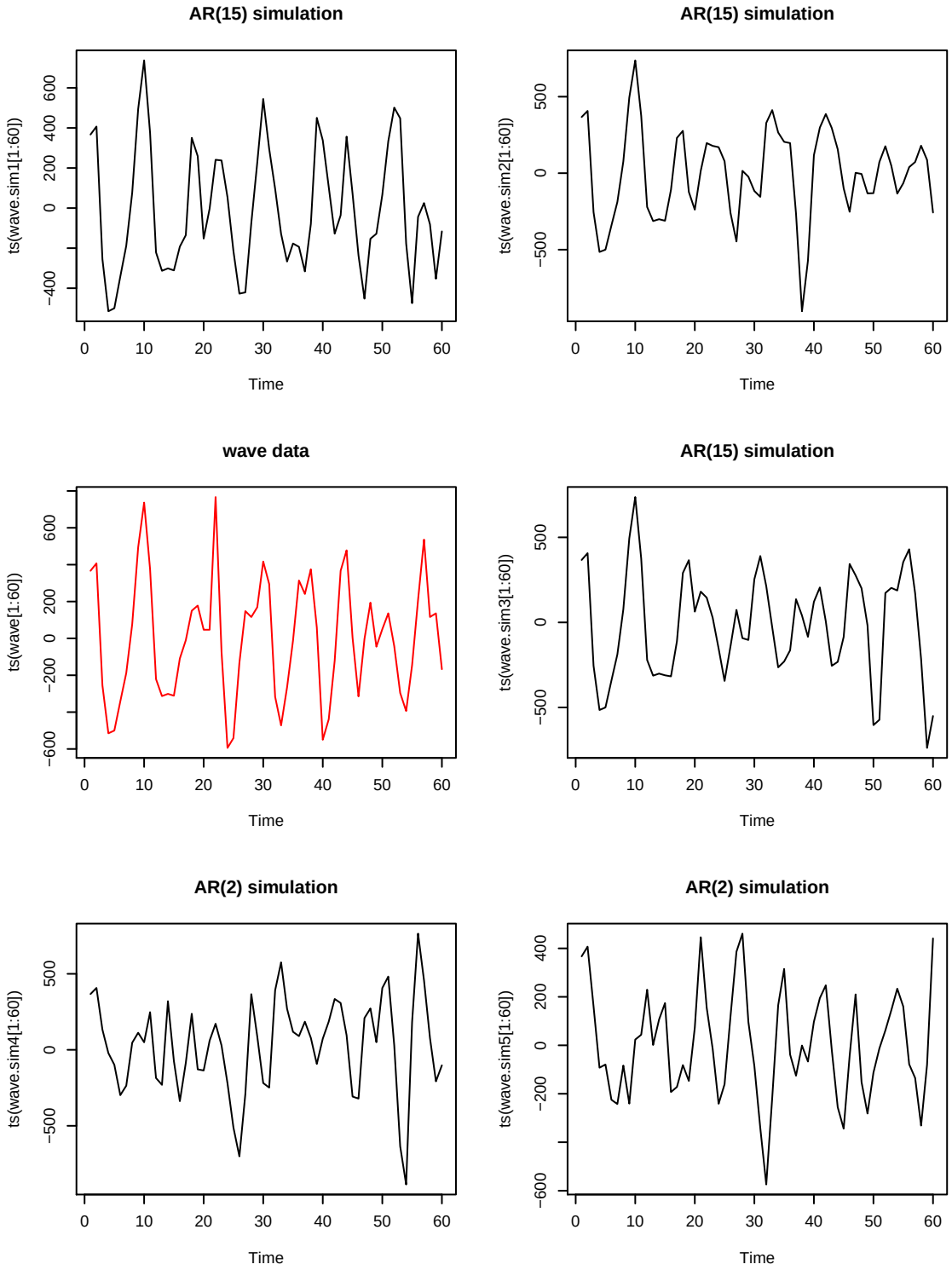


5.2.3 Diagnostik durch Simulation

Ein AR-Modell kann mit den geschätzten Parametern ohne weiteres simuliert werden, wenn man die Verteilung der Innovationen E_t kennt oder modelliert hat. Die Anfangswerte der beobachteten Zeitreihe werden als Startwerte zur Simulation der Modellreihe Y_t^* verwendet. Nun vergleicht man die simulierte und die beobachtete Reihe mit Zeitreihenplots. Falls das Modell die beobachtete Reihe gut beschreibt, muss die simulierte Reihe das gleiche Verhalten zeigen. Aufgrund der simulierten Innovationen stimmen die beiden Reihen sicher nicht exakt überein. Deshalb empfiehlt es sich, mehrere Reihen zu simulieren und diese mit der beobachteten zu vergleichen. Sie können z.B. wie folgt vorgehen: Ordnen Sie die beobachtete Zeitreihe an irgend einer Stelle in der Tabelle der simulierten Zeitreihen ein. Nun zeigen Sie diese Tabelle einem Kollegen. Falls der beobachtete Zeitreihenplot aufgrund spezieller Merkmale, bezüglich derer er sich von den simulierten Reihen abhebt, entdeckt wird, ist das Modell nicht gut gewählt worden.

Für die Wave-Daten habe ich einmal 3 Zeitreihen mit dem $AR(15)$- und 2 Zeitreihen mit dem $AR(2)$-Modell simuliert. Auf der nächsten Seite sind die 4 Simulierten Zeitreiehen für die ersten 60 Zeitpunkte zusammen mit den Wave-Daten dargestellt. Die Simulationen

mit dem $AR(15)$-Modell reproduzieren die Strukturen der Wave-Daten deutlich besser als diejenigen mit dem $AR(2) - Modell$.



5.3 Moving Average Modelle

Das Moving Average-Modell (MA(q)-Modelle) ist eine Erweiterung des Weissen Rauschen, d.h. es ist eine Linearkombination aus der aktuellen Innovation plus ein paar Innovationen aus der Vergangenheit. Wie wir sehen werden, führt dies immer zu einem stationären Zeitreihenprozess, wobei der Prozess nicht iid ist. Ausserdem werden wir sehen, das MA(q)-Modelle die Autoregressiven-Prozesse ergänzen.

5.3.1 Definition und Eigenschaften

Wie oben erwähnt, ist ein *Moving Average Prozess* der Ordnung q , oder Abgekürzt, ein $MA(q)$-Modell für eine ZR X_t eine Linearkombination aus der aktuellen Innovation E_t plus ein paar Innovationen aus der Vergangenheit $E_{t-1}, \ldots, Et - q$. Die Modellgleichung lautet:

$$X_t = E_t + \beta_1 E_{t-1} + \cdots + \beta_q E_{t-q} \tag{17}$$

Es ist nötig, dass E_t eine Innovation ist, also unabhängig und bleichverteilt und zudem auch unabhängig von jedem X_s, wobei $s < t$. Für eine einfachere Schreibweise können wir den Backshift-Operator verwenden:

$$X_t = (1 + \beta_1 B + \cdots + \beta_q B^q) E_t = \Phi(B) E_t \tag{18}$$

$Phi(B)$ ist das charakteristische Polynom des $MA(q)$-Prozesses und definiert alle Eigenschaften der Zeitreihe.

Wozu braucht es die $MA(q)$-Prozesse?

Erstens, sie werden sehr erfolgreich in verschiedenen Anwendungsfeldern, speziell in der Ökonomie, eingesetzt. Zeitreihen wie z.B. ökonomische Indikatoren werden durch eine Reihe von zufälligen Ereignissen wie, Streiks, Regierungsentscheidungen, Referenden, Güterknappheit usw. beeinflusst. Diese Ereignisse haben nicht nur einen sofortigen, direkten Effekt auf den Indikator, sondern beeinflussen seinen Wert auch noch in den kommenden Perioden (allerdings in einem geringen Ausmass). Es ist somit plausibel, dass Moving Average Prozesse in der Praxis auftreten. Zudem sind einige seiner theoretischen Eigenschaften eine schöne Ergänzung zu den $AR(p)$-Prozessen. Nachdem wir die Momente der $MA(q)$-Prozesse eingeführt haben, sollte dies klarer werden.

Momente und Abhängigkeiten

Ein einfache aber sehr wichtige Eigenschaft jedes $MA(q)$-Prozesse X_t ist, dass er als Linearkombination von innovations Termen den Mittelwert null aufweist und eine konstante Varianz hat:

$$E[X_t] = 0 \text{ für alle } t \tag{19}$$

$$Var[X_t] = \sigma_E^2 \cdot (1 + \sum_{j=1}^{q} \beta_j^2) = \text{konst.} \tag{20}$$

Es ist zu beachten, dass wir ein in der Praxis von null verschiedenen $MA(q)$ -Prozess durch addieren einer Konstanten m beschreiben können:

$$Y_t = m + X_t. \tag{21}$$

Daher beeinflusst die Eigenschaft von Gleichung (19) die praktischen Anwendungen nicht. Wenn wir jetzt noch zeigen können, dass die Autokovarianz eines $MA(q)$-Prozesses zeitunabhängig ist, haben wir ihre Stationarität bewiesen. Dies ist auch der Fall. Wir starten mit einem $MA(1)$ Prozess mit $X_t = E_t + \beta_1 E_{t-.1}$:

$$\gamma(k) = Cov[X_t, X_{t-1}] = Cov[E_t + \beta_1 E_{t-1}, E_{t-1} + \beta_1 E_{t-2}] = \beta_1 \sigma_E^2$$

Für jeden Lag k grösser Ordnung $q = 1$, benützen wir den 'plugging-in' Trick in der Modellgleichung und erhalten direkt das vielleicht etwas überraschende Resultat:

$$\gamma(1) = Cov[X_t, X_{t-k}] = 0 \text{ für alle } k > q$$

Es gibt keine unkonditionale serielle Abhängigkeit für Lags > 1. Für die Autokorrelation eines $MA(q)$-Prozesses schreiben wir:

$$\rho(1) = \frac{\gamma(1)}{\gamma(0)} \text{ und } \rho(k) = 0 \text{ für alle} k > q = 1. \tag{22}$$

Daraus schliessen wir, dass $\rho(1) \leqslant 0.5$ unabhängig von der Wahl von β_1. Falls wir in der Praxis einen Autokorrelationskoeffizienten beobachten der diesen Wert deutlich überschreitet, können wir davon ausgehen, dass es sich nicht um einen $MA(1)$-Prozess handelt. Des Weiteren haben wir gezeigt, dass jeder $MA(1)$-Prozess Erwartungswert null besitzt, eine konstante Varianz aufweist und die ACF nur vom Lag k abhängt und somit der Prozess stationär ist. Im Gegensatz zu $AR(p)$-Prozessen hängt die Stationarität nicht von der Wahl des Parameters β_1 ab. Die Stationaritätseigenschaft kann für $MA(q)$ verallgemeinert werden. Mit einigen Rechnungen und $\beta_0 = 1$ erhalten wir:

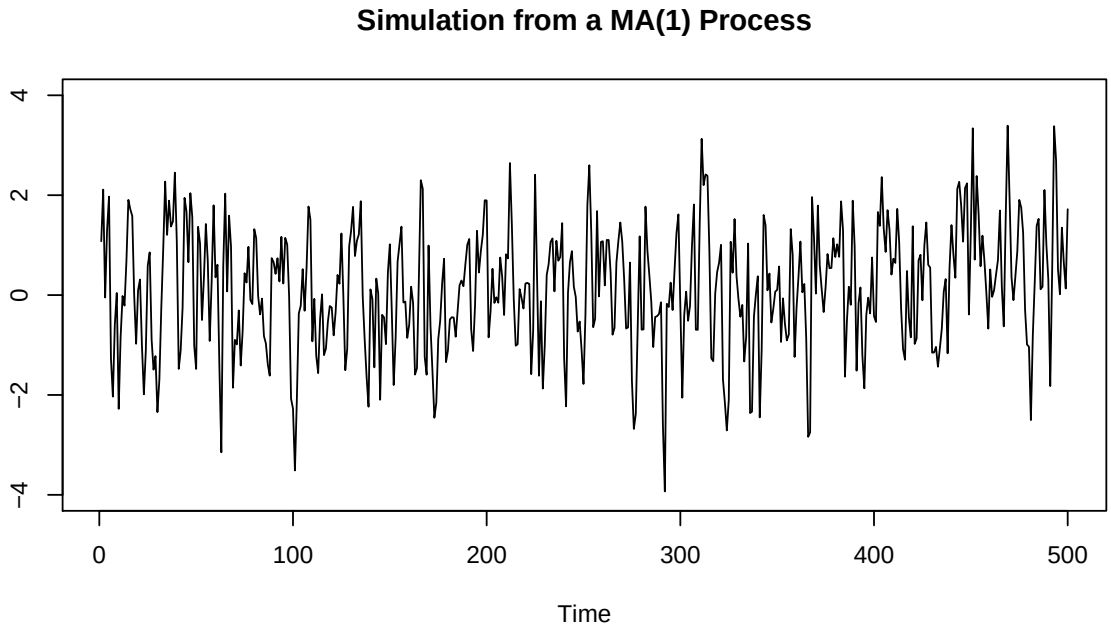
$$\rho(k) = \begin{cases} \sum_{j=0}^{q-k} \beta_j \beta_{j+k} / \sum_{j=0}^{q} \beta^2 & \text{für } k = 1, \ldots, q \\ 0 & \text{für } k > q. \end{cases} \tag{23}$$

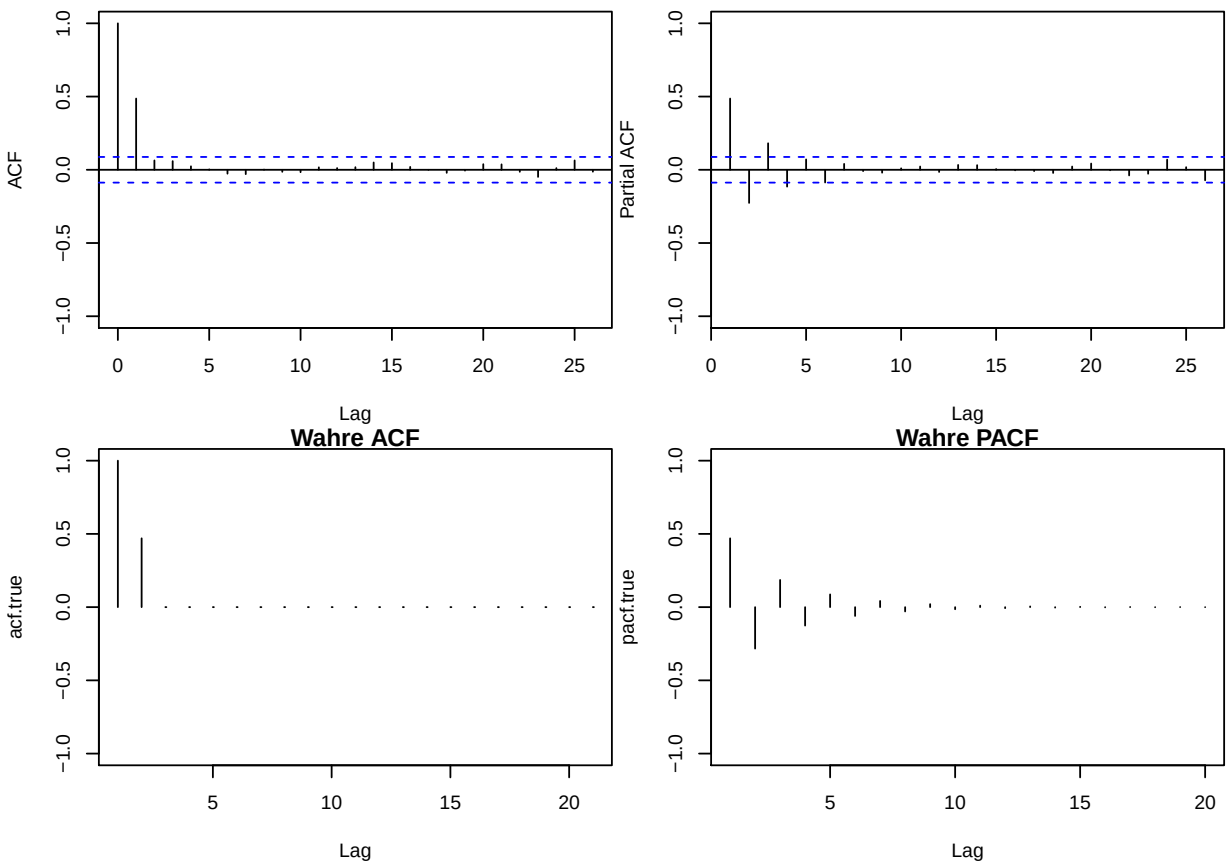
Folglich ist $\rho(k)$ unabhängig von der Zeit t für jeden $MA(q)$-Prozess mit beliebiger Ordnung q. Dies bedeutet das $MA(q)$-Prozesse *immer stationär sind, unabhängig von* $\beta_1, \ldots, \beta_q$. Zudem sehen wir in (23), dass für alle Ordnungen $k > q$ die Autokorrelation null ist und ein Zusammenhang (recht komplex für höhere Ordnungen) zwischen den Parametern und der Autokorrelation existiert. Die Formel (23) kann verwendet werden um für einen gegebenen $MA(q)$ die theoretische ACF zu berechnen. Allerdings geht es auch einfacher mit der **R**-Funktion `ARMAacf()`.

Beispiel: $MA(1)$

Zur Illustration generieren wir eine Realisation von 500 Beobachtungen eines MA(1)-Prozesses mit $\beta_1 = 0.7$. Dazu verwenden wir die **R**-Funktion `arima.sim()` und stellen die generierte Zeitreihe zusammen mit der geschätzten ACF und PACF dar.

```
> set.seed(21)
> ts.ma1 <- arima.sim(list(ma=0.7), n=500)
> acf.true  <- ARMAacf(ma=0.7, lag.max=20)
> pacf.true <- ARMAacf(ma=0.7, pacf=TRUE, lag.max=20)
>
> plot(ts.ma1, ylab="", ylim=c(-4,4))
> title("Simulation from a MA(1) Process")
```






```
acf.true  <- ARMAacf(ma=0.7, lag.max=20)
pacf.true <- ARMAacf(ma=0.7, pacf=TRUE, lag.max=20)
```

Wir beobachten, dass die Schätzungen recht genau sind: die ACF hat einen klaren cut-off, wohingegen die PACF ein alternierendes Muster zeigt und die absoluten Werte exponentiell fallen. Dieses Verhalten ist typisch: die PACF Werte eines MA(q)-Prozesses fallen exponentiell während die ACF Werte einen cut-off aufweisen. In dieser Hinsicht steht ein $MA(q)$-Prozess in vollem Gegensatz zu einem $AR(p)$, d.h. das Muster der ACF und PACF sind getauscht.

Invertierbarkeit

Es ist einfach zu zeigen, dass der erste Autokorrelationskoeffizient $\rho(1)$ eines $MA(1)$-Prozesses sowohl in der Standardform als auch folgendermassen geschrieben werden kann:

$$\rho(1) = \frac{\beta_1}{1+\beta_1^2} = \frac{1/\beta_1}{1+(1/beta_1^2)}. \tag{24}$$

Offenbar hat jeder $MA(1)$-Prozess mit Koeffizient β_1 die exakt gleiche ACF wie der Prozess mit $1/beta_1$. Daher weisen die beiden Prozesse $X_t = E_t + 0.5E_{t-1}$ und $X_t = E_t + 2E_{t-1}$

die gleiche Abhängigkeitsstruktur auf. Mit anderen Worten für eine gegebene ACF könne wir den generierenden Prozess nicht eindeutig bestimmen. Diese Problem der Doppeldeutigkeit führt zum Konzept der Invertierbarkeit. Wenn wir die Prozess X_t und U_t durch $X_{t-1}, X_{t-2}, \ldots$ respektive $U_{t-1}, U_{t-2}, \ldots$ ausdrücken, finden wir durch sukzessive Substitution:

$$E_t = X_t - \beta_1 X_{t-1} + \beta^2 X_{t-2} - \ldots \tag{25}$$

$$E_t = U_t - (1/\beta_1) U_{t-1} + (1/\beta^2) U_{t-2} - \ldots \tag{26}$$

Daher, wenn wir den $MA(1)$-Prozess als $AR(\infty)$ umschreiben, wird nur einer der beiden Prozesse konvergieren. Dies gilt für den Prozess mit $|\beta_1| < 1$, und somit ist dieser Prozess invertierbar. Die Invertierbarkeit ist eine wichtige Eigenschaft der $MA(q)$-Prozesse, vor allem wenn es darum geht, den Prozess an Daten anzupassen, weil die Parameterschätzung darauf beruht den MA-Prozess als $AR(\infty)$ zu formulieren.

Dies gilt auch für $MA(q)$-Prozesse höherer Ordnung. Die Invertierbarkeit eines $MA(q)$-Prozesses ist eingehalten falls die Lösungen des charakteristischen Polynoms $\Phi(B)$ ausserhalb des Einheitskreises liegen. Dies kann anhand der **R**-Funktion `polyroot()` überprüft werden.

5.3.2 MA(q)-Modellanpassung (Fitting)

Die Schätzung der Modellparameter eines $MA(q)$-Prozesses ist schwieriger als für einen $AR(p)$-Prozess. Grundsätzlich stehen nur zwei Verfahren zur Verfügung:

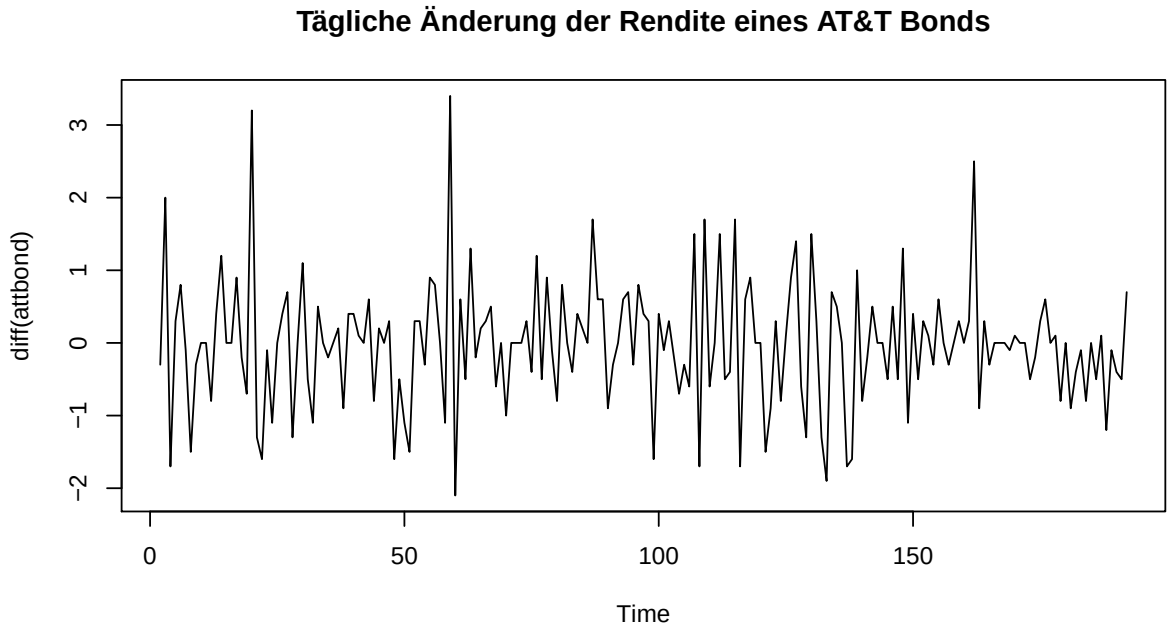
Conditional Sum of Squares (CSS) Dies ist eine adaptierte Kleinste-Quadrate-Schätzung, und wird auch dazu verwendet Startwerte für die ML-Schätzung zu generieren.

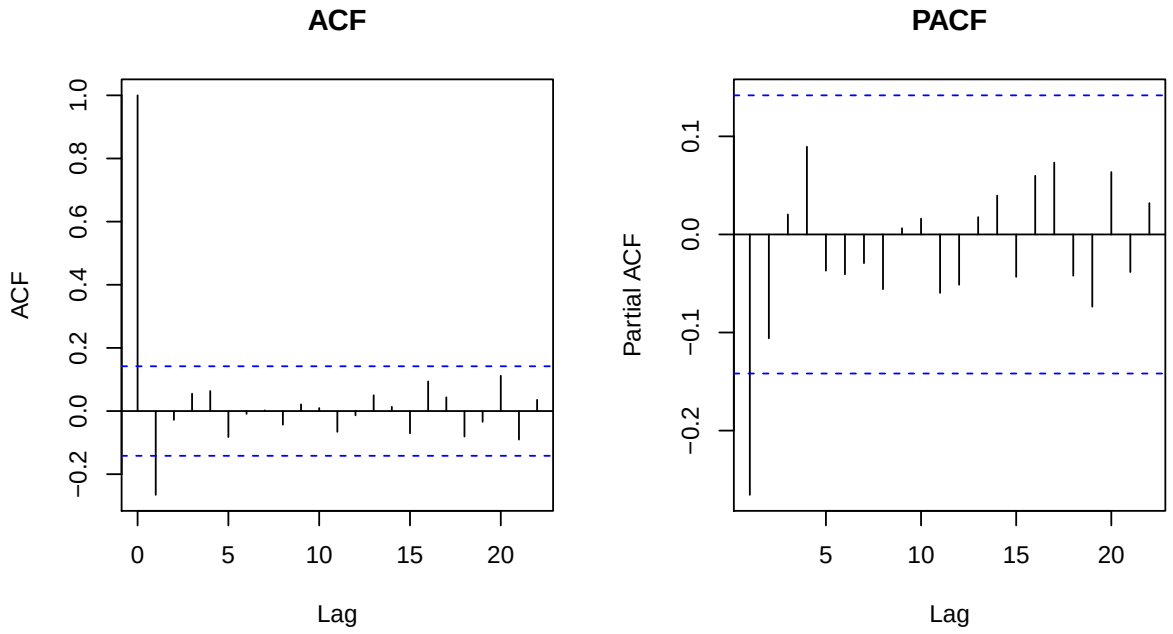
Maximum-Likelihood (ML) Die ML-Schätzung beruht darauf, dass die Daten aus einer multivariaten Gaussverteilung stammen. Die Parameter m, falls der $MA(q)$-Prozess geschiftet ist, $\beta_1, \ldots, \beta_q$ und σ_E^2 werden simultan geschätzt, in dem die Dichtefunktion der multivariaten Gaussverteilung, gegeben die Daten $x_1, \ldots, x_n$, maximiert wird.

In **R** können mit der Funkton `ariama()` die Parameter eines $MA(q)$-Prozesses geschätzt werden. Defaultmässig wird dazu die ML-Schätzung verwendet. Mit dem Argument `method` der Funktion `arima()` kann das Schätzverfahren gewählt werden. Bei der praktischen Anwendung sollte die `arima()`-Funktion immer mit der Default-Methode `method='CSS-ML'` verwendet werden.

5.3.3 Beispiel: Rendite eines AT&T Bondes

Als Beispiel betrachten wir die täglichen Änderungen der Rendite eines AT&T Bonds zwischen April und Dezember 1975 (Total: 182 Beobachtungen). Die Daten sind zusammen mit der ACF und PACF dargestellt.





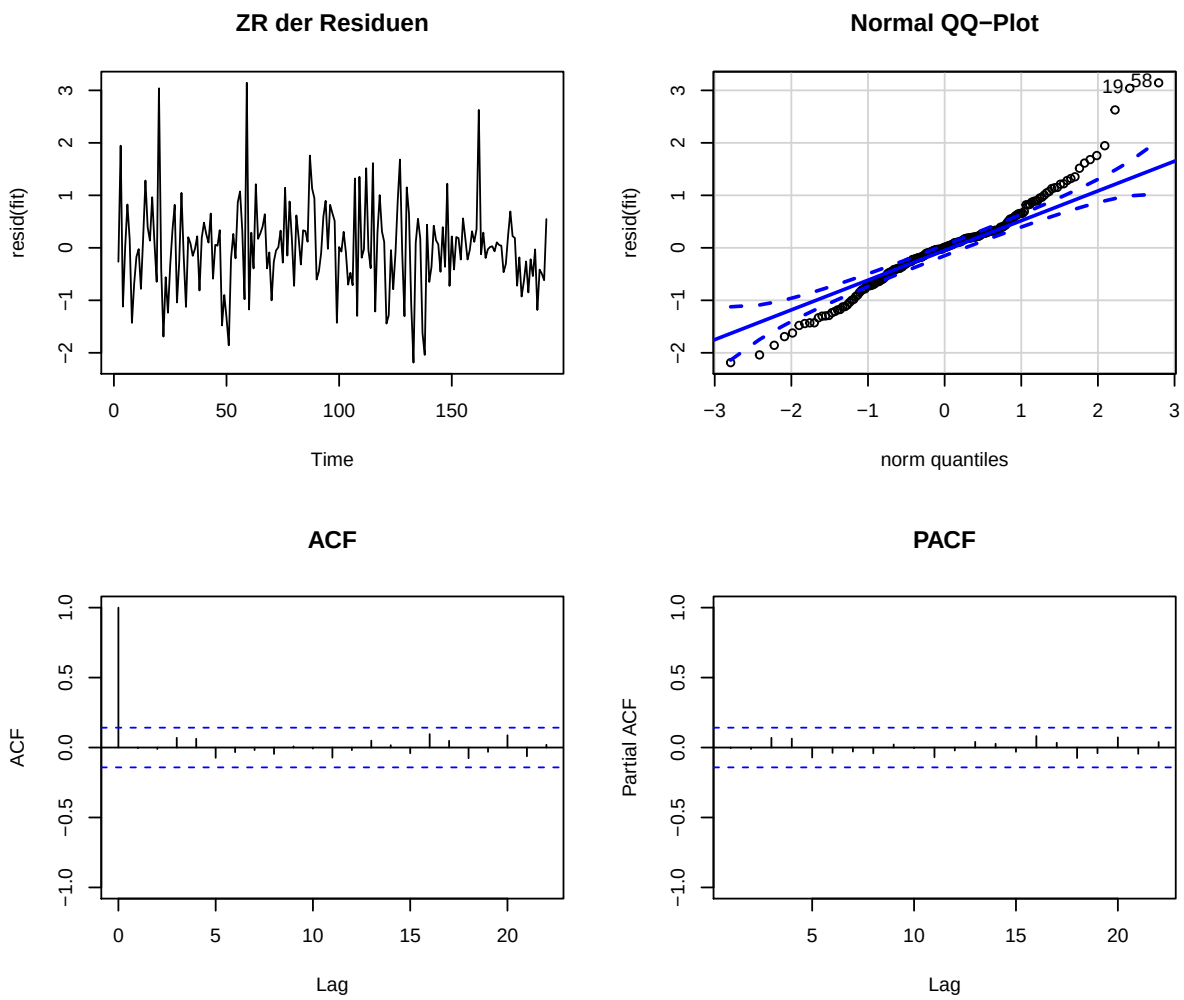
Die Reihe scheint von einem stationären Prozess zu kommen. Mit Hilfe von ACF und PACF beobachten wir einen klaren cut-off in der ACF nach Lag 1, was uns darauf schliessen lässt, dass es sich um einen $MA(1)$ handeln könnte. Allerdings fällt die PACF ebenfalls sehr sehr schnell auf kleine Werte. Wir passen nun mit der **R**-Funktion `arima()` einen $MA(1)$ an die Daten an.

```
> arima( diff( attbond), order = c(0,0,1))


Call:
arima(x = diff(attbond), order = c(0, 0, 1))

Coefficients:
          ma1  intercept
      -0.2865    -0.0247
s.e.   0.0671     0.0426

sigma^2 estimated as 0.6795:  log likelihood = -234.16,  aic = 474.31
```

Um den Fit zu überprüfen ist eine Residuenanalyse zwingend notwendig. Die Residuen des $MA(1)$-Prozesses sind die Schätzungen der Innovationen E_t . Das Modell kann also als adequate betrachtet werden, falls die Residuen die Haupteigenschaften der Innovationen wiedergeben. Sie müssen daher stationär sein und frei von jeder Abhängigkeitsstruktur und zudem approximativ eine Normalverteilung aufweisen (ML-Schätzung). Wir überprüfen diese Eingeschalten mit der ACF und PACF, sowie mit eine Normal QQ-Plot. Manchmal ist es auch instruktiv die gefitteten Werte zusammen mit den original Daten darzustellen, oder Realisationen des gefitteten Prozesses zu simulieren.

```
Loading required package: carData
```

```
[1] 58 19
```



Es gibt keine (partielle) Autokorrelation (Lag $\geqslant 1$), welche das Konfidenzband überschreitet. Daher können wir annehmen, dass die Residuen keine Korrelation mehr aufweisen und die gesamte Abhängigkeitsstruktur durch das $MA(1)$ Modell beschrieben wird. Die ZR der Residuen scheint stationär zu sein. Allerdings fallen die drei Ausreisser auf und die Langschwänzigkeit der Residuen. Allenfalls würden wir das mit einer *log*-Transformation wegbringen, aber das überlassen wir dem Leser.

5.4 ARMA(p,q) Modelle

In diesem Abschnitt diskutieren wir Modelle die sowohl eine Abhängigkeit auf vorhergehende Beobachtungen $X_{t-1}, X_{t-2}, \ldots$, als auch auf vorhergehende Innovationen $E_{t-1}, E_{t-2}, \ldots$ berücksichtigen. Daher sind sie ein Hybrid zwischen $AR(p)$- und $MA(q)$-Modellen und werden passend als $ARMA(p,q)$ bezeichnet. Mit diesen Modellen kann ein deutlich breiteres Spektrum von Abhängigkeitstrukturen beschrieben werden, die sehr sparsam sind:: ein $ARMA(p,q)$ braucht häufig (deutlich) weniger Parameter als ein reiner $AR(p)$ oder

$MA(q)$.

5.4.1 Definitionen und Eigenschaften

Die formale Definition eines $ARMA(p, q)$ ist folgendermassen¡:

$$X_t = \alpha_1 X_{t-1} + \cdots + \alpha_p X_{t-p} + E_t + \beta_1 E_{t-1} + \cdots + \beta_{t-q} E_{t-q} \tag{27}$$

Dabei nehmen wir wiederum an, dass E_t kausales weisses Rauschen ist, d.h. eine Innovation mit $E[E_t] = 0$ und endlicher Varianz $Var[E_t] = \sigma_E^2$. Es ist weitaus praktischer die charakteristischen Polynome $\Phi(\cdot)$ für den $AR(p)$ und $\Theta(\cdot)$ für den $MA(q)$ Teil zu benutzen, weil die Notation dadurch viel einfacher und kompakter wird:

$$\Phi(B)X_t = \Theta(B)E_t. \tag{28}$$

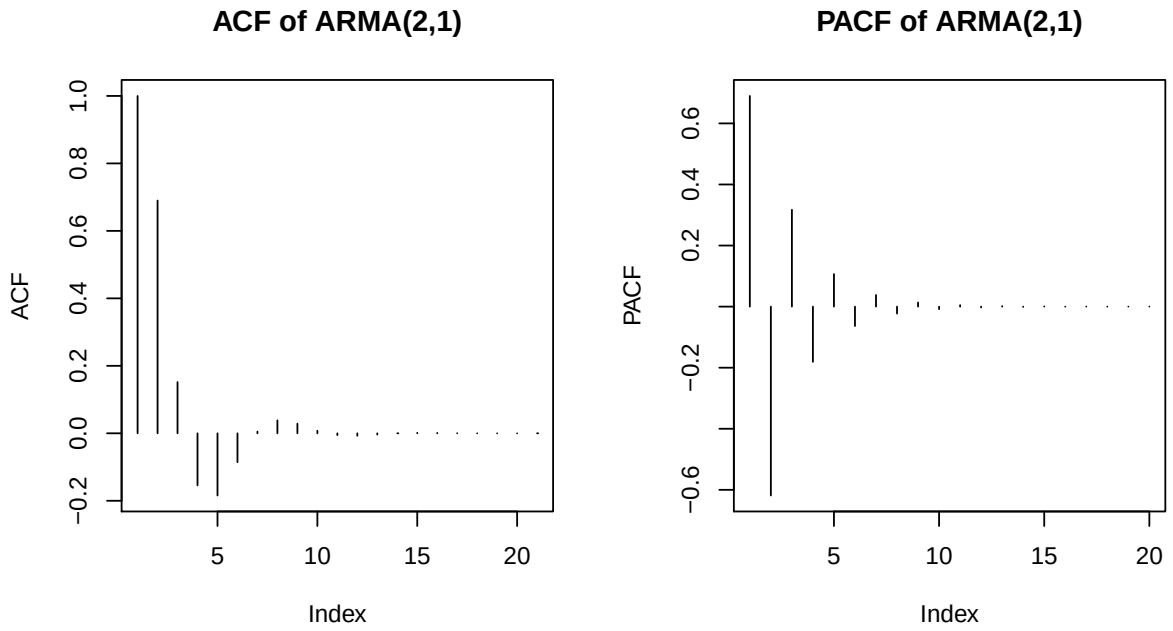
Offensichtlich liegen die relevanten Eigenschaften eines $ARMA(p, q)$-Prozesse auch in den charakteristischen Polynomen. Falls die Lösungen (Einheitswurzeln) von $\Phi(\cdot)$ ausserhalb des Einheitskreises liegen, ist der Prozess stationär und hat den Erwartungswert null. Falls die Lösungen von $\Theta(\cdot)$ ausserhalb des Einheitskreises, liegen ist der Prozess invertierbar. Beides sind wichtige Eigenschaften für die praktische Anwendung. Falls sie zutreffen, kann jeder $ARMA(p, q)$ in einen $AR(\infty)$ oder $MA(\infty)$ umformuliert werden. Daher sieht man auch häufig, dass statt eines $ARMA(p, q)$ ein AR- oder MA-Modell höherer Ordnung an die Daten angepasst wird (obwohl dies keine gute Idee ist!). Wie oben bereits bemerkt, hat jeder stationäre $ARMA(p, q)$ den Erwartungswert null, d.hj. $E[X_t] = 0$. In der Praxis betrachten wir daher oft einen verschobenen $ARMA$-Prozess von der folgenden Form:

$$Y_t = m + X_t, \text{ wobei } X_t \text{ ein } ARMA(p, q) \text{ ist } . \tag{29}$$

Im Prinzip ist es einfach die ACF eines $ARMA(p, q)$ abzuleiten, aber algebraisch recht mühsam. Aus diesem Grund und weil uns in diesem Skript eher die praktischen Anwendung interessieren, lassen wir das und fokussieren uns eher auf die Resultate und damit verbundenen Auswirkungen. Wir zeigen das typisch Verhalten der Autokorrelation eines $ARMA$-Prozesses anhand eines $ARMA(2, 1)$, welcher folgendermassen definiert ist:

$$X_t = -0.8X_{t-1} + 0.4X_{t-2} = E_t + 0.6 * Et - 1. \tag{30}$$

Die folgende Abbildung zeigt die *theoretischen* ACF und PACF des Prozesses.



Typischerweise zeigt weder die ACF noch die PACF einen klaren cut-off ab einem bestimmten Lag. Stattdessen weisen beide Korrelogramme auf eine unbeschränkte Fortsetzung hin, wobei die absoluten Werte der Koeffizienten mit zunehmenden Lag exponentiell kleiner werden. Dennoch zeigen die Werte der Koeffizienten beider Korrelogramme einen Sprung auf. In unserem Beispiel ist dies nach dem Lag 1 bei der ACF, hervorgerufen durch die moving average Ordnung $q = 1$, und nach dem Lag 2 bei der PACF verursacht durch die autoregressive Ordnung $p = 2$. Das generelle Verhalten der ACF und PACF ist in der folgenden Tabelle zusammen gefasst:

Model	ACF	PACF
$AR(p)$	unbeschränkt, exponentiell fallend	cutt-off nach Lag p
$MA(q)$	cutt-off nach Lag q	unbeschränkt, exponentiell fallend
$ARMA(p,q)$	unbeschränkt mix aus exp. fallend und cutt-off	

Es ist wichtig zu wissen, dass sich mit $ARMA(p,q)$-Prozessen eine sehr grosse Fülle von Autokorrelationstrukturen generiert lässt. Wie gut die Sprünge der Koeffizienten in den beiden Korrelogrammen (ACF und PACF) zu sehen sind und welche Struktur stärker überwiegt, Sprung oder unbeschränkter exponentieller Abfall, hängt von den Koeffizienten des Modells ab. Es gibt $ARMA$-Prozesse die vom AR Teil dominiert werden und andere, in dem der MA stärker ist und natürlich können auch beide gleich stärkt vertreten sein.

5.4.2 ARMA(p,q)-Modellanpassung (Fitting)

Die oben beschriebenen Eigenschaften der ACF und PAC können dazu verwendet werden, den Typ und Ordnung eines Zeitreihenmodells zu bestimmen. Falls weder die ACF noch die PACF einen deutlichen Sprung aufweisen und sich die Korrelationen schon nach ein paar

Lags (p und $q < 10$) nicht mehr signifikant von null unterscheiden, ist es im Allgemeinen sinnvoll, ein $ARMA(p,q)$-Modell zu wählen. Um die Ordnung (p,q) zu bestimmen, sucht man den überlagerten cutt-off in der ACF für q und PACF für p. Der Abfall ist in der Praxis nicht immer einfach zu identifizieren. In unklaren Situationen hat sich für die Wahl der Ordnung das AIC Kriterium zur Entscheidungsunterstützung bewährt. Wir können z.B. eine Rastersuche durchführen mit allen in Frage kommenden $ARMA(p,q)$-Modellen die nicht mehr als 5 Parameter verwenden, d.h. $p + q \leqslant 5$. Dies kann in R mit einer `for`-Schleife programmiert werden.

Ein wesentlicher Punkt den es beim Modellieren von $ARMA$-Prozessen zu berücksichtigen gilt, ist dass $ARMA$-Modelle nur eine geringe Ordnung von p und q benötigen (es sind sparsame Modelle). Gerade im Vergleich zu reinen AR -oder MA-Modellen benötigen $ARMA$-Modelle eine deutliche geringe Parameteranzahl. Oder mit anderen Worten ausgedrückt: ein $ARMA(p,q)$ kann oft durch ein $AR(p)$ oder $MA(q)$ höherer Ordnung modelliert werden. Dies ist ja auch nicht weiter erstaunlich wie wir in Abschnitt 5.4.1 bereits bemerkt haben, dass ein $ARMA(p,q)$ durch ein $AR(\infty$ oder ein $MA(\infty$ beschrieben werden kann. Dies ist jedoch in der Praxis keine gute Idee, da Parameterschätzungen etwas kosteten, d.h. je mehr Parameter geschätzt werden müssen, desto unpräziser werden die Schätzungen. Daher ist ein $ARMA(p,q)$- niedriger Ordnung immer einem $AR(p)$- oder $MA(q)$-Modell höherer Ordnung vorzuziehen.

Für die Schätzung der $ARMA(p,q)$-Koeffizienten gilt das Gleiche wie bei den $MA(q)$-Modelle, es sind keine expliziten Schätzer (= Formel zur Schätzung der Koeffizienten) vorhanden. Daher werden $ARMA$-Modelle so umgeschrieben, dass sie nur noch von den Beobachtungen aus der Vergangenheit von X_t abhängen. Danach werden Koeffizienten mit der ML-Methode geschätzt, wobei die Startwerte über die CSS-Methode bestimmt werden (siehe Abschnitt 5.3.2). In **R** kann man dazu die Funktion `arima()` verwenden mit den Argumenten: x = univarite Zeitreihe, `order = c(p,0,q)` Angabe der gewählten Ordnung, sowie weiteren Argumentes....

Der Vorteil der ML-Schätung liegt darin, dass (unter relativ milden und meist erfüllten Bedingungen) eine gewisse Optimalität garantiert ist. Im Besonderen sind die Schätzungen asymptotisch Normalverteilt mit minimaler Varianz unter allen asymptotisch normalverteilten Schätzungen. Das Schätzverfahren liefert zudem Standardfehler, welche einen Angabe der Präzision der Schätzung erlauben.

6 Vorhersage

Eine wesentliche Motivation zur Zeitreihenanalyse ist das Erzeugen von Vorhersagen für zukünftige Werte. Dabei handelt es sich um eine Extrapolation, und wie wir alle wissen, sind diese immer (zumindest ein wenig) problematisch, es besteht die Gefahr von Fehlschlüssen. Achtung: dies bedeutet nicht, dass man mit Zeitreihen keine nützlichen Vorhersagen erzeugen kann, doch man muss sich der Gefahren durch die Extrapolation bewusst sein. Für Laien kann die folgende Metapher illustrativ sein:

„Zeitreihen-Vorhersage ist wie Autofahren mit Blick durch den Rückspiegel. Auf einer Autobahn, die breit ist und kaum Kurven aufweist, wird dies vermutlich ganz gut funktionieren. Bewegen wir uns hingegen auf einer engen Bergstrasse mit scharfen Kurven, so wird es schwierig. Wenn, dann müssen wir sehr langsam und vorsichtig fahren, damit wir nicht von der Strasse abkommen.“

Die Aussage lässt sich leicht in die Welt der Zeitreihenanalyse übertragen. Es gibt Reihen, wo das Signal, d.h. der systematische Teil der Modellgleichung, viel stärker ist als die Innovation, d.h. die zufällige Streuung. Dann sind wir quasi „auf der Autobahn“, und eine Vorhersage ist vergleichsweise einfach. Der andere Fall, wo die Innovation stark und die Streuung gross ist, entspricht der Bergstrasse. Hier sind Vorhersagen, wenn überhaupt, nur für einen kurzen Zeithorizont informativ.

Die obige Einführung zielt darauf hin, dass die Schwierigkeit bei der Vorhersage von Zeitreihen vor allem im Verhältnis zwischen Signal und Innovation liegt. Das ist nicht falsch, doch in der Praxis treten noch weitere Schwierigkeiten auf:

- Wir müssen sicher sein, dass der in der Vergangenheit beobachtete Zeitreihen-Prozess sich in der Zukunft nicht ändert. Mit anderen Worten: *mit Blick durch den Rückspiegel merken wir es zu spät, wenn die Autobahn endet und wir uns plötzlich auf einer Bergstrasse befinden.*

- Für das Erzeugen von Vorhersagen verwenden wir Zeitreihenmodelle. Wir haben keine Gewähr, dass diese Modelle korrekt sind, d.h. die Modellwahl verursacht zusätzliche Unsicherheit.

- Die Zeitreihenmodelle enthalten Parameter, welche wir aus einer beobachteten Zeitreihe schätzen. Schätzungen sind nie exakt, sondern immer fehlerbehaftet. Dies wirkt sich auf die erzeugten Vorhersagen aus.

Nach diesen Vorsichtshinweisen werden wir nun aber verschiedene Herangehensweisen zur Zeitreihenanalyse kennen lernen. Wir betrachten zuerst stationäre Reihen und damit die Vorhersage aufgrund von $AR(p)$-Modellen. Dann wird erklärt, wie in Trend, Saison und stationären Rest zerlegte Zeitreihen vorhergesagt werden können. Zuletzt betrachten wir dann *Exponential Smoothing*, eine modellfreie, intuitive Vorhersagemethode für Zeitreihen.

6.1 Vorhersage mit $AR(p)$

In diesem Kapitel gehen wir davon aus, dass wir eine stationäre Zeitreihe X_t mit Beobachtungen $\{x_1, \ldots, x_n\}$ haben, für welche ein passendes $AR(p)$-Modell gefunden wurde. Passend bedeutet hier, dass die Parameter $\widehat{\alpha}_1, \ldots, \widehat{\alpha}_p$ aus den Daten geschätzt werden konnten, und dass die Residuen die Eigenschaften eines White-Noise-Prozesses zeigen.

Um der Mathematik willen gehen wir auch davon aus, dass $E[X_t] = 0$ gilt. Für Reihen mit $E[X_t] = \mu \neq 0$ können wir zuerst den Mittelwert abziehen, die unten beschriebenen Prognosen erzeugen, und am Schluss den Mittelwert wieder aufrechnen. Man beachte, dass dies in der Vorhersageprozedur von **R** automatisch ausgeführt wird. Wir führen nun etwas mathematische Notation ein:

$$\widehat{X}_{n+k|1:n} \text{ ist die } k\text{-Schritt-Vorhersage aufgrund von } \{x_1, \ldots, x_n\},$$

d.h. wir extrapolieren k Zeitschritte über die letzte Beobachtung zum Zeitpunkt n hinaus. Der Einfachheit halber verwenden wir in diesem Skript stets die abgekürzte Schreibweise $\widehat{X}_{n+k}$.

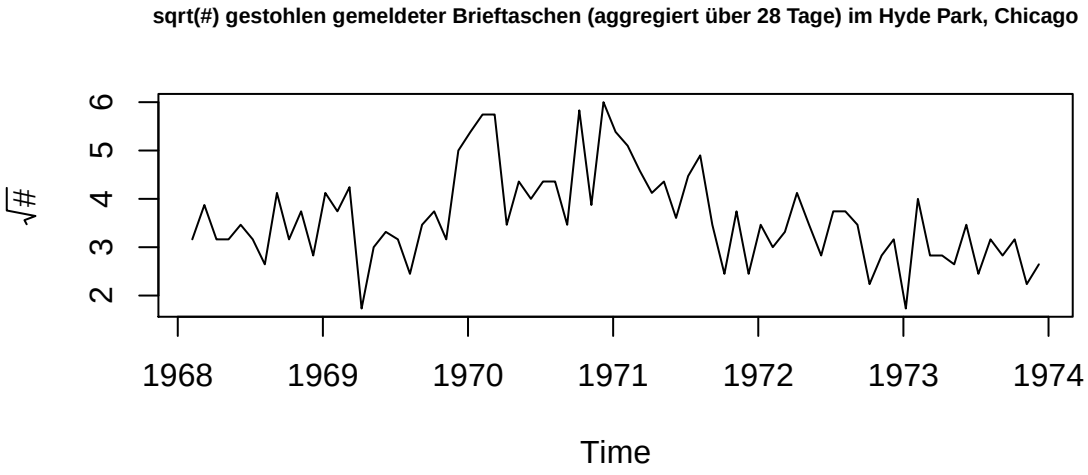
1-Schritt-Vorhersage

Die 1-Schritt-Prognose aufgrund von $AR(p)$-Modellen ist besonders einfach. Wir modellieren die nächste Beobachtung ja als Linearkombination über die letzten p, da erstaunt es nicht, dass wir die Modellgleichung auch zur Vorhersage heranziehen:

$$\text{AR(1): } \widehat{X}_{n+1} = \widehat{\alpha}_1 x_n \tag{31}$$

$$\text{AR(p): } \widehat{X}_{n+1} = \widehat{\alpha}_1 x_n + \widehat{\alpha}_2 x_{n-1} + \cdots + \widehat{\alpha}_p x_{n-p+1} \tag{32}$$

In der Modellgleichung (14) kommt bekanntlich auch noch der Innovationsterm E_t vor. Dessen beste Vorhersage ist aber der Wert null, weshalb er in den obigen Vorhersagegleichungen nicht vorkommt. Wenn wir in **R** ein $AR(p)$-Modell angepasst haben, ist das Erzeugen von 1-Schritt-Prognosen ganz einfach, wie wir am Beispiel der Brieftaschen-Diebstähle (Abbildung unten) sehen können:




```
> fit <- ar.burg(series)
> predict(fit)

$pred
Time Series:
Start = 1974.01666666667
End = 1974.01666666667
Frequency = 12
[1] 2.801125

$se
Time Series:
Start = 1974.01666666667
End = 1974.01666666667
Frequency = 12
[1] 0.7749009
```

Um Zeitreihen-Vorhersagen zu generieren, verwenden wir die Prozedur `predict()`. In der Default-Einstellung generiert diese eine 1-Schritt-Vorhersage. Beim Output handelt es sich um eine Liste mit 2 Zeitreihen-Objekten. Das erste ist die Vorhersage an sich, für den Zeitpunkt $n + 1$. Das zweite Objekt ist eine Zeitreihe mit dem Vorhersagefehler. Mit der Formel

$$\widehat{X}_{n+1} \pm 1.96 \cdot se(\widehat{X}_{n+1}) \tag{33}$$

erhalten wir ein 95%-Vorhersageintervall. Bei der Interpretation ist aber Vorsicht geboten: dieses Intervall deckt nur die durch die Innovation verursachte Unsicherheit ab. Weitere Unabwägbarkeiten wie sich zeitlich allenfalls verändernde Prozesse, Modell-Missspezifikation

und fehlerhafte Parameterschätzungen sind nicht berücksichtigt. Das Vorhersageintervall markiert also gewissermassen eine optimistische Grenze für die Genauigkeit. Oder mit anderen Worten: es dürfte real häufig grösser sein, als die Berechnung vorgibt.

Mehr-Schritt-Vorhersage

Das nächste Ziel ist nun nicht mehr nur die Vorhersage des nächsten Wertes über das Ende der Zeitreihe hinaus, sondern gleich der nächsten k Beobachtungen. Weil wir davon ausgehen, dass die Modellgleichung auch in Zukunft gilt, haben wir:

$$\widehat{X}_{n+k} = a_1 \widehat{X}_{n+k-1} \tag{34}$$

Die k-Schritt-Vorhersage erhält man also iterativ aus der $(k-1)$-Schritt-Prognose. Durch Rückwärtseinsetzen können wir den Prognosewert auf den letzten Beobachtungswert zurückführen:

$$\widehat{X}_{n+k} = a_1 \widehat{X}_{n+k-1} = a_1(a_1 \widehat{X}_{n+k-2}) = a_1^2(a_1 \widehat{X}_{n+k-3}) = \cdots = a_1^k x_n \tag{35}$$

Da ja wegen der Stationarität von X_t die Bedingung $|a_1| < 1$ gelten muss, geht a_1^k für wachsenden Prognosehorizont k gegen null. Die Prognose konvergiert also gegen null, den Erwartungswert der Reihe. Für $AR(p)$-Modelle lautet die allgemeine Vorhersageformel:

$$\widehat{X}_{n+k} = a_1 \widehat{X}_{n+k-1} + a_2 \widehat{X}_{n+k-2} + \cdots + a_p \widehat{X}_{n+k-p} \tag{36}$$

Prinzipiell könnte man auch hier Rückwärtseinsetzen, um die Vorhersage in Abhängigkeit der Beobachtungen zu schreiben, es ist jedoch nicht zweckmässig. Wir berufen uns hier auf die Konvention, dass man für $\widehat{X}_{n+k-j}$ falls erhältlich den beobachteten Wert x_{n+k-j} einsetzt. Existiert dieser nicht, so setzen wir den iterativ erhaltenen Prognosewert ein.

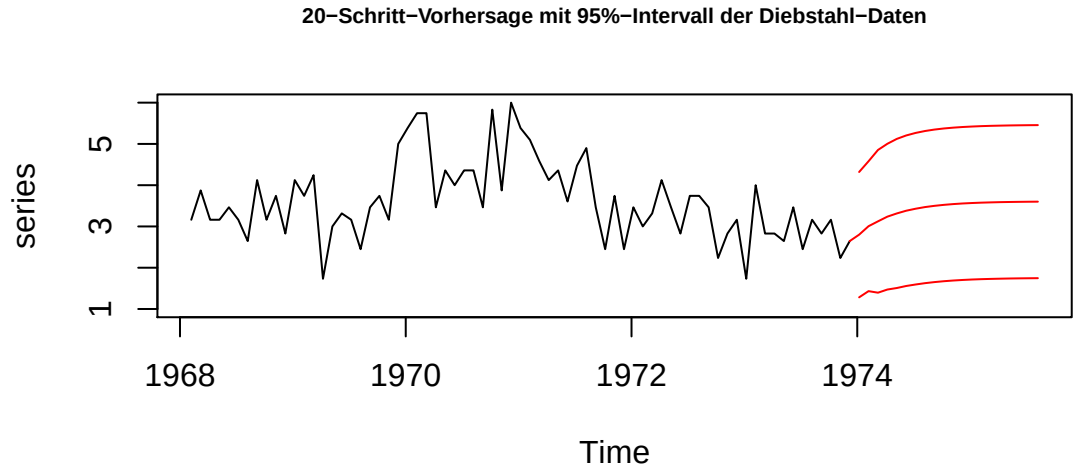
Wiederum, in R brauchen uns diese Schwierigkeiten nicht weiter zu kümmern. In die Prozedur `predict()` können wir ganz bequem ein angepasstes Modell stecken, dazu die Angabe der Vorhersagehorizonts mit dem Argument `n.ahead`. Wir erhalten dann die Zeitreihe der Vorhersagen, wie auch die Zeitreihe der Vorhersagefehler.

```
> fore <- predict(fit, n.ahead  = 20)
> fore

$pred
Time Series:
Start = 1974.01666666667
End = 1975.6
Frequency = 12
 [1] 2.801125 3.006212 3.123742 3.237219 3.314794 3.380993 3.429829 3.469422
 [9] 3.499600 3.523544 3.542050 3.556600 3.567912 3.576772 3.583677 3.589077
[17] 3.593290 3.596581 3.599151 3.601158

$se
```

```
Time Series:
Start = 1974.01666666667
End = 1975.6
Frequency = 12
 [1] 0.7749009 0.8030468 0.8822306 0.9010328 0.9218903 0.9304351 0.9371178
 [8] 0.9405592 0.9428905 0.9442138 0.9450576 0.9455568 0.9458670 0.9460538
[15] 0.9461686 0.9462382 0.9462809 0.9463068 0.9463226 0.9463323
```

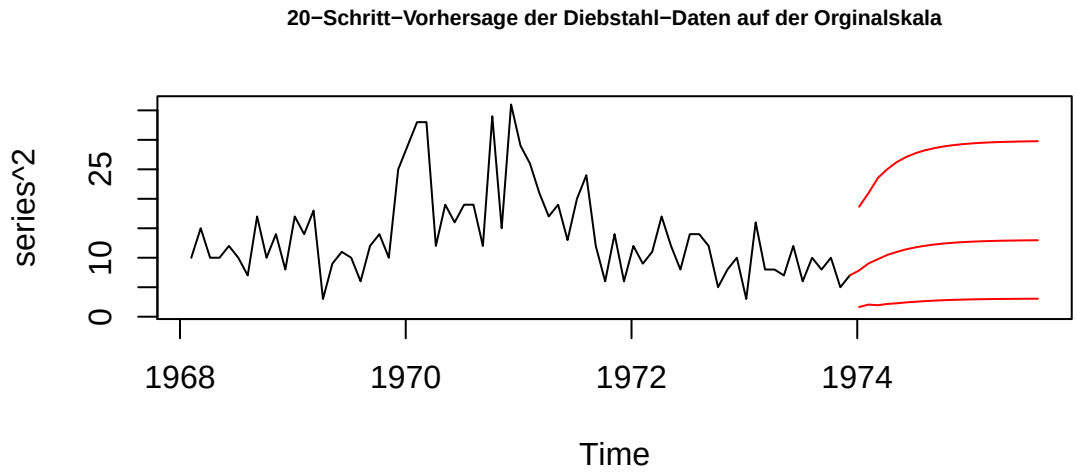
Informativer als ein Text-Output ist aber die grafische Darstellung der Prognosen und der Vorhersagefehler. Dazu ist der folgende Code nötig:

```
> as <- start(series); es <- end(series); kk = 20
> plot(series, xlim = c(as, es+ kk/12), ylim = c(1,6))
> title('20-Schritt-Vorhersage mit 95%-Intervall der Diebstahl-Daten',
+       cex.main = 0.7)
> #VPrognose einzeichnen
> lines(seq(es, es+kk/12, length = kk+1),
+       c( series[length(series)], fore$pred ),col="red")
> lines(fore$pred + 1.96*fore$se,col= "red")
> lines(fore$pred - 1.96*fore$se, col = "red")
```



Wir beobachten, dass die Vorhersage exponentiell schnell gegen den Mittelwert der Reihe konvergiert. Das 95%-Prognoseintervall wird mit zunehmendem Horizont grösser. Die Tatsache, dass es hier ziemlich breit ist, weist darauf hin, dass das Signal im Vergleich zur Innovation relativ schwach ist. Es ist auch zu beachten, dass wir hier die wurzeltransformierte Zeitreihe vorhergesagt haben. Eigentlich interessiert uns ja aber nicht die Wurzel der gestohlenen Portemonnaies, sondern deren tatsächliche Anzahl. Wir können aber sowohl die Vorhersage wie das zugehörige Intervall einfach zurücktransformieren, d.h. wir

bestimmen schlicht und einfach das Quadrat des vorhergesagten Wertes und quadrieren auch die Werte des 95%-Prognoseintervalls



6.2 Vohersage mit $MA(1)$-Modell

Wir betrachten einen invertierbaren **MA(1)-Prozess**, der gemäss

$$X_t = E_t + \beta_1 E_{t-1}, \quad E_t \text{ weisses Rauschen mit } E[E_t] = 0 \tag{37}$$

definiert ist. Wir wollen Vorhersagewerte für k Zeitschritte in die Zukunft berechnen. Wir unterscheiden die beiden Fälle $k = 1$ und $k \geqslant 2$. Wir betrachten zuerst den Fall $k \geqslant 2$, da dieser einfacher zu behandeln ist.

Für $k \geqslant 2$ erhalten wir für den bedingten Erwartungswert $\widehat{X}_{n+k|1:n}$

$$\begin{aligned}
\widehat{X}_{n+k|1:n} &= E[X_{n+k}|X_1, \ldots, X_n] \\
&= \underbrace{E[E_{n+k}|X_1, \ldots, X_n]}_{E[E_{n+k}]=0} + \beta_1 \cdot \underbrace{E[E_{n+k-1}|X_1, \ldots, X_n]}_{E[E_{n+k-1}]=0} \\
&= 0.
\end{aligned}$$

Beide Erwartungswerte sind gleich 0, da in $X1, \ldots, X_n$ nur Zufallsvariablen vorkommen, die aus dem weissen Rauschen (E_t) mit $t \leqslant n$ bestehen, daher sind E_{n+k} und $X1, \ldots, X_n$ unkorreliert. Somit ist die beste $MA(1)$ Vorhersage für $k \geqslant 2$ null.

Für Reihen mit $\widehat{X}_{n+k|1:n} = \mu \neq 0$ können wir zuerst den Mittelwert abziehen, die unten beschriebenen Prognosen erzeugen, und am Schluss den Mittelwert wieder aufrechnen.

Für $k = 1$ erhalten wir aber

$$\begin{aligned}\widehat{X}_{n+1|1:n} &= E[X_{n+1}|X_1, \ldots, X_n] \\ &= \underbrace{E[E_{n+1}|X_1, \ldots, X_n]}_{=0} + \beta_1 \cdot E[E_n|X_1, \ldots, X_n] \\ &= \beta_1 \cdot E[E_n|X_1, \ldots, X_n] \neq 0 \text{ (im Allgemeinen)}\end{aligned}$$

Eine einfache Aussage über den Term $E[E_n|X_1, \ldots, X_n]$ ist nicht möglich, insbesondere ist in der Regel $E[E_n|X_1, \ldots, X_n]$ ungleich 0. Dies liegt daran, dass X_n und E_n abhängige Zufallsvariablen sind. Diese Abhängigkeit besteht, da $X_n = E_n + \beta_1 \cdot E_{n-1}$ ist und somit E_n in X_n vorkommt. Da x_n in $E[E_n|x_1, \ldots, x_n]$ vorkommt, ist dieser bedingte Erwartungswert $E[E_n|x_1, \ldots, x_n]$ sehr schwierig zu berechnen.

Mit mathematischem Tricksen könne wir jedoch einen approximativen Wert dafür angeben. Der Trick besteht darin, das wir auf die unendliche Vergangenheit $\{-\infty \ldots, X_{-2}, X_{-1}, X_0, X_1, X_2, \ldots, X_n\}$ bedingen. Wir tun also so, wie wenn wir Beobachtungen bis beliebig weit zurück kennen würden. Dann können wir den bedingten Erwartungswert $E[E_n| - \infty \ldots, x_0, \ldots, x_n]$ berechnen. Die Basis für diese Lösung liegt darin, dass wir den MA(1) Prozess als $AR(\infty)$ schreiben können. Wir starten mit den Gleichungen

$$\begin{aligned}X_n &= E_n + \beta_1 E_{n-1} \\ X_{n-1} &= E_{n-1} + \beta_1 E_{n-2}\end{aligned}$$

und lösen die zweite Gleichung nach E_{n-1} auf und setzen dies in die erste ein. Dann erhalten wir

$$\begin{aligned}X_n &= E_n + \beta_1(X_{n-1} - \beta_1 E_{n-2}) \\ &= \ldots \quad \text{laufend } E_{n-k} \text{ ersetzen} \\ &= E_n + \beta_1(X_{n-1}) - \beta_1^2(X_{n-2}) + \beta_1^3(X_{n-3}) - \ldots\end{aligned}$$

Somit ist der MA(1)-Prozess nun als $AR(\infty)$-Prozess geschrieben. X_n kann also als Linearkombination der unendlichen Vergangenheit des Prozesses geschrieben werden.

Wir lösen nun die Gleichung (38) nach E_n auf

$$\begin{aligned}E_n &= X_n - \beta_1(X_{n-1}) + \beta_1^2(X_{n-2}) - \beta_1^3(X_{n-3}) + \ldots \\ &= \sum_{j=0}^{\infty}(-\beta_1)^j(X_{n-j}). \qquad (38)\end{aligned}$$

Wenn wir nun E_n auf die unendliche Vergangenheit $\{-\infty \ldots, x_0, \ldots, x_n\}$ bedingen, so kennen wir in (38) die rechte Seite vollständig und erhalten so

$$E[E_n| - \infty \ldots, x_0, \ldots, x_n] = \sum_{j=0}^{\infty} \beta_1^j(x_{n-j}).$$

Zur praktischen Durchführung ist diese Berechnug etwas problematisch, da wir nur eine endliche Vergangenheit des Prozesses kennen. Aber der Einfluss von vergangenen Beobachtungen nimmt exponentiell schnell ab, da es sich um einen invertierbaren $MA(1)$-Prozess, $|\beta_1| < 1$, handelt. Daher können wir die unbekannte Vergangenheit $t < 1$ durch $x_t = x_{t-1} = ... = 0$ ersetzen und somit gilt:

$$E[E_n| - \infty \ldots, x_0, \ldots, x_n] = 0.$$

Für den Modellparameter nehmen wir die Schätzung $\widehat{\beta}_1$. Die 1-Schritt-Vorherage eines $MA(1)$ berechnet sich dann folgendermassen:

$$\widehat{X}_{n+1|1:n} = \sum_{j=0}^{\infty} \widehat{\beta}_1^j (x_{n-j}).$$

6.3 Vohersage mit $MA(q)$-Modell

Für die Vorhersage eines $MA(q)$-Prozesses sind wir mit den gleichen Problemen konfrontiert wie oben beschrieben. Die Vorhersage für $k > q$ sind alle null, aber für k zwischen 0 und q müssen die gleichen Überlegungen wie in 6.2 gemacht werden. Wir werden die Formel an dieser Stelle nicht angeben, sondern geben die Formel für $ARMA(p, q)$ im nächsten Abschnitt an, von der die $MA(q)$-Vorhersage abgeleitet werden kann.

6.4 Vorhersage $ARMA(p, q)$-Modell

Wir betrachten nun den stationären und invertierbaren $ARMA(p, q)$-Prozess

$$\begin{aligned} X_{n+1} &= \alpha_1(X_n) + \cdots + \alpha_p(X_{n+1-p}) + E_{n+1} + \beta_1 E_n + \cdots + \beta_q E_{n+1-q} \\ &= \sum_{i=1}^{p} \alpha_i (X_{n+1-i}) + E_{n+1} + \sum_{j=1}^{q} \beta_j E_{n+1-j}. \end{aligned}$$

Wir stellen uns wieder auf den Standpunkt, dass wir die gesamte Vergangenheit $-\infty \ldots, x_0, \ldots, x_n$ kennen und erhalten dadurch die 1-Schritt-Vorhersage

$$\begin{aligned} \widehat{X}_{n+1|-\infty:n} &= E[E_n| - \infty \ldots, x_0, \ldots, x_n] \\ &= \sum_{i=1}^{p} \alpha_i E[E_{n+1-i}| - \infty \ldots, x_0, \ldots, x_n] + E[E_{n+1}| - \infty \ldots, x_0, \ldots, x_n] \\ &\quad + \sum_{j=1}^{q} \beta_j E[E_n + 1 - j| - \infty \ldots, x_0, \ldots, x_n] \\ &= \sum_{i=1}^{p} \alpha_i (x_{n+1-i}) + 0 + \sum_{j=1}^{q} \beta_j E[E_{n+1-j}| - \infty \ldots, x_0, \ldots, x_n]. \end{aligned}$$

Für die k-Schritt-Vorhersage erhalten wir das folgende rekursive Schema

$$\begin{aligned}\widehat{X}_{n+k|-\infty:n} = &+ \sum_{i=1}^{p} \alpha_i E[E_{n+k-i}| - \infty \ldots, x_0, \ldots, x_n] \\ &+ \sum_{j=1}^{q} \beta_j E[E_n + k - j| - \infty \ldots, x_0, \ldots, x_n],\end{aligned} \tag{39}$$

wobei

$$\text{AR-Teil: } E[E_n| - \infty \ldots, x_0, \ldots, x_n] = \begin{cases} x_t & \text{falls } t \leqslant n \\ \widehat{X}_{t|1:n} & \text{falls } t \geqslant n \end{cases} \tag{40}$$

$$\text{MA-Teil: } E[E_n| - \infty \ldots, x_0, \ldots, x_n] = \begin{cases} E[E_t| - \infty \ldots, x_0, \ldots, x_n] & \text{falls } t \leqslant n \\ 0 & \text{falls } t \geqslant n. \end{cases} \tag{41}$$

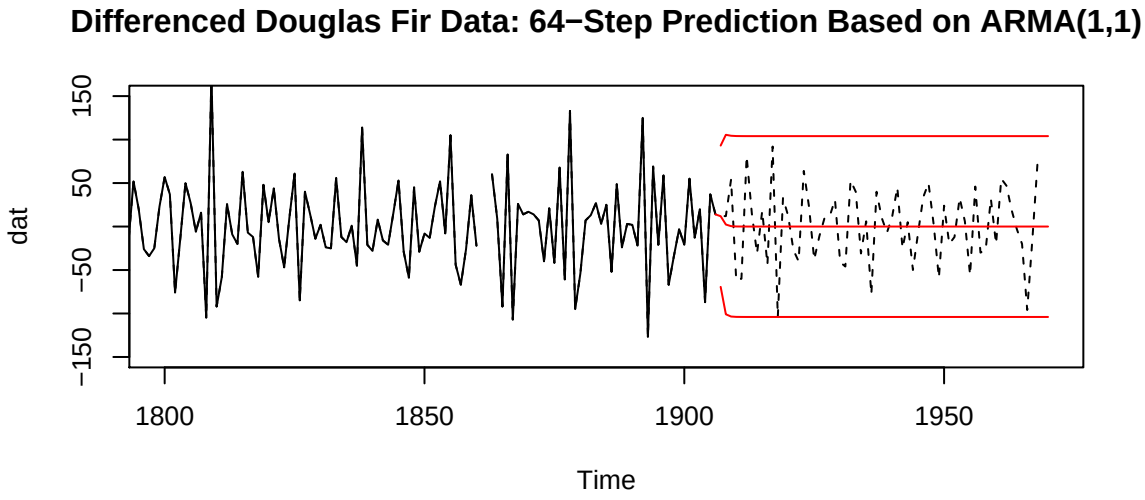
Der Term $E[E_t| - \infty \ldots, x_0, \ldots, x_n]$ wird dann wie in 6.2 bestimmt und für die Modellparameter setzt man die Schätzungen ein. Damit können wir eine beliebige Vorhersage von einem $ARMA(p, q)$ generieren. Diese Methode heisst Box-Jenkins-Methode und ist nach den beiden Forschern benannt, welche dieses Verfahren das erste Mal vorgeschlagen haben.

Jetzt zeigen wir dieses Vorgehen anhand eines praktischen Beispiels in **R**, wofür wir lediglich die Funktionen `arima()` und `predict()` benötigen.

Parktisches Beispiel:

Wir betrachten hier die Breite von Jahresringe einer Föhre über zwsichen 1107 bis 1964. Da die Daten nicht stationär sind, nehmen wir die Differenzen und modellieren diese. Ein $ARMA(1, 1)$ erscheint angemessen. Wir haben die letzten 64 Beobachtungen zur Seite gelegt, so dass wir unsere Vorhersagen überprüfen können. Dann wird das Modell angepasst und man erhällt die Box-Jenkins Vorhersagen. Das Ergebnis, mit einem 95% Prognoseintervall, wird unten gezeigt.

```
fit <-  arima( dat, order= c(1,0,1) )
fca <- predict( fit, n.ahead = 64 )
```



Wir beobachten, dass die Prognose exponentiell schnell auf null abfällt. Tatsächlich sind die Differenzen im gesamten Vorhersagebereich jedoch verschieden von null. Zudem werden alle Beobachtungen bis und mit der Ersten für die Vorhersage verwendet. Wiederum , das ARMA-Modell vereint die Eigenschaften von reinen AR- und MA-Prozessen.

6.5 Vorhersage von Saison-Trend-Zeitreihen

Falls eine Zeitreihe entweder einen Trend und/oder einen Saisoneffekt aufweist, so können wir nicht einfach ein $AR(p)$-Modell anpassen und damit Vorhersagen erzeugen. Zur deskriptiven Analyse zerlegen wir solche Zeitreihen meistens in Trend, Saisoneffekt und stationären Rest. Auf dieser Basis kann man dann auch Vorhersagen erzeugen. Das Paradigma ist wie folgt:

Trend Wir nehmen an, dass der Trend glatt verläuft, sich über die Zeit also nur relativ langsam ändert. Zur Extrapolation wird ein linearer Ansatz empfohlen (siehe Details unten).

Saisoneffekt Wir gehen davon aus, dass der Saisoneffekt sich zeitlich nicht verändert. Er wird also als gleich bleibend in die Zukunft extrapoliert.

Stationärer Rest Für den stationären Restterm der Reihe können wir ein AR(p)-Modell anpassen. Mit diesem werden die Vorhersagen dieses Teils bestimmt.

Wir müssen hier nun also besprechen, wie der Trend zeitlich extrapoliert wird. Wir wollen dies am Beispiel der Zeitreihe „Inverkehrssetzungen im Strassenverkehr in der Schweiz, 2000-2010“ illustrieren. Ziel soll es sein, den Verlauf der Reihe über die nächsten 2 Jahre, bzw. 24 Datenpunkte, vorherzugsagen. Für den Trend gilt die folgende Empfehlung:

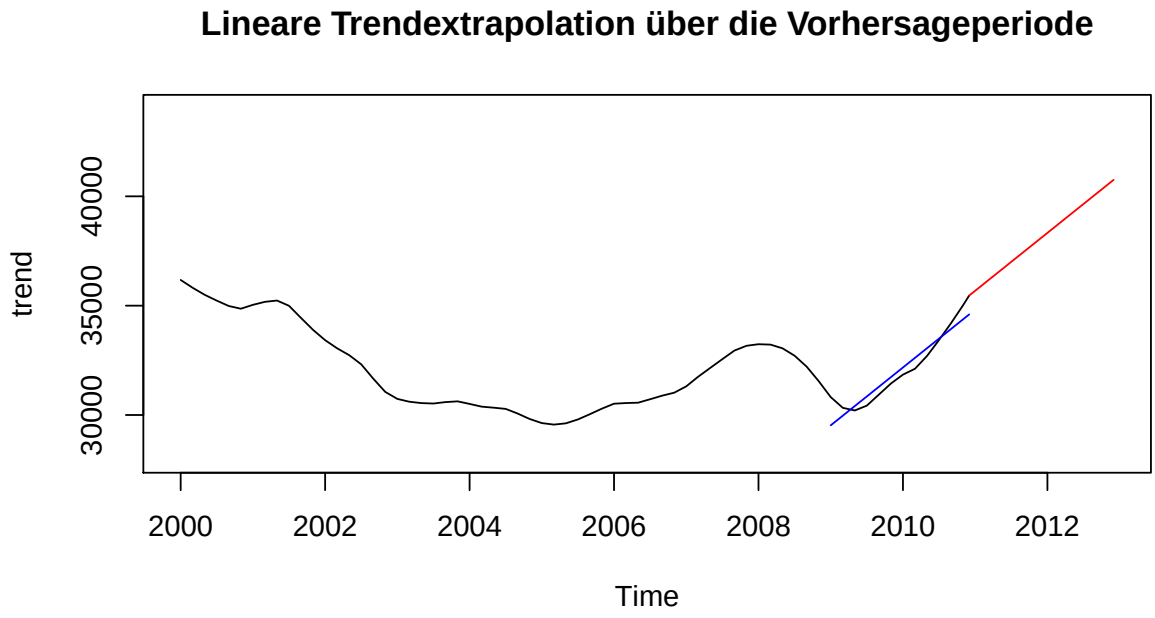
Man lege eine KQ-Gerade in die Vergangenheit des Trends. Die Zeitdauer für die Anpassung soll dabei gleich sein wie der Vorhersagehorizont, hier also 24 Datenpunkte. Mit

anderen Worten: wir bestimmen den durchschnittlichen Trend über die letzten 2 Jahre (blaue Gerade). Für die Vorhersage gehen wir vom letzten beobachteten Trend-Wert aus und extrapolieren mit der gefundenen Steigung linear (rote Gerade).

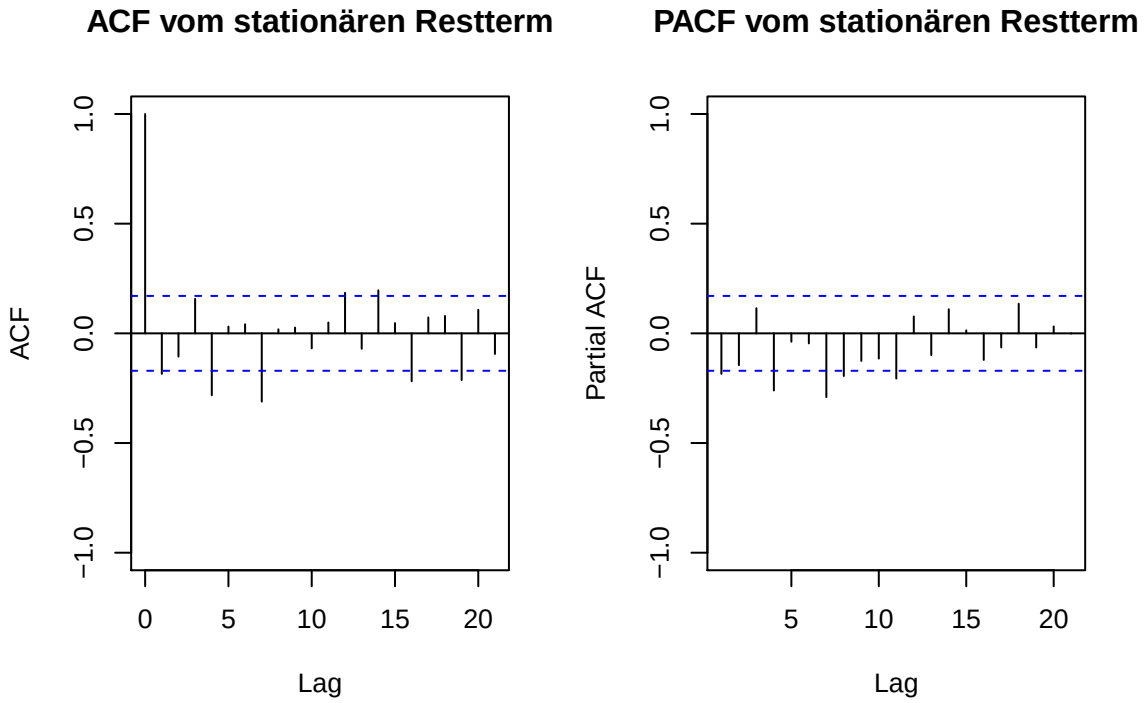
Eine so erzeugte Trend-Extrapolation ist stets als Vorschlag zu betrachten. Falls Fachwissen einen anderen Verlauf suggeriert, ist es durchaus zulässig, eine manuelle Korrektur vorzunehmen. In der Praxis ist es einfach wichtig, dass man klar deklariert, wie die Vorhersage zu Stande gekommen ist.

```
# Daten laden, STL, Trend extrahieren
load("...inverkehr.rda")
```

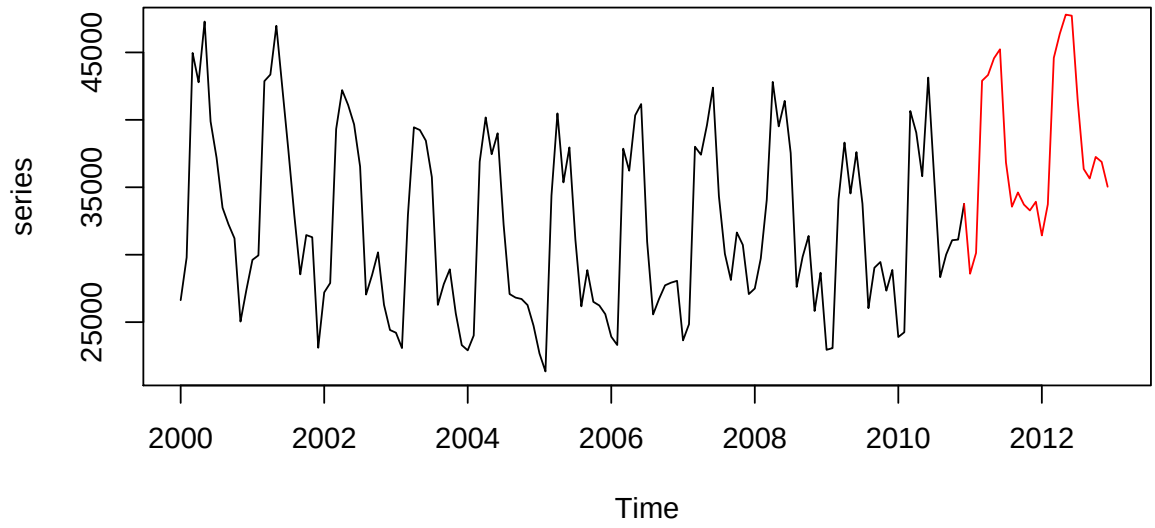
```
> series <- ts(inverkehr, start=c(2000,1), freq=12)
> fit <- stl(series,s.window='periodic')
> trend <- fit$time.series[,2]
>
> # KQ-Anpassung \"uber die letzten 2 Jahre
> yy <- window(trend, c(2009,1),c(2010,12))
> xx <- as.numeric(time(yy))
> reg <- lm(yy~xx)
> reg.ts <- ts( fitted(reg), start = xx[1], freq = 12 )
>
> # Trend-Extrapolation
> kk <- 24
> trend.ex <- rev(trend)[1]+((0:kk)/12)*coef(reg)[2]
> trend.ex.ts<- ts( trend.ex, start = rev(xx)[1], freq = 12)
```



Die Vorhersage für den Saisoneffekt ist trivial, weil wir diesen als gleich bleibend annehmen. Wir kümmern uns nun noch um den stationären Restterm: aus der PACF drängt sich keine bestimmte Ordnung p für das autoregressive Modell auf, siehe unten:



Wir bestimmen eine geeignete Ordnung deshalb mit dem AIC-Kriterium. Es ergibt sich $p = 18$. Mit der Prozedur `predict()` erzeugen wir dann eine 24-Schritt-Prognose des stationären Restterm. Zuletzt müssen wir die Vorhersagen für die drei Teile (Trend, Saison und Rest) noch zusammenfügen, um eine 2-Jahres-Prognose für die Inverkehrssetzungen zu erhalten.



Auf den ersten Blick sieht die erzeugte Vorhersage plausibel aus. Ob sie sich bewahrheiten wird, kann aber erst die Zukunft zeigen. Hinweis: sie finden die neusten Daten beim ASTRA.

Das vorgeschlagene Verfahren zur Vorhersage von zerlegten Zeitreihen funktioniert also wie gewünscht. Es bietet sich insbesondere dann an, wenn man die Trend-Extrapolation noch händisch korrigieren möchte. Das kann durchaus sinnvoll sein, wenn man z.B. wegen einer sich abzeichnenden Rezession in den Folgejahren wieder mit tieferen Zahlen rechnet. Für eine blosse out-of-the-box-Vorhersage, also ein genaues Anwenden der beschriebenen Methode, mag die Programmierarbeit etwas aufwendig sein. Bequemer geht es mit Exponential Smoothing, das wir nun besprechen.

6.6 Exponential Smoothing (Exponetielles Glätten)

In diesem Skript werden die praktischen Beispiele zum Exponentiellen Glätten mit der **R**-Funktion `HoltWinters()` aus dem `stats`-Package demonstriert. Im Unterricht wird das praktische Vorgehen des Exponentiellen Glättens jedoch mit Hilfe der **R**-Funktionen aus dem `forecast`-Package veranschaulicht und erklärt.

6.6.1 Einfaches Exponential Smoothing

Das Ziel in diesem Kapitel ist weiterhin die Prognose von zukünftigen Werten $\widehat{X}_{n+k}$ für eine stationäre Zeitreihe X_t aufgrund von Beobachtungen $\{x_1, \ldots, x_n\}$.Stationarität bedeutet hier, dass der Erwartungswert $E[X_t] = \mu$ ist, d.h. konstant über die Zeit. Hingegen ist der bedingte Erwartungswert, gegeben die Beobachtungen in der unmittelbaren Vergangenheit wegen der Abhängigkeit nicht immer gleich. Wir schreiben also $E[X_t|X_{t-1}, X_{t-2}, \ldots] = \mu_t$. Im Exponential Smoothing nennt man μ_t den Level der Reihe zur Zeit t. Man geht dann davon aus, dass sich die aktuelle Beobachtung aus Erwartungswert plus einer Abweichung/Innovation E_t ergibt:

$$X_t = \mu_t + E_t. \tag{42}$$

Wir verwenden im folgenden dieselbe Notation wie in **R** und schreiben a_t für die Schätzung des aktuellen Levels μ_t. Da es wegen der Stationarität in der vorliegenden Reihe weder Trend noch Saisoneffekt gibt, ist es intuitiv, für die Schätzung ein gewichtetes Mittel aus der aktuellen Beobachtung und dem vorangehenden Level zu verwenden:

$$a_t = \alpha x_t + (1 - \alpha)a_{t-1}, \text{ mit } 0 < \alpha < 1. \tag{43}$$

Ganz offensichtlich ist α ein Glättungsparameter: ist er nahe bei 1, so glättet man nur wenig, und die Level-Reihe a_t trackt die Zeitreihe x_t sehr nahe. Dies ist dann sinnvoll, wenn das Signal im Vergleich zur Innovation stark ist. Umgekehrt glättet ein α-Wert nahe bei null sehr stark, und dementsprechend hat die aktuelle Beobachtung x_t nur wenig Einfluss. Dies wäre der zu wählende Weg, wenn das Signal in der Zeitreihe schwach, bzw. die Innovation stark ist. Ein typischer Default-Wert ist $\alpha = 0.2$. Er begründet sich dadurch, dass für die meisten Zeitreihen die Level-Änderung zwischen t und $t - 1$ kleiner ist als σ_E^2. In **R** ist auch ein Algorithmus zur Schätzung von *alpha* implementiert, siehe unten.

Gemäss dem Modell der exponentiellen Glättung, für stationäre Reihen ohne Trend und Saisoneffekt, ist die beste Vorhersage zur Zeit $t = n$, und zwar für jeden Vorhersagehorizont k, gegeben durch die Level-Schätzung:

$$\widehat{X}_{n+k} = a_n \text{ für alle } k = 1, 2, \ldots \tag{44}$$

Wir können (und werden) die obige Definitionsgleichung für den Level nun auf zwei verschiedene Arten anders darstellen. Dies ist instruktiv, da es vertieft zeigt, wie Exponential Smoothing funktioniert. Zuerst schreiben wir den Level zur Zeit t als Summe von a_{t-1} und dem 1-Schritt-Prognosefehler. Man nennt die auch die Update-Formel:

$$a_t = \alpha(x_t - a_{t-1}) + a_{t-1}. \tag{45}$$

Durch wiederholtes Rückwärtseinsetzen erhält man daraus die folgende Formel:

$$a_t = \alpha x_t + \alpha(1 - \alpha)x_{t-1} + \alpha(1 - \alpha)^2 x_{t-2} + \ldots. \tag{46}$$

Auf diese Art und Weise geschrieben ist a_t nichts anderes als eine Linearkombination der aktuellen, sowie aller vergangenen Beobachtungen der Zeitreihe. Da $\alpha \in (0,1)$ gilt, sind die Gewichte $\alpha(1-\alpha)^i$ exponentiell abfallend, d.h. weiter zurückliegende Beobachtungen erhalten weniger und weniger Gewicht.

Zusätzlich definieren die Gewichte eine geometrische Reihe. Wenn man die Terme von $i = 0$ bis nach ∞ aufsummiert, so erhält man den Wert 1. In Realität können wir natürlich nicht über unendlich viele Terme summieren. Wir können dies aber vermeiden, indem wir das Rückwärtseinsetzen zum Zeitpunkt $t = 1$ abbrechen, und $a_1 = x_1$ setzen.

Hat man den Glättungsparameter α einmal festgelegt, so kann man mit der Update-Formel und dem Setzen von $a_1 = x_1$ den Level a_t für alle Zeiten $t = 2, 3, \ldots$ bestimmen. Als 1-Schritt-Vorhersagefehler ergibt sich:

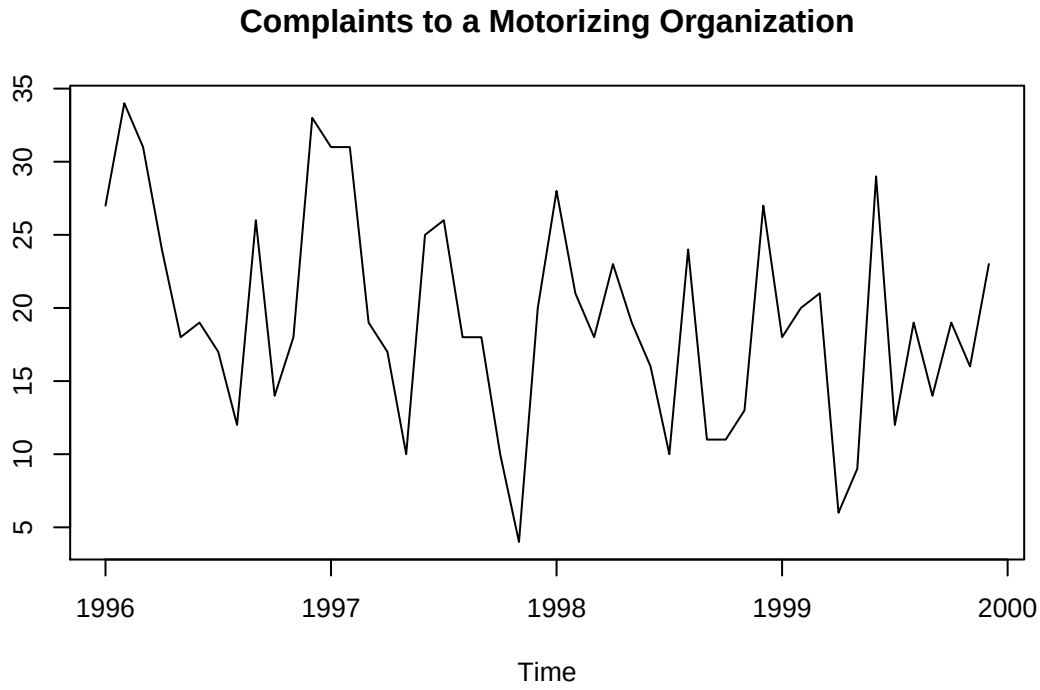
$$e_t = x_t - \widehat{x}_{t,1:(t-1)} = x_t - a_{t-1}. \tag{47}$$

Man könnte nun versuchen, den Glättungsparameter α so zu wählen, dass die Quadratsumme der Vorhersagefehler innerhalb der beobachteten Reihe so klein wie möglich ist. In **R** ist genau dieses Vorgehen als Default implementiert, d.h. **R** bestimmt eine Schätzung von α durch Minimierung von $SS1PE$:

$$SS1PE = \sum_{t=2}^{n} e_t^2. \tag{48}$$

In den meisten Fällen ergibt sich so eine gute, zweckmässige Schätzung, die einem a priori gewählten Wert wie z.B $\alpha = 0.2$ überlegen ist.

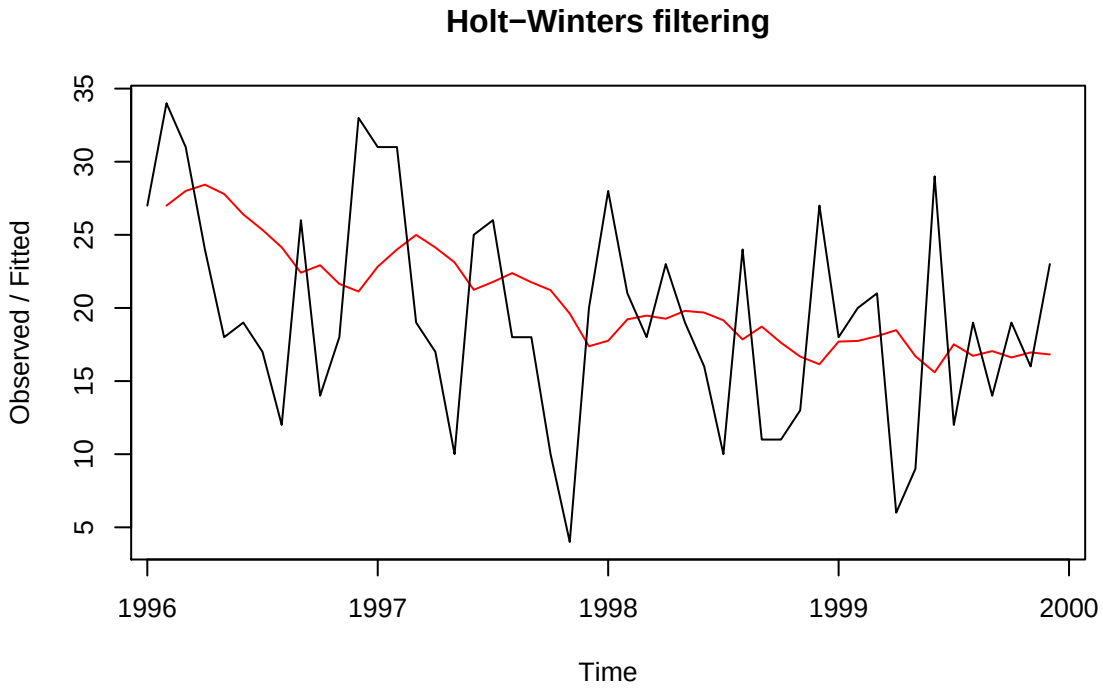
Wir verwenden hier eine Zeitreihe, welche die Anzahl Beschwerdebriefe beschreibt, welche bei einer Motorizing Organization (ein Analogon zum TCS im angelsächsischen Raum) eingegangen sind. Es sind monatliche Beobachtungen für die Jahre $1996 - 1999$ vorhanden. Nun, am Anfang des Jahres 2000, möchte die Organisation den aktuellen Level bestimmen, und auch herausfinden, wie sich dieser im Lauf der Vergangenheit entwickelt hat. Wir importieren zuerst die Daten und erzeugen einen Zeitreihenplot:



Die Zeitreihe ist mit 48 Beobachtungen relative kurz. Es gibt keine Hinweise auf das Vorliegen eines Trends und/oder Saisoneffekts. Wir können also die einfache exponentielle Glättung verwenden. Sie ist in der **R**-Funktion `HoltWinters()` implementiert. Achtung: `HoltWinters()` offeriert mehr als nur die einfache exponentielle Glättung, daher müssen wir die Argumente `beta=FALSE` und `gamma=FALSE` setzen. Falls wir Argument `alpha` leer lassen, d.h. kein α spezifizieren, so wird dieses durch Minimierung von $SS1PE$ geschätzt.

```
> fit <- HoltWinters(cmpl, beta=FALSE, gamma=FALSE); fit

Holt-Winters exponential smoothing without trend and without seasonal component.

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

Smoothing parameters:
 alpha: 0.1429622
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 17.70343
```

```
> plot(fit)
```



Aus dem Output ist sofort ersichtlich, dass der Level der Reihe im Dezember 1999 gleich 17.70 war. Dies entspricht dem Wert a_{48}. Ebenso zeigt der Output, dass der optimale Glättungsparameter α gemäss dem $SS1PE$-Kriterium $0.143war$. Die Summe der quadrierten (Insample-)Vorhersagefehler ergab sich zu 2502. Logischerweise würde jeder andere Wert für α zu einem grösseren Vorhersagefehler führen. Mit dem `plot()` Befehl können wir das Ergebnis bequem visualisieren.

6.6.2 Die Holt-Winters-Methode

Die oben beschrieben, einfache exponentielle Glättung kann verallgemeinert warden, so dass sie auch für die Vorhersage von Zeitreihen benutzt werden kann, welche sowohl Trend und/oder saisonale Effekte zeigen. Da ja in der Praxis die wenigsten Zeitreihen stationär sind, ist das natürlich bequem. Die Holt-Winters-Methode baut auf den folgenden Formeln auf:

$$a_t = \alpha(x_t - s_{t-p}) + (1-\alpha)(a_{t-1} + b_{t-1}) \tag{49}$$
$$\beta_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1} \tag{50}$$
$$s_t = \gamma(x_t - a_t) + (1-\gamma)s_{t-p} \tag{51}$$

In dieser Definition ist a_t man die Steigung, und s_t ist der Saisoneffekt. Anstatt nur einen einzigen gibt es hier nun drei Glättungsparameter α, β, γ, welche die Variabilität von Level,

Steigung und Saisoneffekt kontrollieren. Die drei Gleichungen sind wie folgt zu interpretieren:

- Die erste Update-Gleichung bildet ein gewichtetes Mittel zwischen der letzten, saisonbereinigten Beobachtung mit der 1-Schritt-Vorhersage für den Level, die man aus der Kombination von Level zur Zeit $t-$ und Steigung zur Zeit t erhält.
- Die zweite Update-Gleichung bildet ein gewichtetes Mittel zwischen der Differenz des aktuellen und des vorherigen Levels mit der Steigung vom Zeitpunkt $t-1$.Diese Gleichung kann nur berechnet werden, falls a_t bereits berechnet wurde.
- Die dritte Update-Gleichung kümmert sich um den Saisoneffekt. Es wird ein gewichtetes Mittel der Differenz zwischen Beobachtung und Level und dem "alten", d.h. eine Periode zurückliegenden Saisoneffekt gebildet.

Ohne Vorwissen, und ohne (wie in **R** vorhanden) Schätzprozedur für die Parameter ist ein sinnvoller Default wiederum $\alpha, \beta, \gamma = 0.2$. Zudem sind auch Startwerte für die rekursiven Update-Gleichungen notwendig. Hierbei wähle man $a_1 = x_1$ und $\beta_1 = 0$. Die Saisoneffekte $s_1, s_2, \ldots, s_p$ setzt man entweder alle auf null, oder man wählt entsprechende Durchschnittswerte. Die **R**-Funktion `HoltWinters()` beziehtihre Startwerte aus einer Zerlegung mit `decompose()`, die drei Glättungsparameter werden erneut durch Minimierung von $SS1PE$ bestimmt. Der häufig am meisten interessierende Aspekt einer Holt-Winters-Anpassung ist die Vorhersage von zukünftigen Werten. Formelmässig lautet die k-Schritt-Prognose zum Zeitpunkt $t = n$:

$$\widehat{X}_{n+k} = a_n + k \cdot b_n + s_{n+k-p}, \tag{52}$$

d.h. die Prognose setzt sich zusammen aus dem aktuellen Level, einer linearen Extrapolation des aktuellen Trends und gleichbleibendem Saisoneffekt.

Praxisbeispiel

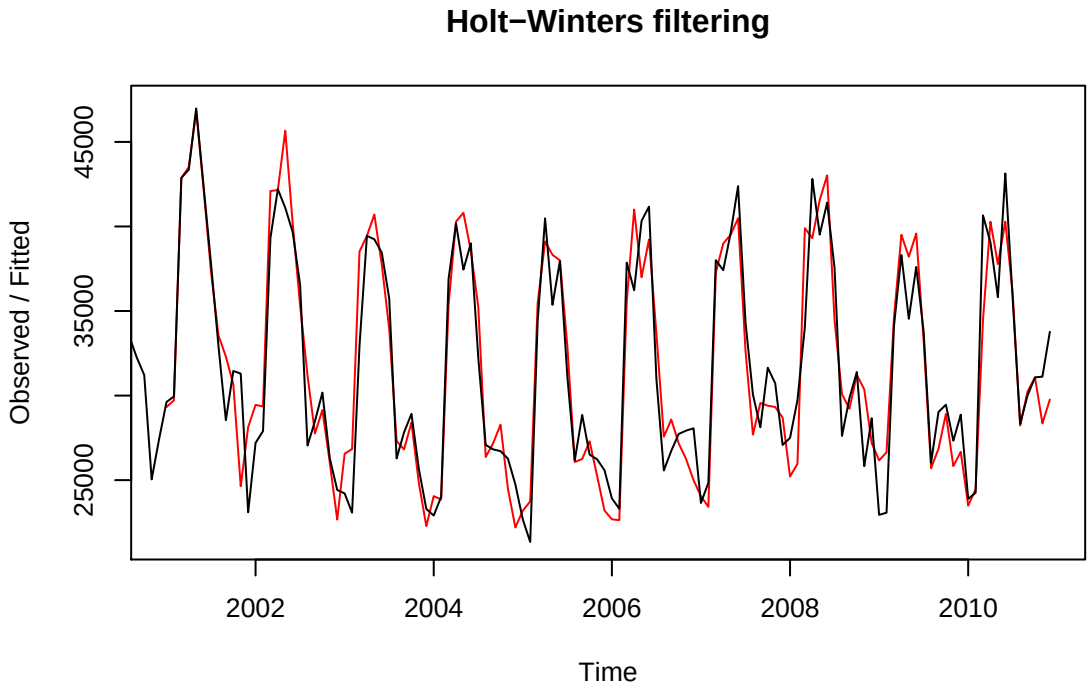
Um einen guten Vergleich zur manuellen Methode, d.h. der Extrapolation von Trend, Saison und stationärem Rest zu erhalten, versuchen wir hier die Reihe der Inverkehrssetzungen auch noch mit der Holt-Winters-Methode vorher. Wenn wir die Glättungsparameter mit dem $SS1PE$-Kriterium schätzen, so ist der **R**-Befehl sehr, sehr einfach:

```
> fit.hw <- HoltWinters(inverkehr)
> fit.hw

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = inverkehr)
```

```
Smoothing parameters:
 alpha: 0.1803887
 beta : 0
 gamma: 0.5938609

Coefficients:
           [,1]
a   34443.85535
b      10.67118
s1  -8414.96860
s2  -7832.49696
s3   5336.09158
s4   6400.49274
s5   3750.89386
s6   8940.51075
s7   2604.38876
s8  -4919.72343
s9  -3100.74972
s10 -2126.73545
s11 -3509.82096
s12 -2004.77815
```

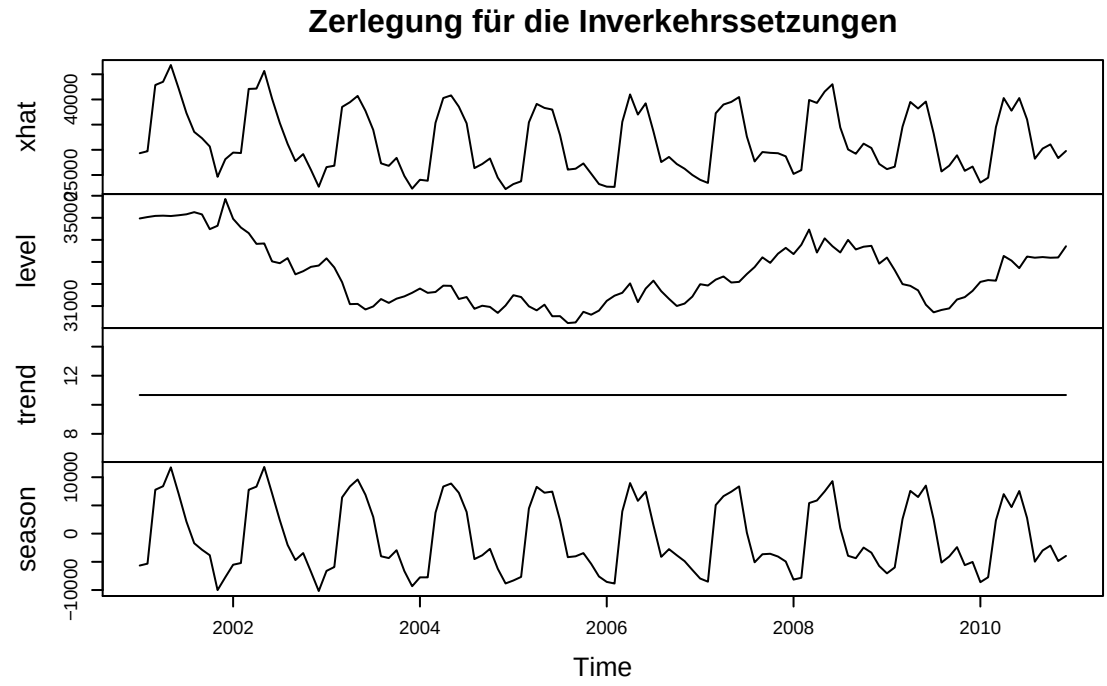
Im Output erhalten wir die Schätzungen für die Glättungsparamter α, β und γ, sowie Level, Trend und die Saisoneffekte zum Ende der Zeitreihe. Achtung, während erstere (d.h. $\widehat{\alpha}, \widehat{\beta}, \widehat{\gamma}$) für die ganze Reihe gelten, sind letztere nur für den Zeitpunkt $t = n$ gültig, nicht jedoch für die ganze Zeitreihe. Wir betrachten zuerst einmal den Verlauf der angepassten Werte im Vergleich zur Originalreihe:

```
> plot(fit.hw)
```



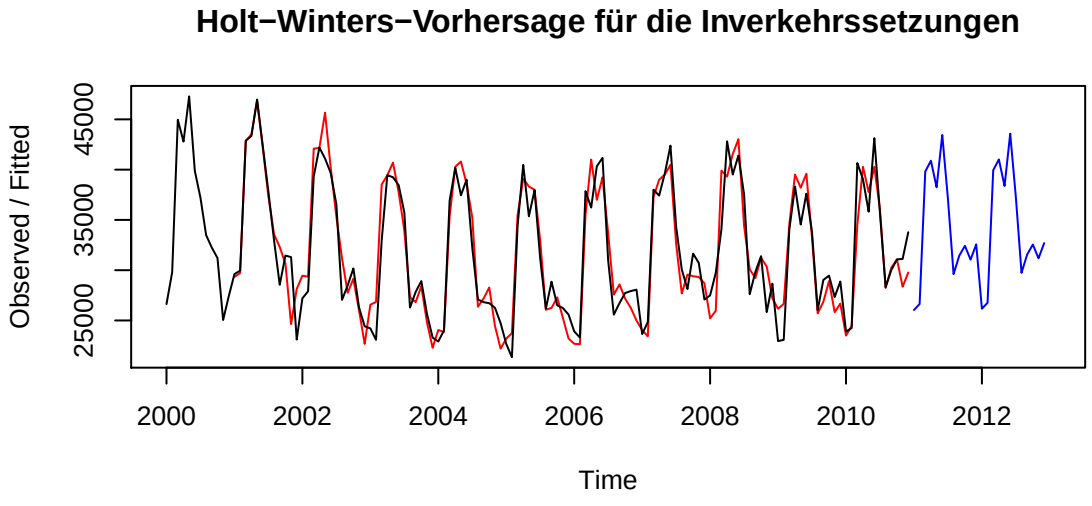
Ebenso instruktiv ist es, den Verlauf von Level, Trend und Saisoneffekt gegen die Zeit aufzutragen. Dies erreicht man durch:

```
> plot(fit.hw$fitted, main = "")
> title('Zerlegung für die Inverkehrssetzungen')
```

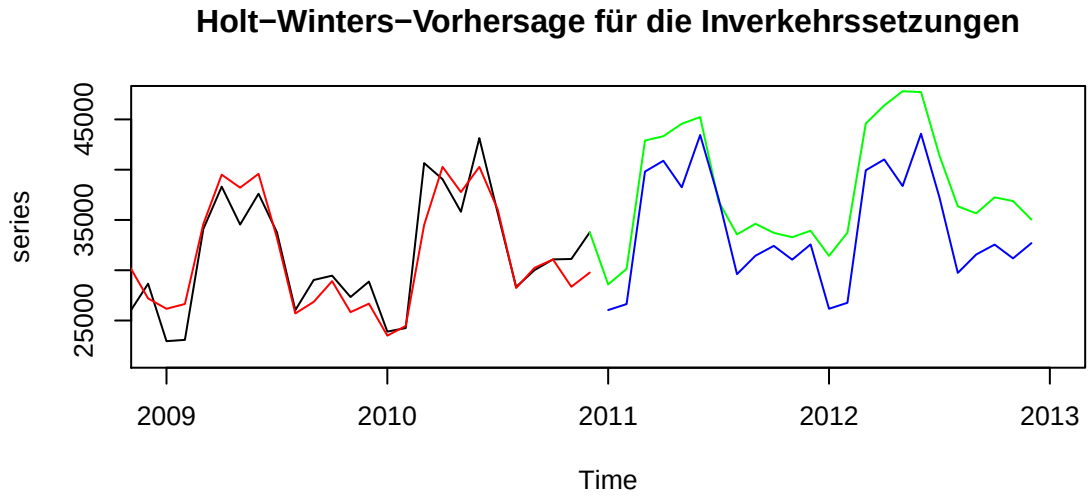


Die letzten Schätzungen von Level, Trend und Saisoneffekt (d.h. jene zur Zeit $t = n$) werden nun auch für die Vorhersage benützt. Wie schon zuvor erzeugen wir eine 2-Jahres-Vorhersage, d.h. extrapolieren die nächsten 24 Beobachtungen:

```
> plot(fit.hw, xlim=c(2000, 2013), main = "")
> title('Holt-Winters-Vorhersage für die Inverkehrssetzungen')
> lines(predict(fit.hw, n.ahead=24), col="blue")
```



Wir beschliessen unsere Analyse mit einem Vergleich der Holt-Winters-Vorhersagen und den manuell durch Zerlegung erzeugten. Dies sieht wie folgt aus:



Die manuelle Vorhersage ist dunkelgrün dargestellt, diejenige von Holt-Winters mit blau. Schwarz ist die Originalreihe, welche nur bis Ende 2010 bekannt ist. In roter Farbe ist der Insample-Fit der Holt-Winters-Methode aufgetragen. Es ist deutlich erkennbar, dass Holt-Winters keine Zunahme (des Trends) vorhersagt. Dies darum, weil sich in der (ganzen) Vergangenheit der Reihe Zu- und Abnahmen des Trends abgewechselt haben. Bei der manuellen Dekomposition und der daraus erzeugten Vorhersage haben wir uns bezüglich der Trend-Extrapolation hingegen nur auf die letzten 2 Jahre abgestützt.

7 Modelle und Vorhersage für nicht-stationäre Zeitreihen

Wie wir ja bereits wissen, sind die meisten Zeitreihen aus der Praxis nicht stationär, sondern weisen einen Trend und/oder eine Saisonalität auf. Während wir gelernt haben, wie wir diese entfernen und mit einem Zeitreihenmodell den stationären Rest erklären können, gibt es weitere Modelle, die den Trend und Saison direkt beinhalten. Normalerweise geschieht die Zerlegung in Trend, Saison und Resterm in diesen Modellen nicht immer ganz transparent, aber sie sind ein *all-in-one* Ansatz und erlauben uns somit sehr bequem Vorhersagen zu erzeugen. Zudem ermöglichen sie eine auf dem AIC-Kriterium basierte Entscheidung um den richtigen Anteil für den Trend und Saisonalität zu bestimmen.

7.1 $ARIMA$-Modelle

$ARIMA$ werden dazu benutzt, Zeitreihen zu beschreiben die einen Trend aufweisen, der sich durch Differenzieren entfernen lässt und die Differenzen durch ein $ARMA(p,q)$-Modell beschrieben werden können. Die Definition eines $ARIMA(p,d,q)$-Modells ergibt sich daher auf natürliche Weise:

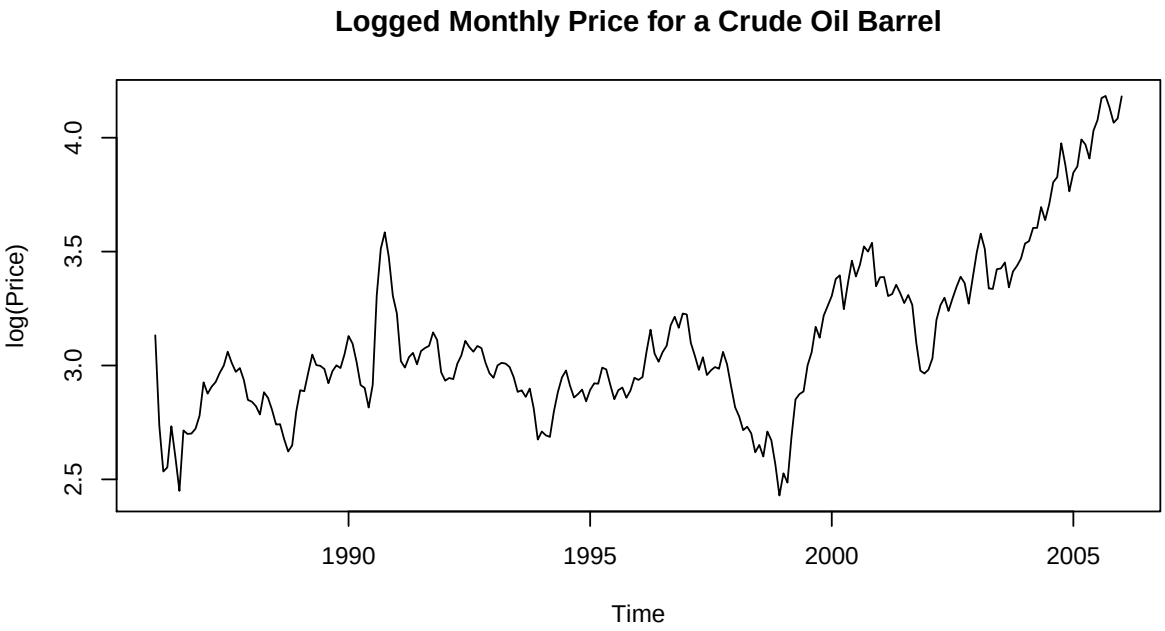
Definition: Eine Zeitreihe X_t kann als $ARIMA(p,d,q)$-Modell geschrieben werden, falls die Differenz d-ter Ordnung von X_t ein $ARMA(p,q)$-Prozess ist. Dazu führen wir

$$Y_t = (1-B)^d X_t$$

ein, wobei B der Backshift-Operator ist. Dann können wir den $ARIMA$-Prozess mit dem charakteristischen Polynomen $\Phi(\cdot)$, Beitrag des AR-Prozesses und $\Theta(\cdot)$, Beitrag des MA-Prozesses, angeben:

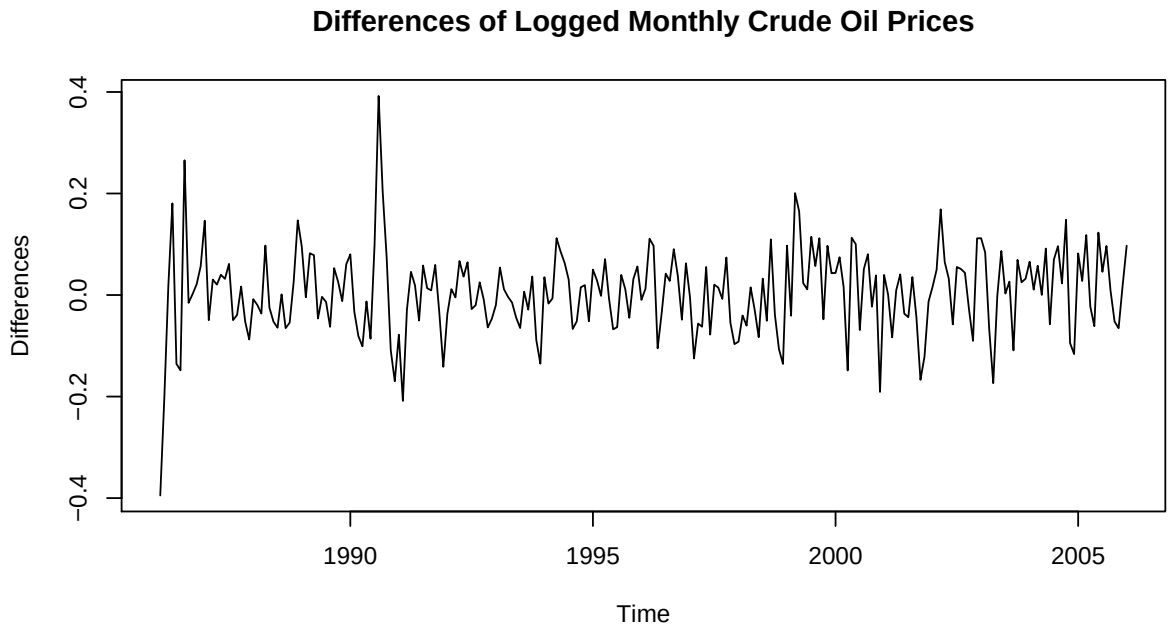
$$\begin{aligned} \Phi(B)Y_t &= \Theta(B)E_t \\ \Phi(B)(1-B)^d X_t &= \Theta(B)E_t \end{aligned} \tag{53}$$

Solche Zeitreihen treten in Praxis auf. Als Beispiel schauen wir uns dazu die Zeitreihe des monatlichen Ölpreises (in US \$) eines Barrels Rohöl zwischen Januar 1986 und Januar 2006 an.



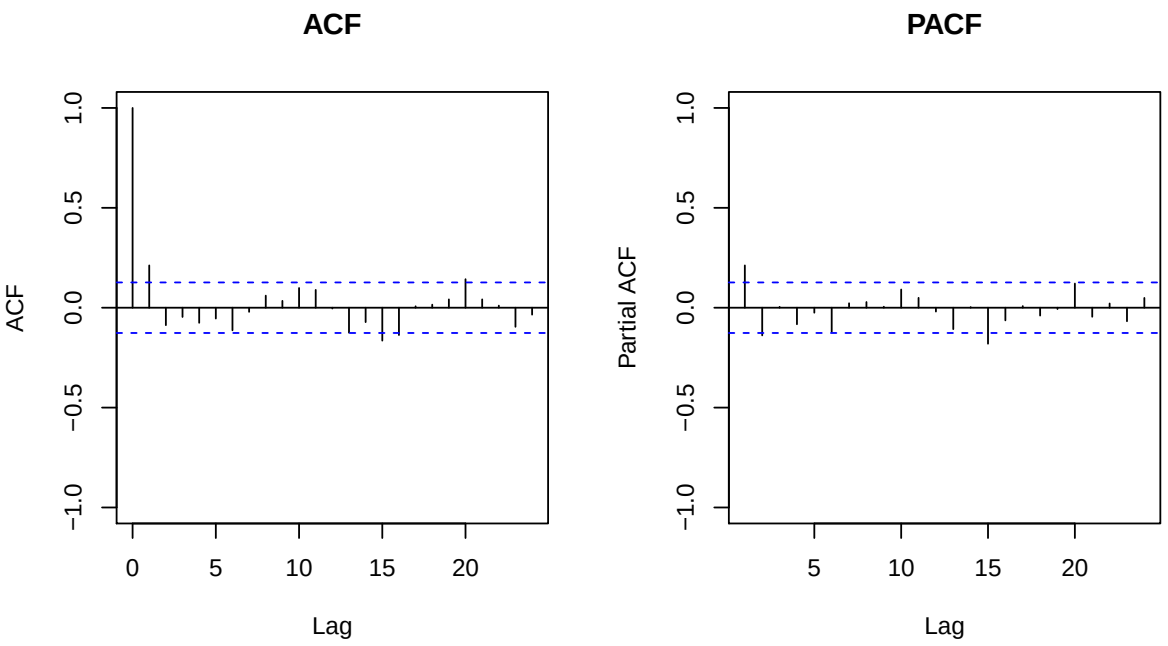
Die Reihe weist keine erkennbare Saisonalität auf, dafür einen klaren Trend und ist somit nicht stationär. Wir bilden die Differenz erster Ordnung (d.h. $d = 1$) und schauen uns an, ob die Differenzen stationär sind.

```
> dlop <- diff(lop,lag = 1)
> plot(dlop, ylab="Differences")
> title("Differences of Logged Monthly Crude Oil Prices")
```



Das Differenzieren hat den Trend erfolgreich entfernt. ACF und PACF zeigen das die Differenzen noch eine serielle Abhängigkeitsstruktur aufweisen. Wir sehen einen Abfall in der ACF nacht dem Lag 1 und in der PACF nach dem Lag 1 oder 2. Dies lässt uns vermuten, dass der logarithmierte Ölpreis von einem $ARIMA(1,1,1)$- oder $ARIMA(2,1,1)$-Prozess stammt. Allerdings hat, etwas überraschend, ein $ARIMA(1,1,2)$ die beste Performance.

```
> par(mfrow=c(1,2))
> acf(as.numeric( dlop) , main="ACF", ylim = c(-1,1), lag.max=24)
> pacf(as.numeric( dlop) , main="PACF", ylim= c(-1,1), lag.max=24)
```



Die Anpassung des Modells an die Zeitreihen (model fitting) kann mit der **R**-Funnktion `arima()` gemacht werden, welche die Koeffizienten mit der ML-Methode schätzt, wobei die Startwerte aus einer CSS-Schätzung stammen. Entweder überlassen wir das Differenzieren der `arima`-Funktion:

```
> fit.arima <- arima(lop, order=c(1,1,2))
> fit.arima


Call:
arima(x = lop, order = c(1, 1, 2))

Coefficients:
         ar1      ma1      ma2
      0.8429  -0.5730  -0.3104
s.e.  0.1548   0.1594   0.0675

sigma^2 estimated as 0.006598:  log likelihood = 261.88,  aic = -515.75
```

Oder wir verwenden die differenzierte Zeitreihe des logarithmierten Ölpreises als input und passen ein $ARMA(1,2)$-Modell an. Allerdings, müssen wir dann die Funktion darauf hinweisen (`include.mean = FALSE`), dass kein Mittelwert berücksichtig werden soll. Der Aufruf ist dann folgendermassen:

```
> dlop <- diff(lop,lag = 1)
> fit.arma <- arima(dlop, order=c(1,0,2), include.mean = FALSE)
> fit.arma


Call:
arima(x = dlop, order = c(1, 0, 2), include.mean = FALSE)

Coefficients:
         ar1      ma1      ma2
      0.8429  -0.5730  -0.3104
s.e.  0.1548   0.1594   0.0675

sigma^2 estimated as 0.006598:  log likelihood = 261.88,  aic = -515.75
```

Die Ausgabe entspricht genau der Ausgabe von vorhin. Als nächsten Schritt müssen wir nun mittels einer Residuenanalyse überprüfen, ob die Modellvoraussetzungen erfüllt sind, d.h. die Residuen müssen wie weisses gaussches Rauschen aussehen. Dies ist hier mehr oder weniger der Fall (nicht gezeigt). Zur Entscheidung der Modellordnung kann auch die AIC-Statistik herangezogen werden.

Wir beenden diesen Abschnitt mit einigen Überlegungen zur Modellgleichung. Wir haben: Daher können wir das $ARIMA(1,1,2$-Modell in ein nicht-stationäres $ARMA(1,2)$ umschreiben. Der nicht stationäre Teil ist aufgrund der Lösung des charakteristischen Polynoms für den AR Teil, welche innerhalb des Einheitskreises liegt. Zuletzt noch ein paar zusammenfassende Angaben zu Vorgehen beim Anpassen eines $ARIMA$-Modell an eine Zeitreihe:

1. Geeignete Wahl der Differenzordnung d, normalerweise $d = 1$ oder (in seltenen Fällen) $d = 2$, so dass die resultierende Zeitreihe stationär wird.

2. Analyse von ACF und PACF der differenzierten Zeitreihe. Falls die typischen Anzeichen eines $ARMA$-Prozesses zu erkennen sind, Ordnungswahl von p und q.

3. Modellanpassung mit der `arima()`-Funktion. Das Modell kann direkt auf die originale Zeitreihe angepasst werden. In diesem Fall muss beim Argument `order = c(p,d,q)` die Differenzenordnung d angegeben werden. Falls das Modell auf die differenzierte Zeitreihe angepasst wird, muss `d`$= 0$ und das Argument `include.mean` auf `FALSE` gesetzt werden.

4. Residuenanalyse: Die Residuen müssen sich wie weisses gausssches Rauschen verhalten. Falls mehrere Modell in Frage kommen, sollte man die AIC-Statistik als Entscheidungshilfe verwenden.

7.2 *SARIMA*-Modelle

Nachdem wir nun mit den ARIMA-Modellen vertraut sind, ist es nur natürlich sich zu fragen, ob wir diese Modelle nicht auf saisonale Zeitreihen erweitern können; vor allem da wir ja auch wissen, dass Differenzieren mit einem Lag der gerade der Periode entspricht den saisonalen Effekt beseitigt. Gehen wir davon aus das wir eine Zeitreihe X_t haben mit monatlichen Messwerten. Dann ist die saisonale Komponente von der Zeitreihe

$$Y_t = X_t - X_{t-12} = (1 - B^{12})X_t$$

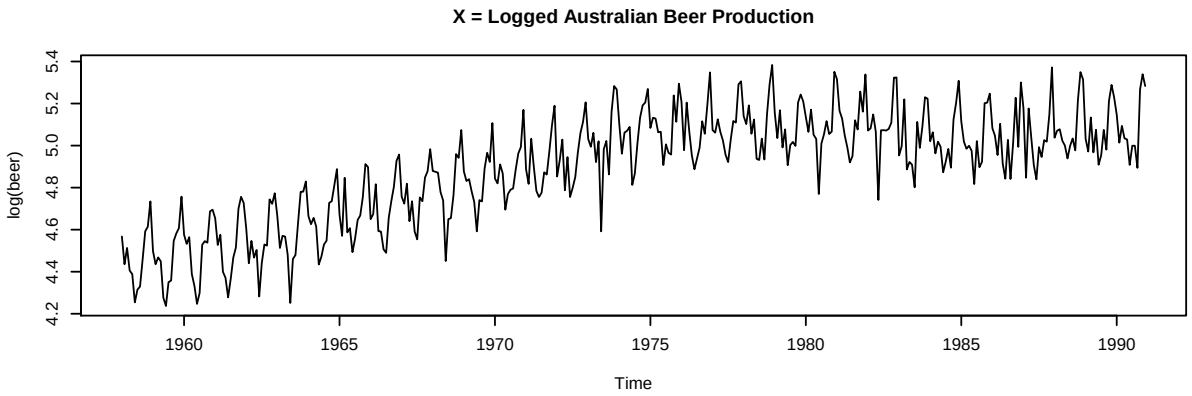
entfernt. Allerdings ist der Mittelwert dieser Zeitreihe Y_t häufig noch nicht konstant, so dass noch eine Differenzierung zum Lag 1 nötig ist um Stationarität zu erreichen:

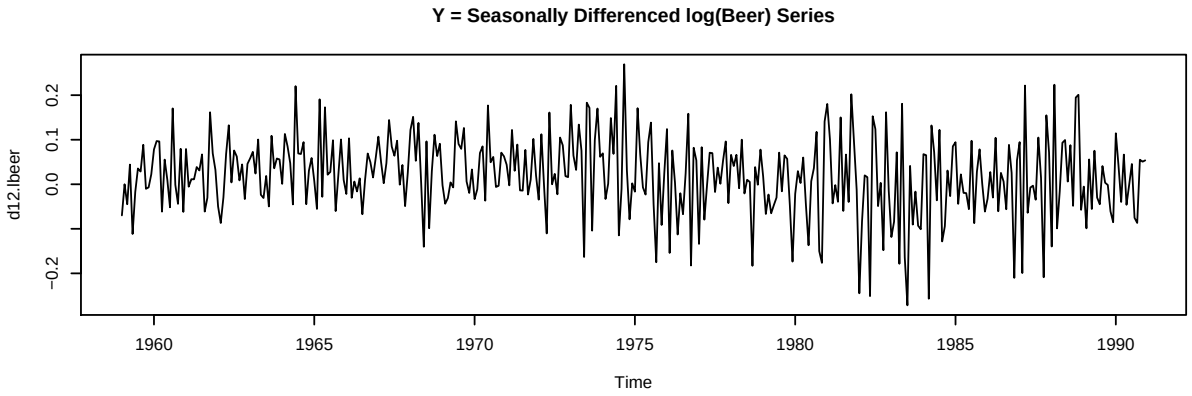
$$Z_t = Y_t - Y_{t-1} = (1 - B)Y_t = (1 - B)(1 - B^{12})X_t = X_t - X_{t-1} - X_{t-12} + X_{t-13}. \quad (54)$$

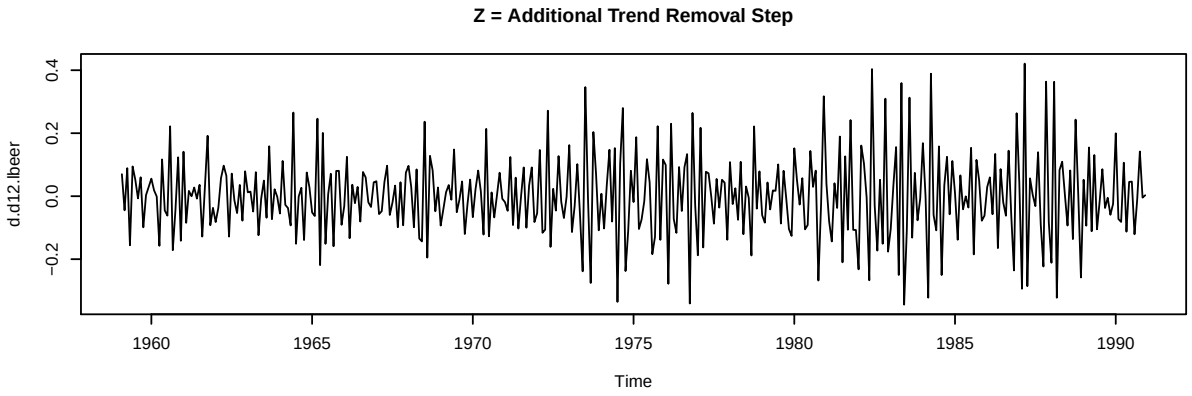
Wir illustrieren diese Vorgehen an der Zeitreihe der australischen Bierproduktion, welche wir bereits schon kennen. Dieses Zeitreihe von Januar 1958 bis Dezember 1990 enthält monatliche Messwerte zur Bierproduktion. Um die Varianz zu stabilisieren logarithmieren wir die Reihe vorher.

```
load(".../cbe.rda")
```

```
> par( mfrow = c(3,1) )
> beer <- ts(cbe$beer, start=1958, freq=12)
> d12.lbeer <- diff(log(beer), lag=12)
> d.d12.lbeer <- diff(d12.lbeer)
> plot(log(beer), main = 'X = Logged Australian Beer Production')
> plot(d12.lbeer, main = 'Y = Seasonally Differenced log(Beer) Series' )
> plot(d.d12.lbeer, main = 'Z = Additional Trend Removal Step')
```



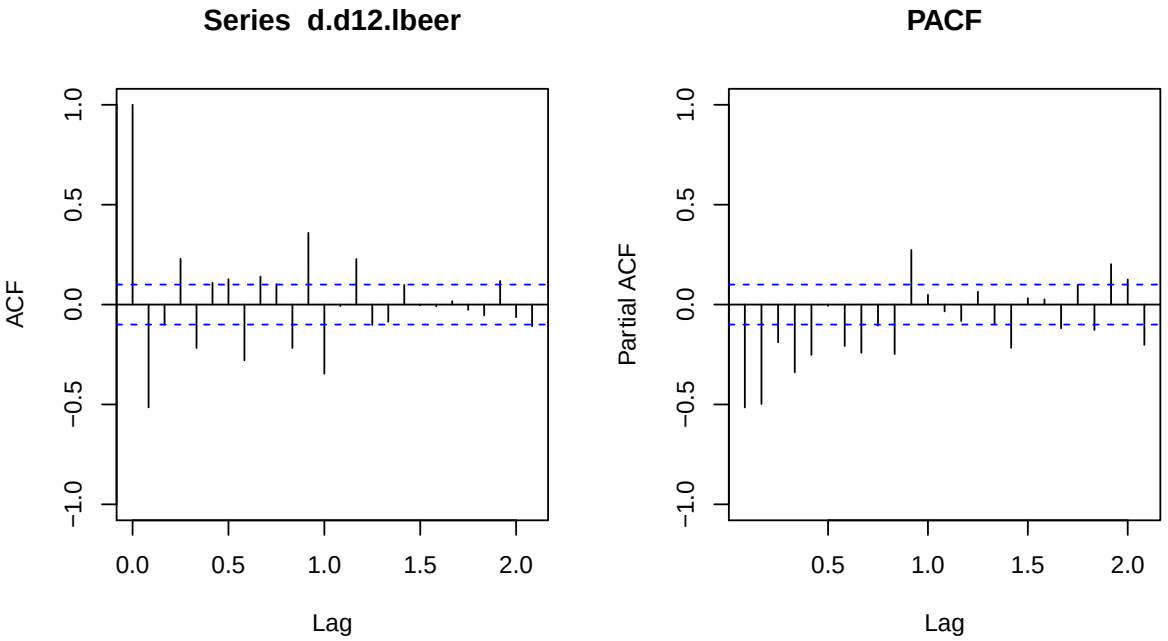




Wie wir sehen sind die beiden Zeitreihen X_t und Y_t nicht stationär. Die letzte Z_t scheint am ehesten stationär zu sein , allerdings kann man darüber diskutieren, ob die Varianz konstant ist. Wir machen weiter indem wir uns die ACF und PACF von Z_t anschauen.

```
> load(paste(data.dir, 'cbe.rda',sep = ''))
> beer <- ts(cbe$beer, start=1958, freq=12)
> d12.lbeer <- diff(log(beer), lag=12)
> d.d12.lbeer <- diff(d12.lbeer)
```

```
> par( mfrow = c(1,2) )
> acf(d.d12.lbeer, ylim=c(-1,1))
> pacf(d.d12.lbeer, ylim=c(-1,1), main="PACF")
```



Es ist deutlich sichtbar, dass Z_t noch eine serielle Abhängigkeitsstruktur aufweist und wir können versuchen ein $ARMA(p,q)$ daran anzupassen. Die der Ordnungen für p und q ist aufgrund der obigen Abbildungen nicht einfach. Die Abbildungen suggerieren grosse Werte für p und q und die Modellanpassung mit der nachfolgenden Residuenanalyse sowie einer AIC Inspektion bestätigen dies auch: ein gutes Resultat wird mit $p = 14$ und $q = 11$ erzielt.

Es ist (nicht so sehr in den obigen Abbildungen, aber im Allgemeine wenn wir diese Art von Daten analysieren) recht auffällig das die ACF und PACF Koeffizienten an Vielfachen der Periode grosse Werte aufweisen. Dies ist ein sehr typisches Verhalten für saisonal differenzierte Zeitreihen und liegt darin begründet, dass die saisonalen Effekte nicht konstant sind, sonder sich über die Jahre verändern. Ein einfaches Modell welches diesem Phänomen Rechnung trägt, ist das sog. Airline-Modell:

$$\begin{aligned} Z_t =& (1-\beta_1 B)(1+\zeta_1 B^{12})E_t \\ =& (1-\beta_1 B+\zeta_1 B^{12}+\beta_1\zeta_1 B^{13})E_t \\ =& E_t+\beta_1 E_{t-1}+\zeta_1 E_{t-12}+\beta_1\zeta_1 E_{t-13}. \end{aligned} \tag{55}$$

Dies ist ein MA(13-Modell, wobei die meisten Koeffizienten gleich null sind. Da es aus einem $MA(1)$-Modell gebildet wurde mit B als ein Operator des charakteristischen Polynoms und einem zweiten Operator B^S nennt man dieses Modell $SARIMA(0,1,1)(0,1,1)^{12}$. Diese Idee kann generalisiert werden: man passt AR und MA Teil an Z_t an, mit B und B^S als

Operatoren in den charakteristischen Polynomen, was wiederum zu einem $ARMA$-Modell für Z_t führt.

Definition: Eine Zeitreihe Z_t folgt einem $SARIMA(p, d,)(P, D, Q)^S$ Prozess falls die folgende Gleichung gilt:

$$\Phi(B)\Phi_S(B^S)Z_t = \Theta(B)\Theta(B^S)E_t \tag{56}$$

wobei die Zeitreihe Z_t von X_t kommt nachdem die Saison und der Trend durch Differenzierung entfernt wurde, d.h $Z_t = (1-B)^d(1-b^S)^D X_t$.

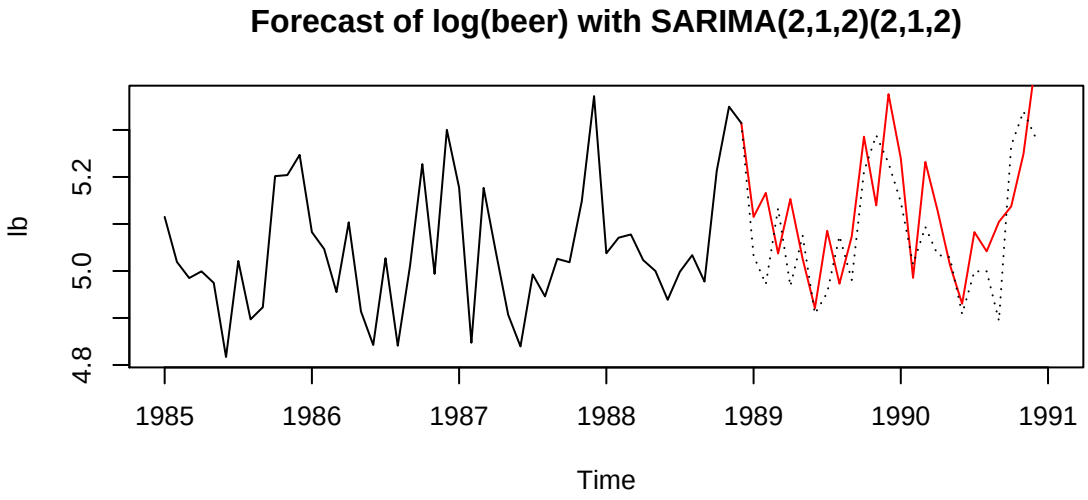
Glücklicherweise stellt sich heraus, dass in der Regel $d = D = 1$ genügt. Die Wahl der Ordnung von p, q, P und Q, kann anhand der ACF und PACF gemacht werden in dem man nach cut-offs sucht. Meistens sind diese cut-offs jedoch alles andere als offensichtlich, und somit wird häufig die folgende Alternative angewendet: alle Modelle mit p, q, P und $Q \leqslant 2$ werden berechnet und mit einer AIC-basierten Rastersuche bewertet.

Für unser Beispiel hat ein $SARIMA(2,1,2)(2,1,2)^{12}$ den kleinsten AIC-Wert und zeigt zudem befriedigende Residuen. Allerdings würde ein $SARIMA(14,1,11)(0,1,0)^{12}$ noch etwas besser performen.

```
fit <- arima( log(beer),  order = c(2,1,2), seasonal = c(2,1,1) )
```

Der Vorteil von $ARIMA$ und $SARIMA$ besteht vor allem darin, schell, bequeme und einfache eine Vorhersage zu berechnen. Wir werden dies in Kapitel **??** noch genauer besprechen, zeigen jetzt aber schon anhand eines Beispiels das potential. Von der logarithmierten Daten der Bierproduktion, lassen wir die letzten zwei Jahre weg, bevor wir das $SARIMA$-Modell an die gekürzte Zeitreihe anpassen. Auf Basis diese Modells berechnen wir eine 2-jahres Vorhersage, welche in der Abbildung unten als rote Linie dargestellt ist. Die original Daten sind als schwarze Linie dargestellt und ab Januar 1989 als schwarze, gepunktete Linie (out-of-sample, 1989-1990). Die Vorhersage ist recht vernünftig.

```
> lb <- window( log(beer) , start = c(1985, 1), end = c(1988,12), freq = 12)
> fit <- arima( lb,  order = c(2,1,2), seasonal = c(2,1,1) )
> plot( lb, xlim = c(1985, 1991),
+   main= 'Forecast of log(beer) with SARIMA(2,1,2)(2,1,2)')
> x.pred <- time(window(log( beer), start = c(1988, 12), freq = 12))
> ### rot = Vorhersage, schwarz gepunktet = echte Messwerte
> lines( x = as.numeric( x.pred),
+          y = c( 5.314683, as.numeric( predict( fit, n.ahead = 24)$pred)),
+          col= 'red' )
> lines( window(log( beer), start = c(1988, 12), freq = 12), lty = 3 )
```



Der Vorteil von $ARIMA$ und $SARIMA$ besteht vor allem darin, schell, bequem und einfache eine Vorhersage zu berechnen. Wir werden dies in Kapitel **??** noch genauer besprechen, zeigen jetzt aber schon anhand eines Beispiels das potential.

Die folgenden Ratschläge sollen beim Vorgehen helfen $SARIMA$-Modelle anzupassen (model fitting):

1. Saisonale Differenzierung. Der Lag s wird aufgrund der Periodizität der Daten bestimmt und für die Ordnung D genügt in den meisten Fällen 1.

2. Darstellen der saisonal differenzierten Zeitreihe. Entscheide ob die Reihe stationär ist, oder ob zusätzliches Differenzieren (Lag = 1 notwendig ist, um einen potentiellen Trend zu entfernen. Falls nicht, setze $d = 0$ und fahre weiter. Fall ja, setze $d = 1$, was in den meisten Fällen genügt. most series.

3. Darstellen der ACF und PACF des Outputs von Punkt 2, d.h. Z_t, um die serielle Abhängikeitsstruktur zu analysieren. Suche nach Koeffizienten/cut-offs niedriger Lags, welche die direkte, kurzfristige Abhängigkeit anzeigen und bestimme p und q. Danach müssen die Koeffizienten/cut-offs untersucht werden, welche Vielfache der Periode s sind und auf die saisonale Abhängigkeit hindeuten. Bestimme dann P und Q.

4. Berechne das Modell mit der `arima()`-Funktion. Im Gegensatz zur Anpassung eines $ARIMA$-Modells, geschieht dies ausschliesslich an der original Daten, indem die beiden Argumente `order=c(p,d,q)` und `seasonal=c(P,D,Q)` gesetzt werden.

5. Residuenanalyse (Residuen müssen sich wie weisses Rauschen verhalten). Falls mehrere Modell in Frage kommen, sollte man die AIC-Statistik als Entscheidungshilfe verwenden.

8 Regression zwischen Zeitreihen

8.1 Problemstellung

Häufig hat man neben der primär interessierenden Zeitreihe y_t noch andere Zeitreihen $x_t^{(1)}, \ldots, x_t^{(q)}$ zur Verfügung, die den Verlauf von y_t beeinflussen oder beschreiben können. Falls die Beziehung zwischen y_t und den $x_t^{(j)}$ linear ist, ergibt sich das aus der Regression bekannte lineare Modell

$$\begin{aligned} Y_t &= \beta_0 + \beta_1 x_t^{(1)} + \ldots + \beta_q x_t^{(q)} + E_t \\ &= \sum_{j=0}^{q} \beta_j x_t^{(j)} + E_t, \qquad \text{mit } x_t^{(0)} = 1 \text{ für alle } t = 1, \ldots, N. \end{aligned} \tag{57}$$

Wie wir gleich sehen werden, können die erklärenden Variablen $x_t^{(j)}$ dabei deterministisch oder zufällig sein.

Ein einfaches Beispiel für den *deterministischen* Fall ist ein polynomiales Trendmodell $Y_t = \beta_0 + \beta_1 t + \ldots + \beta_q t^q + E_t$, bei dem $x_t^{(j)} = t^j$ ist, oder ein saisonales Modell, bei dem $x_t^{(j)}$ beim j-ten Monat eine Eins und sonst Nullen hat, oder eine Kombination von beidem.

Weitere Beispiele sind die Modelle, die den Effekt einer Intervention zu einem bekannten Zeitpunkt T beschreiben. Dann ist $x_t = 0$ für $t < T$, während für $t \geqslant T$ die Form von x_t je nach vermuteter Art des Effekts verschieden ist. Für einen konstanten Effekt wird man $x_t = 1\ (t \geqslant T)$ wählen, für einen kumulativen Effekt $x_t = t - T + 1\ (t \geqslant T)$ und für einen abklingenden Effekt $x_t = \delta^{t-T}\ (t \geqslant T)$ mit einem $\delta < 1$.

Von *zufälligen* x-Variablen sprechen wir, wenn diese selber aus Messungen bestehen, wie zum Beispiel wenn x_t die gemessene Temperatur am Tag t ist und y_t der Schadstoffgehalt in der Luft.

Wie in der Regressionsrechnung werden wir alle x-Variablen als feste Grössen betrachten, egal ob sie auch aus Messungen bestehen oder nicht.

Im Modell (57) wird angenommen, dass der Einfluss der Reihen $x_t^{(1)}, \ldots, x_t^{(q)}$ auf die Zielgrösse y_t simultan geschieht. Häufig ist aber eine gewisse Zeitverschiebung im Zusammenhang der Reihen plausibel, d.h. y_t hängt auch von $x_{t-k}^{(j)}$ ab mit $k > 0$. Eine Änderung in der erklärenden Reihe zur Zeit $t - k$ (Vergangenheit!) wirkt sich erst zur Zeit t auf die Ziel-Zeitreihe y_t aus. Diese Verallgemeinerung kann man leicht in unser Modell (57) einbauen, indem man zeitverschobene Variablen (sogenannte lagged Variables) als zusätzliche erklärende Variablen ins Modell aufnimmt. Dabei bekommt man aber sehr schnell ein Modell mit vielen unbekannten Parametern.

Wesentlich beim Modell (57) ist, dass es *keine* Rückkopplung von y auf $x_t^{(j)}$ gibt. Wir können für y_t und $x_t^{(j)}$ auch nicht-stationäre Zeitreihen zulassen. Für $x_t^{(j)}$ müssen wir keine Modelle entwickeln. Dies wird erst dann notwendig, wenn man das Modell (57) allenfalls für Vorhersagen anwenden will und deshalb auch zukünftige Werte der $x_t^{(j)}$ benötigt.

Die Fehler E_t müssen unabhängig von den erklärenden Variablen $x_s^{(j)}$ sein für alle j und alle s, t. Falls dies nicht erfüllt ist, so muss man $y_t, x_t^{(1)}, \ldots, x_t^{(q)}$ als multivariate Zeitreihe analysieren (wird im CAS-DA nicht behandelt).

In der gewöhnlichen Regressionsrechnung macht man üblicherweise die Annahme, dass die Fehler E_t im Modell (57) untereinander unabhängig sind. In diesem Fall ist der Kleinste-Quadrate-Schätzer ein erwartungstreuer Schätzer mit minimaler Varianz. Ausserdem kann er dann aus den Beobachtungen mit Hilfe von Matrizen direkt berechnet werden. Man nennt ihn deshalb auch *Best Linear Unbiased Estimator* (BLUE).

Im Rahmen von **Zeitreihen** kann man im Allgemeinen nicht davon ausgehen, dass die Fehler E_t unkorreliert sind. Wir müssen z.B. damit rechnen, dass auf grosse positive Fehler meist positive Fehler folgen und auf grosse negative Fehler ebenfalls negative Fehler folgen. Dies bedeutet dann, dass die Fehler positiv korreliert sind.

Ein häufiger Grund für diese Autokorrelationen ist, dass eine erklärende Variable bei der Modellbildung nicht berücksichtigt wurde. Wenn aufeinanderfolgende Werte dieser fehlenden Variable korreliert sind, dann sind die Fehler (Residuen) im angepassten Modell (ohne diese Variable) korreliert.

Die Korrelationen der Fehler haben nun verschiedene Auswirkungen:

- Die gewöhnlichen Kleinste-Quadrate-Schätzer sind zwar immer noch **erwartungstreu**, sie haben aber nicht mehr minimale Varianz, d.h. es gibt genauere Schätzer.
- Die Konfidenzintervalle und Tests für die Signifikanz sind **nicht mehr genau** (oder sogar grob falsch), wenn sie mit den gewöhnlichen Formeln für Kleinste-Quadrate-Schätzer berechnet werden, für die die Unkorreliertheit der Fehler vorausgesetzt wird. Die Standardfehler der Koeffizienten β_j werden oft unterschätzt, wenn man die gewöhnlichen Formeln der multiplen Regression verwendet.

Man kann also leider die Korrelationen der Fehler nicht einfach ignorieren. Als erstes möchte man daher einmal entscheiden, ob solche Korrelationen überhaupt auftreten. Deshalb analysiert man die Korrelationsstruktur der Residuen. Dazu verwendet man am besten die gewöhnlichen und partiellen Autokorrelationen (vgl. Kapitel 4.3).

Bei vorhandenen Korrelationen möchte man dann etwas gegen die oben beschriebenen Auswirkungen tun, also korrekte Standardfehler für die Kleinste-Quadrate-Schätzer angeben. Idealerweise möchte man andere Schätzer (inklusive deren Standardfehler) berechnen, die einen kleineren Standardfehler als der Kleinste Quadrate-Schätzer haben. Dies ist das Thema der folgenden Unterkapitel. Wir werden zwei Methoden zur Schätzung der Koeffizienten und deren Standardfehler lernen. Wir betrachten die Problemstellung und Lösung an den folgenden beiden Beispielen.

Simulationsbeispiel

Wir betrachten das folgende Regressionsmodell

$$\begin{aligned} x_t &= t/50 \\ y_t &= x_t + 2x_t^2 + E_t \end{aligned}$$

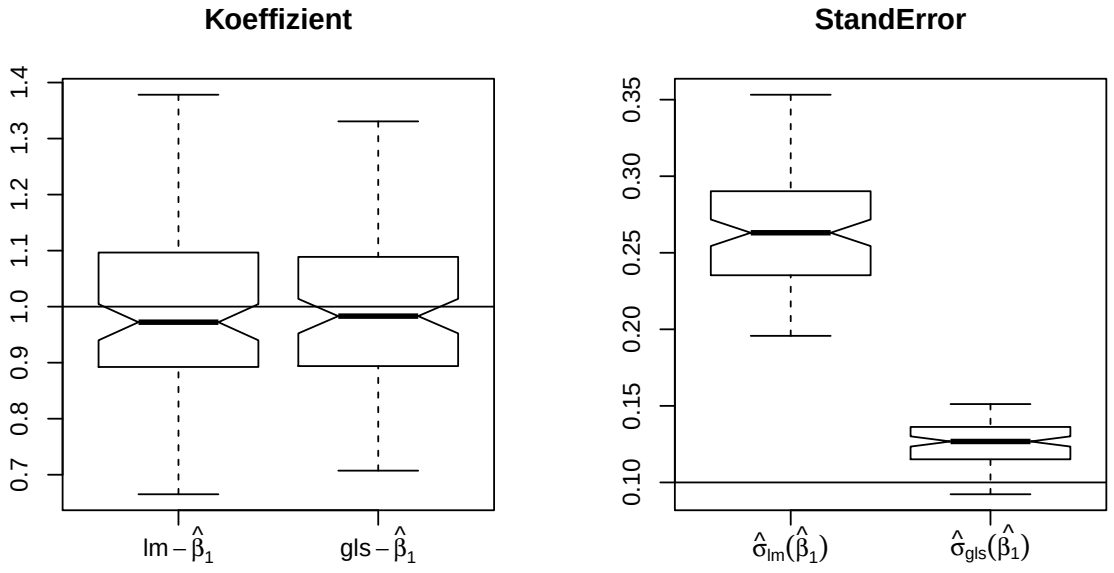
mit E_t ein AR(1)-Prozess mit $\alpha = -0.65$ und $\sigma = 0.1$ und untersuchen die Auswirkungen der beiden Schätzer

- `lm`: lineare Regression ohne Berücksichtigung der korrelierten Fehler
- `gls`: verallgemeinerte Kleinste-Quadrate mit Berücksichtigung der korrelierten Fehler (Details zum Schätzer folgen weiter hinten)

Dabei simulieren wir das Modell 100 Mal und betrachten die Schätzung β_1 des linearen Terms x_t.

In der Abbildung unten sind links die Schätzwerte mit den beiden Methoden abgetragen. Man sieht deutlich, dass beide Schätzer um den wahren Wert ($\beta = 1$) streuen. Dies bestätigt, dass der Kleinste-Quadrate-Schätzer (lm) erwartungstreu ist. Die Streung dieser Werte (Boxplot) ist etwa gleich und entspricht eigentlich einer parametrischen Bootstrap Genauigkeitsschätzung.

In der Figur rechts sind die 100 Schätzwerte fr̈ den Standardfehler, d.h. bei lm gemäss bekannter Formel aus Regression (ohne Korrelationsberücksichtigung) und bei gls mit Berücksichtigung der Korrelationstruktur, aufgetragen. Hier sieht man deutlich, dass ohne Berücksichtigung der Korrelationstruktur der Fehlern die Streuungsschätzwerte (gemäss lm) völlig falsch sind. Die Schätzwerte der Standardfehlers von β_1 mit gls ist dagegen korrekt und streut um den wahren Wert.



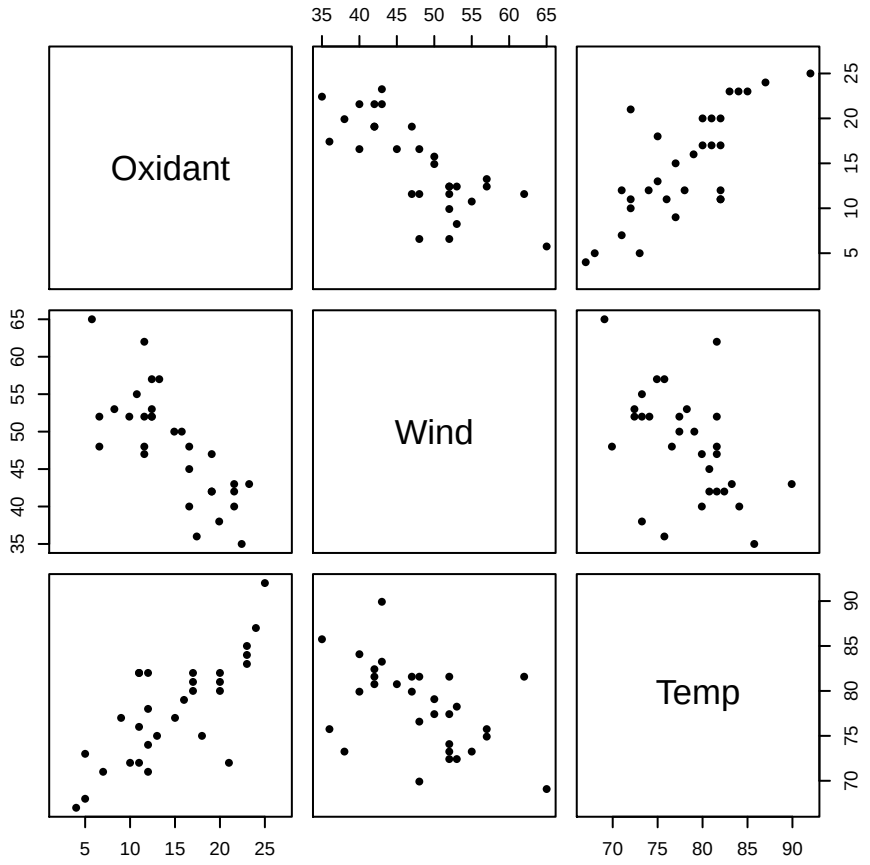
Beispiel Luftschadstoffe

Wir untersuchen in diesem Beispiel die Abhängigkeit eines photochemischen Schadstoffes (morgendliche Maximalwerte) von den beiden meteorologischen Variablen Wind und Temperatur. Die meteorologischen Variablen sind morgendliche Stundenmittel. Die Daten in Tabelle 2 stammen aus Los Angeles von 30 aufeinanderfolgenden Sommertagen.

Tabelle 2: Schadstoffmessungen und meteorologische Daten von 30 aufeinanderfolgenden Sommertagen in Los Angeles (Y: Oxidant, X1: Wind, X2: Temp)

t	X1	X2	Y	t	X1	X2	Y	t	X1	X2	Y
1	50	77	15	11	47	82	11	21	52	73	5
2	47	80	20	12	40	80	17	22	48	68	5
3	57	75	13	13	42	81	20	23	65	67	4
4	38	72	21	14	40	85	23	24	53	71	7
5	52	71	12	15	48	82	17	25	36	75	18
6	57	74	12	16	50	79	16	26	45	81	17
7	53	78	12	17	55	72	10	27	43	84	23
8	62	82	11	18	52	72	11	28	42	83	23
9	52	82	12	19	48	76	11	29	35	87	24
10	42	82	20	20	52	77	9	30	43	92	25

In den Streudiagrammen der unteren Abbildung ist ein starker, positiv linearer Zusammenhang zwischen der Schadstoffmenge und der Temperatur erkennbar. Dagegen besteht ein negativ linearer Zusammenhang zwischen dem Schadstoff und dem Wind. Zwischen den beiden erklärenden Variablen Wind und Temperatur besteht ein leicht negativer Zusammenhang. Diese Struktur der Daten überrascht uns nicht, da der Wind für eine starke Durchmischung der Luftmassen sorgt und so der Schadstoff “besser” verteilt wird. Der Wind wirkt zudem auch abkühlend.


Oxidant
Wind
Temp
35 40 45 50 55 60 65
5 10 15 20 25
35 40 45 50 55 60 65
5 10 15 20 25
70 75 80 85 90
70 75 80 85 90

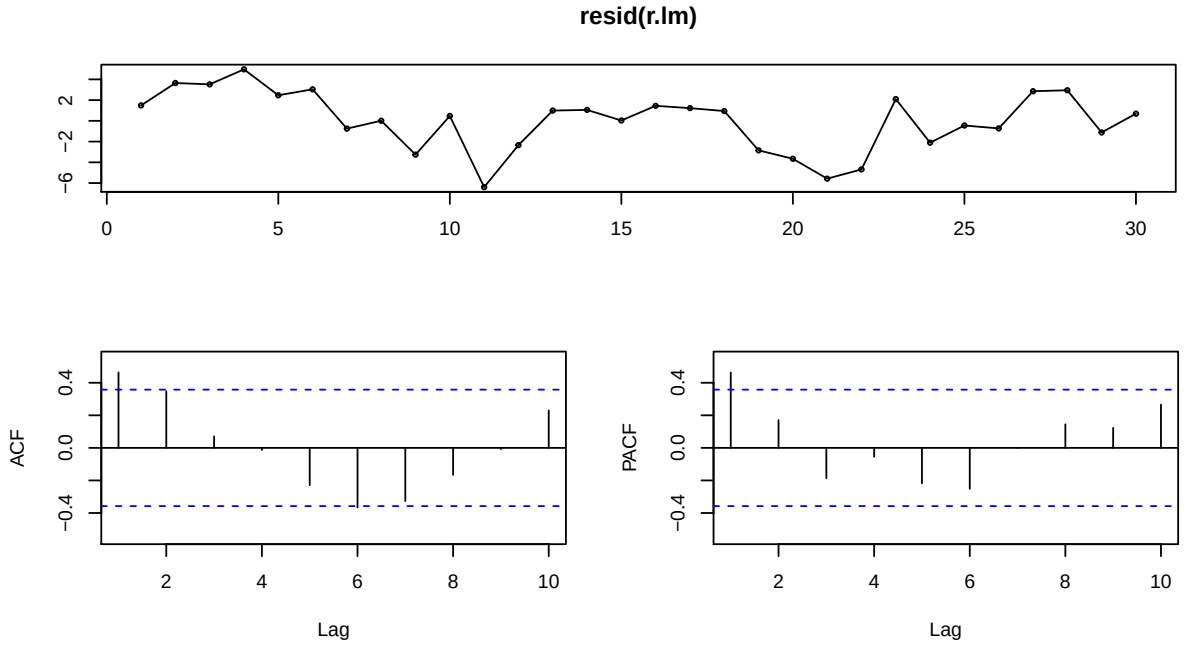
Zur Modellierung der Daten betrachten wir aufgrund der obigen Erkenntnisse das multiple lineare Modell

$$Y_t = \beta_0 + \beta_1 x_t^{(1)} + \beta_2 x_t^{(2)} + E_t$$

Mit R erhalten wir die folgenden Schätzungen.

```
> erg.poll <- lm(Oxidant ~ Wind+Temp,data=Pollute)
> summary(erg.poll,corr=F)

Call:
lm(formula = Oxidant ~ Wind + Temp, data = Pollute)

Residuals:
   Min     1Q Median     3Q    Max
-6.394 -1.861  0.583  1.946  4.966

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -5.2033    11.1181   -0.47     0.64
Wind         -0.4271     0.0864   -4.94  3.6e-05 ***
Temp          0.5204     0.1081    4.81  5.0e-05 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 2.95 on 27 degrees of freedom
Multiple R-Squared: 0.777,      Adjusted R-squared: 0.761
F-statistic: 47.1 on 2 and 27 DF,  p-value: 1.56e-009
```

In Abbildung unten sehen wir deutliche Abhängigkeiten bei den Residuen. Aufgrund der Korrelationsstruktur können wir von einem AR(1)-Prozess bei den Residuen ausgehen.



8.1.1 Bemerkung/Warnung: Durbin-Watson Test

In einigen Statistik-Programmen wird bei der Regressionsrechnung automatisch die geschätzte **erste** Autokorrelation und die Teststatistik $\widehat{D}$ des Durbin-Watson Test angegeben. Wir gehen deshalb hier kurz auf diesen Test ein und zeigen als Warnung vor allem dessen Schwäche, was uns veranlasst, ihn nicht zu empfehlen.

Der Durbin-Watson Test prüft die Nullhypothese $H_0 : \rho(1) = 0$ gegen die Alternative $H_A : \rho(1) \neq 0$. (Zur Erinnerung: mit $\rho(1)$ bezeichnen wir die wahre Autokorrelation zum Lag 1.)

Die Teststatistik $\widehat{D}$ wird wie folgt berechnet

$$\widehat{D} = \frac{\sum_{t=2}^{N}(R_t - R_{t-1})^2}{\sum_{t=1}^{N} R_t^2}$$

mit $R_t = y_t - \widehat{y}_t$ (Residuum zur Zeit t.)

Es gibt einen approximativen Zusammenhang zwischen der Teststatistik $\widehat{D}$ und der geschätzten ersten Autokorrelation $\widehat{\rho}(1)$:

$$\widehat{D} \approx 2(1 - \widehat{\rho}(1))$$

Die Teststatistik nimmt Werte zwischen 0 ($R_t = R_{t-1}$) und 4 ($R_t = -R_{t-1}$) an. Werte nahe bei 2 deuten auf Unkorreliertheit der Residuen hin.

Die Verteilung der Teststatistik hängt von den erklärenden Variablen x ab. Für gewisse Niveaus sind aber Werte d_U und d_L tabelliert, mit denen die folgenden Aussagen möglich sind:

1. $d_U < \widehat{D} < 4 - d_U$ akzeptiere H_0
2. $\widehat{D} < d_L$ oder $\widehat{D} > 4 - d_L$ verwerfe H_0
3. sonst, keine Entscheidung möglich

Aber der Durbin-Watson Test testet nur die erste Autokorrelation. In allen Fällen, in denen ein AR(p) Prozess mit $p > 1$ vorliegt und die erste Autokorrelation sehr klein ist, versagt der Test kläglich.
Fazit: Verwenden Sie ihn nicht, sondern betrachten Sie die ACF und PACF!

8.2 Cochrane-Orcutt Methode für Autoregressive Fehler

Als erste Methode betrachten wir die nach Cochrane und Orcutt benannte Methode, die zur Behandlung von Regressionsfehlern E_t mit AR(p)-Struktur einfach angewandt werden kann. Das Verfahren besteht aus zwei Stufen. Da man nur ein Programm zur gewöhnlichen Regressionsrechnung benötigt, kann man dies mit allen Programmen rechnen und so korrelierte Fehler korrekt behandeln. Wir führen die ganze Methodik am einfachsten Fall der AR(1)-Fehler ein. Diese Korrelationsstruktur der Fehler haben wir auch in unserem Beispiel gefunden. Anschliessend verallgemeinern wir dies auf AR(p)-Strukturen.

Im Kapitel 8.3 werden wir dann eine allgemeinere Methodik zur Berücksichtigung der Korrelationen vorstellen, welche mit den entsprechenden **R**-Funktionen angewendet werden soll.

8.2.1 Autoregressive Fehler der Ordnung 1

Nehmen wir also an, dass die Fehler E_t (wie die Residuen in unserem Simulationsbeispiel) tatsächlich einem AR-Modell der Ordnung 1 folgen

$$E_t = \alpha E_{t-1} + U_t \tag{58}$$

mit U_t i.i.d $(\mathrm{E}(U_t) = 0, \mathrm{Var}(U_t) = \eta^2)$ und kausal. Wir werden sehen, dass wir mit einer einfachen Transformation der Zielgrösse und der erklärenden Variablen zu einem Regressionsmodell gelangen, bei dem die Fehler unkorreliert sind. In diesem neuen Modell können wir dann die gewöhnliche Kleinste-Quadrate-Regression rechnen.

Wir betrachten die Transformation am Beispiel der Luftschadstoffdaten, bei dem wir zwei erklärende Variablen $x_t^{(1)}$ und $x_t^{(2)}$ haben. Wir bilden die Grösse $Y_t^* = Y_t - \alpha Y_{t-1}$. Dann ist

$$\begin{aligned} Y_t^* &= Y_t - \alpha Y_{t-1} \quad | \text{ setze Modell ein} \\ &= \beta_0 + \beta_1 x_t^{(1)} + \beta_2 x_t^{(2)} + E_t - \alpha(\beta_0 + \beta_1 x_{t-1}^{(1)} + \beta_2 x_{t-1}^{(2)} + E_{t-1}) \\ &= \beta_0(1-\alpha) + \beta_1(x_t^{(1)} - \alpha x_{t-1}^{(1)}) + \beta_2(x_t^{(2)} - \alpha x_{t-1}^{(2)}) + E_t - \alpha E_{t-1} \\ &= \beta_0^* + \beta_1 x_t^{*(1)} + \beta_2 x_t^{*(2)} + U_t \end{aligned}$$

mit $\beta_0^* = \beta_0(1-\alpha)$, $x_t^{*(1)} = x_t^{(1)} - \alpha x_{t-1}^{(1)}$ und $x_t^{*(2)} = x_t^{(2)} - \alpha x_{t-1}^{(2)}$. Dieses transformierte Modell mit den "gesternten" Grössen erfüllt die Voraussetzungen, die im einfachen Regressionsmodell gemacht werden, d.h. die Fehler U_t sind (siehe (58)) unabhängig identisch

verteilt, also insbesondere unkorreliert. Um aber die "gesternten" Grössen zu berechnen, müssen wir α kennen, was in der Regel nicht der Fall ist.

Cochrane und Orcutt haben ein schrittweises Vorgehen vorgeschlagen, um mit einem geschätzten α die Berechnungen durchzuführen:

1. Bestimme den gewöhnlichen Kleinste-Quadrate-Schätzer für die Koeffizienten im ursprünglichen Modell.

2. Berechne die Residuen und schätze daraus α: da wir einen AR(1)-Prozess für die Residuen annehmen, können wir α gerade mit $\widehat{\rho}(1)$ (erste gewöhnliche Autokorrelation) schätzen. Man kann natürlich auch eine andere Methode zur Schätzung von α verwenden.

3. Passe an die transformierten Grössen ("gesternte" Variablen) ein lineares Modell an. Die Koeffizienten des ursprünglichen Modells werden durch

$$\widehat{\beta}_0 = \widehat{\beta_0^*}/(1-\alpha), \quad \widehat{\beta}_1 = \widehat{\beta_1^*}, \quad \widehat{\beta}_2 = \widehat{\beta_2^*}$$

   geschätzt. Für die Steigungskoeffizienten können wir nicht nur die Schätzungen sondern auch gerade die Konfidenzintervalle aus dem transformierten Modell entnehmen. Für den Achsenanbschnitt können wir den Standardfehler wie folgt aus dem "gesternten Modell" berechnen

$$\mathrm{se}(\widehat{\beta}_0) \quad = \quad \mathrm{se}(\widehat{\beta_0^*})/(1-\widehat{\alpha}).$$

   Dabei wird $\widehat{\alpha}$ als eine feste Grösse behandelt.
   Hinweis: Falls man eine weitere erklärende Variable $(1-\alpha)$ einfügt (dies entspricht der Transformation der 1er Spalte) und dafür den Achsenabschnitt weglässt, kann man die Umrechnung des Achsenabschnittes umgehen, da dann der Koeffizient gerade gleich β_0 ist. Siehe dazu im folgenden Beispiel den Punkt 3, wo wir die Variable `xo.stern` einführen.

4. Untersuche die Residuen u_t aus dem transformierten Modell auf Unabhängigkeit. Falls keine Autokorrelationen vorhanden sind, kann man dieses Verfahren abschliessen. Falls sich immer noch Abhängigkeiten zeigen, iteriert man das Vorgehen, oder was mehr zu empfehlen ist, man verwendet einen autoregressiven Prozess höherer Ordnung.

5. Mit den korrekten Konfidenzintervallen kann man nun testen, ob eine Modellreduktion möglich ist.

Wir wollen diese Schritte nun am **Beispiel der Schadstoffmessungen** nachvollziehen.

1. Die gewöhnlichen Kleinste-Quadrate-Schätzer haben wir in Kapitel 8.1 bereits berechnet.

2. Die Analyse der Korrelationsstruktur hat uns gezeigt, dass ein AR(1)-Prozess sinnvoll ist. Die Schätzung von α entspricht der ersten Autokorrelation $\widehat{\alpha} = \widehat{\rho}(1) = 0.463$.

3. Im ersten Schritt berechnen wir die transformierten Grössen (Hinweis: Prüfen Sie jeweils die Funktion `lag()` ihres Statistikpaketes nach!!) und passen dann das Regressionsmodell an.

```
> Ox.stern <- t.Oxidant - 0.463*lag(t.Oxidant,-1)
> Wi.stern  <- t.Wind - 0.463*lag(t.Wind,-1)
> Te.stern  <- t.Temp - 0.463*lag(t.Temp,-1)
>
> x0.stern  <- rep(1 - 0.463,length(Te.stern))
> Poll.stern <- lm(Ox.stern ~ x0.stern + Wi.stern + Te.stern -1 )
> summary(Poll.stern)

Call:
lm(formula = Ox.stern ~ x0.stern + Wi.stern + Te.stern - 1)

Residuals:
   Min     1Q Median     3Q    Max
-6.565 -2.150  0.362  1.767  3.892

Coefficients:
         Estimate Std. Error t value Pr(>|t|)
x0.stern  -4.5291    11.2676   -0.40  0.69100
Wi.stern  -0.4104     0.0726   -5.65    6e-06 ***
Te.stern   0.5006     0.1240    4.04  0.00043 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 2.64 on 26 degrees of freedom
Multiple R-Squared: 0.929,      Adjusted R-squared: 0.921
```

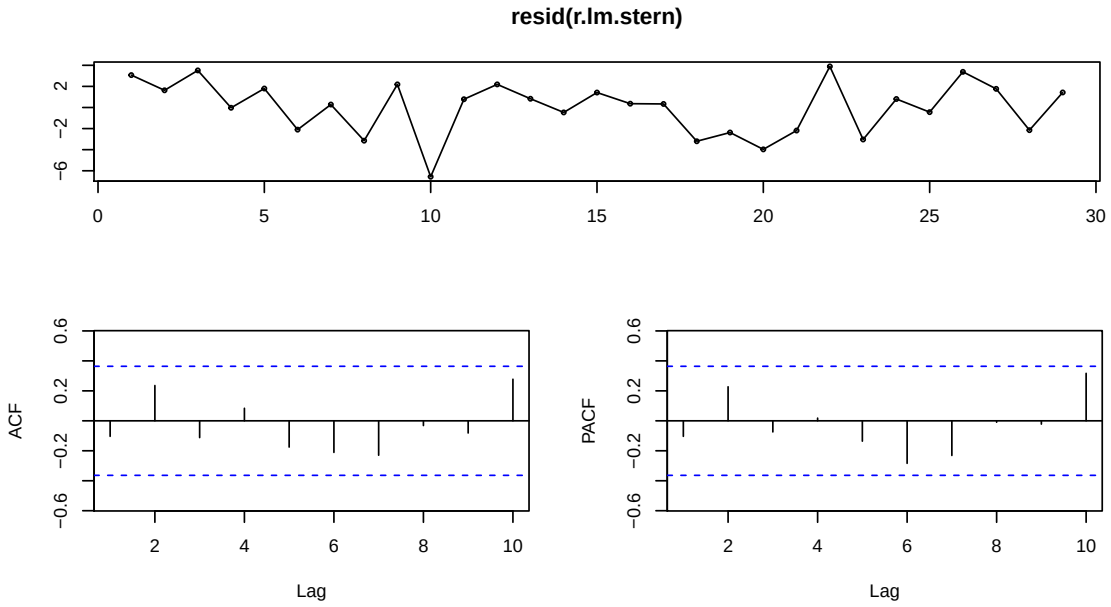
4. Die Residuen zeigen keine Korrelationen mehr (siehe Abbildung unten). Somit können wir das Verfahren hier abschliessen.

   In der nachfolgenden Tabelle sind die Resultate zusammengefasst.

	Kleinste-Quadrate		**Cochrane-Orcutt**	
	Schätzung	se	Schätzung	se
β_0	-5.203	11.118	-4.529	11.268
β_1	-0.427	0.086	-0.410	0.073
β_2	0.520	0.108	0.501	0.124

   Der Vergleich der beiden Methoden zeigt, dass sich die Koeffizientenschätzungen nur leicht ändern. Für β_1 ist der Standardfehler kleiner, während derjenige f—ür β_2 grösser wurde, wenn wir die Korrelationsstruktur berücksichtigen.

Den Achsenabschnitt $\widehat{\beta}_0$ und den zugehörigen Standardfehler können wir direkt ablesen, da wir die 1er Spalte auch transformiert haben (Variable `xo.stern` eingefügt) und beim Modellaufruf von `lm`, mit `-1` den Achsenabschnitt unterdrücken.



8.2.2 Autoregressive Fehler der Ordnung p

Wir betrachten wieder das Modell (57), wobei wir jetzt annehmen, dass die Fehler E_t einem AR-Prozess mit höherer Ordnung p folgen

$$E_t = \sum_{k=1}^{p} \alpha_k E_{t-k} + U_t, \qquad t = p+1, \ldots, N.$$

Dabei sind die U_t i.i.d. mit Erwartungswert 0 und Varianz η^2. Wenn wir diese Gleichung mit (57) kombinieren, erhalten wir

$$Y_t - \sum_{k=1}^{p} \alpha_k Y_{t-k} = \sum_{j=0}^{q} \beta_j (x_t^{(j)} - \sum_{k=1}^{p} \alpha_k x_{t-k}^{(j)}) + U_t. \tag{59}$$

Wir haben also ein lineares Modell mit den unabhängigen Fehlern U_t, wenn wir die Variablen Y_t durch $Y_t^* = Y_t - \sum_{k=1}^{p} \alpha_k Y_{t-k}$ und $x_t^{(j)}$ durch $x_t^{*(j)} = x_t^{(j)} - \sum_{k=1}^{p} \alpha_k x_{t-k}^{(j)}$ ersetzen.

$$Y_t^* = \sum_{j=0}^{q} \beta_j x_t^{*(j)} + U_t, \quad U_t \text{ iid}$$

Man kann nun zeigen, dass der Kleinste-Quadrate-Schätzer $\widehat{\boldsymbol{\beta}}$ im umgeschriebenen Modell (59) gleich dem besten linearen erwartungstreuen Schätzer $\widetilde{\boldsymbol{\beta}}$ im ursprünglichen Modell (57) ist (bis auf einen kleinen Randeffekt, weil das Modell (59) erst mit $\boldsymbol{t} = \boldsymbol{p} + \mathbf{1}$ beginnt).

Wenn $\boldsymbol{\alpha} = (\boldsymbol{\alpha_1}, \ldots, \boldsymbol{\alpha_p})$ unbekannt ist, schätzen wir $\boldsymbol{\alpha}$ wie bei $\boldsymbol{p} = \mathbf{1}$ aus den Residuen $\boldsymbol{e_t} = \boldsymbol{y_t} - \sum_j \widehat{\boldsymbol{\beta}}_j \boldsymbol{x_{t,j}}$ (mit $\widehat{\boldsymbol{\beta}}$ dem Kleinste-Quadrate-Schätzer) und ignorieren nachher den Unterschied zwischen dem wahren und dem geschätzten $\boldsymbol{\alpha}$. Für einen grossen Stichprobenumfang $\boldsymbol{N}$ kann man das auch rechtfertigen.

Bemerkung

- Mit der R Funktion `arima`, können wir grundsätzlich auch Regression mit korrelierten Fehlern berechnen. Wir empfehlen aber die Funktion `gls()` aus dem `nlme`-Package zu verwenden.
- Der Nachteil der Cochrane-Orcutt-Methode besteht darin, dass man in zwei Stufen vorgeht und insbesondere im zweiten Schritt den Fehler bei der Schätzung der AR-Parameter ignoriert und so tut, als ob diese AR-Parameter exakt bekannt wären.
- Anstelle eines solchen zweistufigen Verfahrens kann man auch $\boldsymbol{\alpha}$ und $\boldsymbol{\beta}$ simultan schätzen. Dies betrachten wir im folgenden Abschnitt 8.3.

8.3 Verallgemeinerte Kleinste-Quadrate: Anwendung

Der Nachteil der Cochrane-Orcutt-Methode besteht darin, dass man in zwei Stufen vorgeht und insbesondere im zweiten Schritt den Fehler bei der Schätzung der AR-Parameter ignoriert und so tut, als ob diese AR-Parameter exakt bekannt wären.

Besser ist es, beide Parametersets gleichzeitig zu schätzen, in dem man in (59) die Summe der Fehler im Quadrat minimiert. Man minimiert also

$$\sum_{t=p+1}^{n} \left(y_t - \sum_{k=1}^{p} \alpha_k y_{t-k} - \sum_{j=0}^{q} \beta_j x_t^{(j)} + \sum_{k=1}^{p} \sum_{j=0}^{q} \alpha_k \beta_j x_{t-k}^{(j)} \right)^2$$

bezüglich $\boldsymbol{\alpha}$ und $\boldsymbol{\beta}$. Bei normal verteilten Fehlern ist dies gleich der Maximum-Likelihood-Schätzung (im CAS-DA nicht behandelt).

Dabei handelt es sich um ein nichtlineares Problem. Die Lösung ist numerisch nicht ganz einfach.

Für R muss man die Library `nlme` vom Web heruntergeladen. Dort ist die Funktion `gls` enthalten, mit der man Kleinste-Quadrate-Probleme mit korrelierten Fehlern lösen kann. Dabei können neben AR-Fehlern auch allgemeinere Fehlerstrukturen behandelt werden.

Beispiel Luftschadstoffe

```
> r.Poll.gls <- gls(Oxidant ~ Wind+Temp,data=Pollute,
                    correlation = corARMA(form = ~ t,p=1),method="REML")
> summary(r.Poll.gls)
Generalized least squares fit by maximum likelihood
  Model: Oxidant ~ Wind + Temp
  Data: Pollute
  AIC BIC logLik
  150 157  -69.8
```

```
Correlation Structure: AR(1)
 Formula: ~t
 Parameter estimate(s):
  Phi
0.464

Coefficients:
            Value Std.Error t-value p-value
(Intercept) -4.24     11.09   -0.38  0.7054
Wind        -0.41      0.07   -5.73  <.0001
Temp         0.50      0.12    4.07  0.0004

 Correlation:
     (Intr) Wind
Wind -0.557
Temp -0.952  0.286

Standardized residuals:
   Min     Q1    Med     Q3    Max
-2.279 -0.679  0.229  0.522  1.774

Residual standard error: 2.79
Degrees of freedom: 30 total; 27 residual
```

Wir sehen, dass kein grosser Unterschied zu den Cochran-Orcutt Schätzungen besteht. Die Analyse der Residuen (wir verzichten hier auf die Bilder) ist etwas komplizierter. Mit `resid(...)` erhält man die Residuen aus dem Modell (57), d.h. Schätzungen für $\boldsymbol{E_t}$. Diese entsprechen in unserem Beispiel einem AR(1)-Prozess. Somit müssen wir in der Residuenanalyse überprüfen, ob die Residuen `resid(r.Poll.gls)` vom entsprechenden AR-Prozess kommen können.

Es ist leider nicht so, dass die gls-Residuen unkorreliert sind, sondern sie müssen der im Funktionsaufruf spezifizierter Korrelationsstruktur entsprechen!!!

Bemerkungen

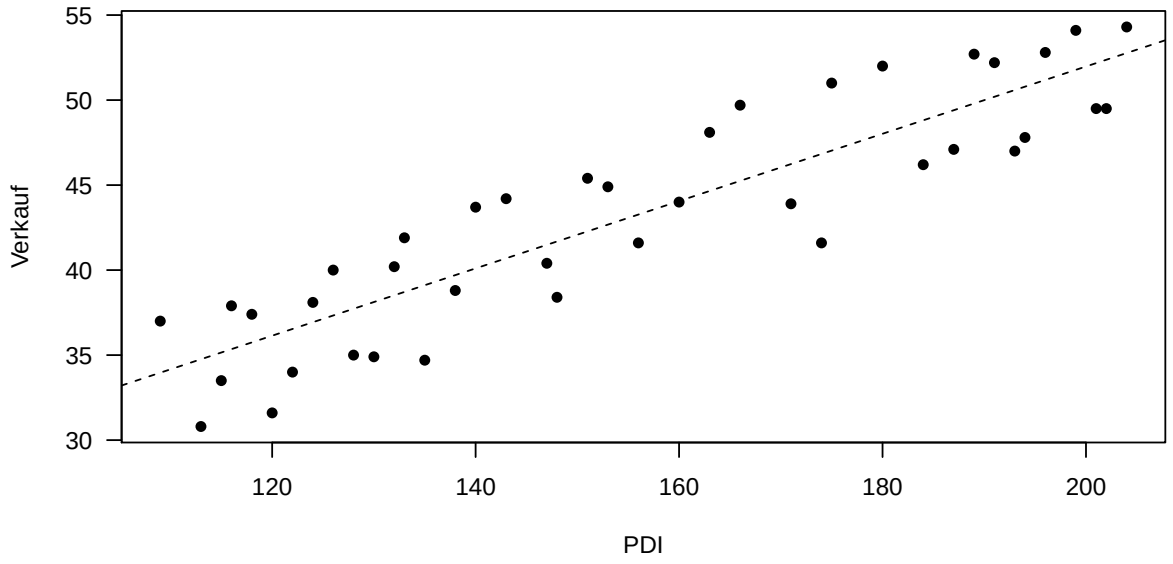
- Auch bei nicht-linearen Regressionsproblemen können die Korrelationsstrukturen berücksichtig werden, siehe die Funktion `gnls`.
- Die Library nlme enthält weiter die Funktionen `lme` und `nlme` bei denen komplizierte Korrelationsstrukturen und vielfältige varianzanalytische Modelle (z.B. geschachtelte und zufällige Faktoren) berechnet werden können.

8.4 Fehlende erklärende Variablen

Die Präsenz von Autokorrelation in den Residuen ist oft ein Hinweis auf das Fehlen von erklärenden Variablen. Wir betrachten dies an folgendem Beispiel. Eine Skifirma in den USA

untersuchte ihre Quartalsverkaufszahlen $\boldsymbol{Y_t}$ in Abhängigkeit von ökonomischen Grössen. Sie wählten als ökonomische Variable das persönlich verfügbare Einkommen (PDI, $\boldsymbol{x_t}$).

```
Call:
lm(formula = Verkauf ~ PDI, data = Ski)

Residuals:
   Min     1Q Median     3Q    Max
-5.229 -2.686  0.554  2.728  4.454

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.39215    2.53942    4.88 1.93e-05 ***
PDI          0.19791    0.01602   12.35 7.09e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.019 on 38 degrees of freedom
Multiple R-squared:  0.8006,Adjusted R-squared:  0.7953
F-statistic: 152.5 on 1 and 38 DF,  p-value: 7.089e-15
```



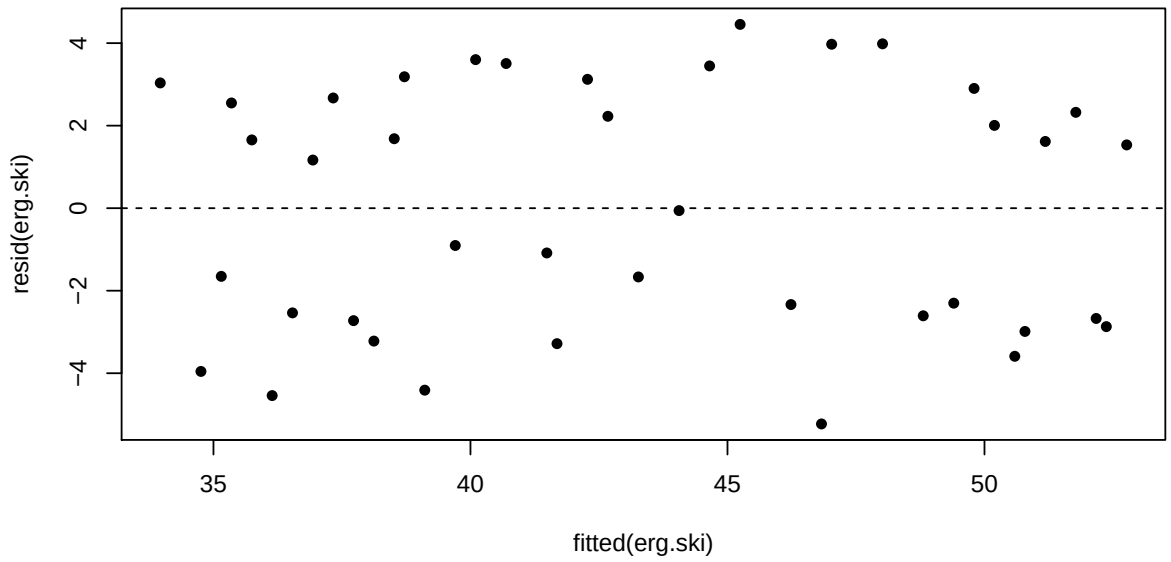
In der Abbildung oben sehen wir einen starken linearen Zusammenhang, den wir mit dem Modell

$$\boldsymbol{Y_t} = \boldsymbol{\beta_0} + \boldsymbol{\beta_1} \cdot \boldsymbol{x_t} + \boldsymbol{E_t} \tag{60}$$

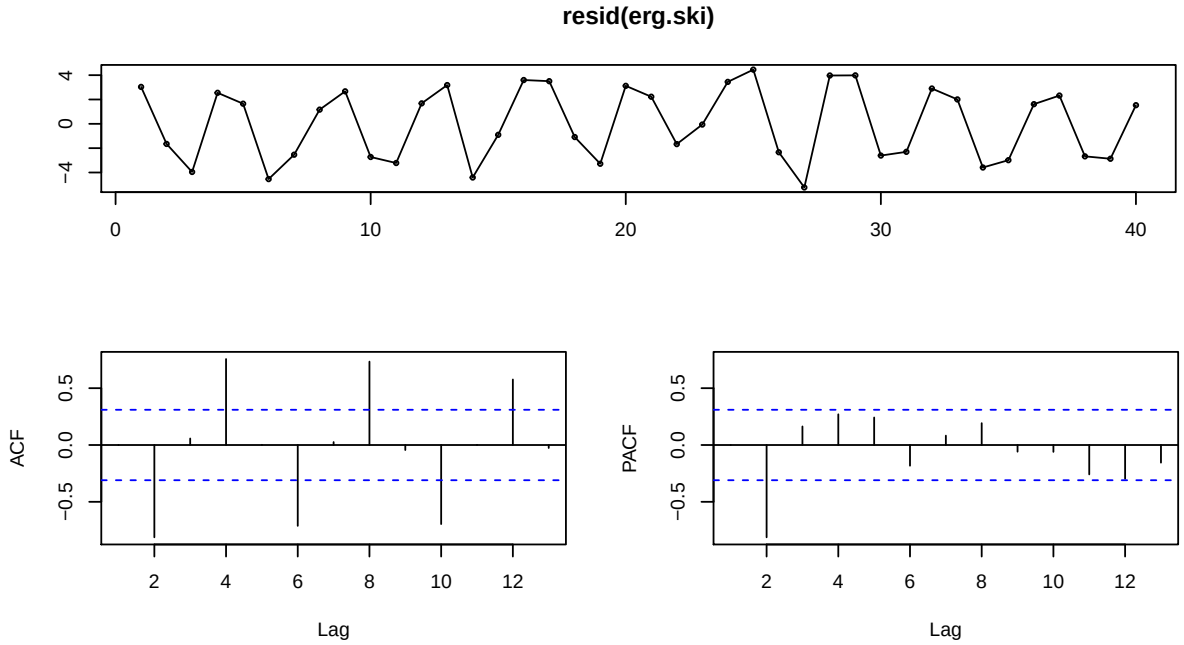
beschreiben können.

```
> erg.ski <- lm(Verkauf~PDI,data=Ski)
> summary(erg.ski)

Call:
lm(formula = Verkauf ~ PDI, data = Ski)

Residuals:
   Min     1Q Median     3Q    Max
-5.229 -2.686  0.554  2.729  4.454

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   12.392      2.539    4.88  1.9e-05 ***
PDI            0.198      0.016   12.35  7.1e-15 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 3.02 on 38 degrees of freedom
Multiple R-Squared: 0.801,      Adjusted R-squared: 0.795
F-statistic:  153 on 1 and 38 DF,  p-value: 7.11e-015
```

Das Bestimmtheitsmass ($\boldsymbol{R^2 = 0.801}$) ist sehr gross. Der Tukey-Anscombe-Plot weiter unten zeigt eine Struktur, in dem die Residuen wie auf zwei Linien um Null liegen, insbesondere nahe Null gibt es fast keine Residuen.

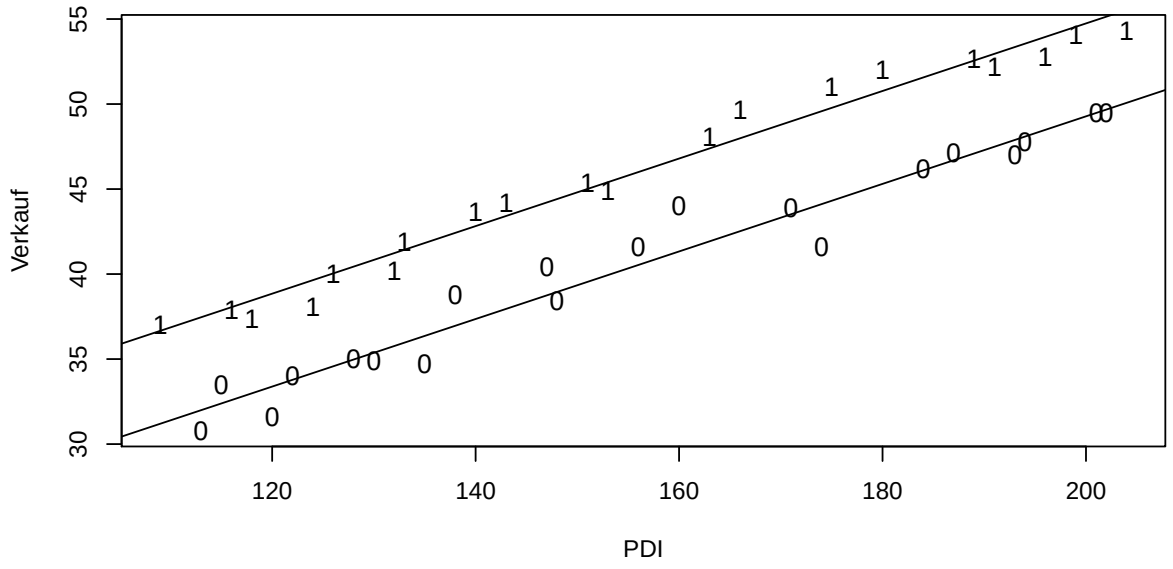


Wenn wir die Residuen gegen die Zeit auftragen und die Korrelogramme betrachten, sehen wir, dass deutliche Korrelationen vorkommen.



Bemerkung: Wir haben hier ein schönes Beispiel, bei dem der Durbin-Watson Test versagt, da die erste Autokorrelation praktisch Null ist. Zur Erinnerung: der Durbin-Watson-Test prüft eben nur die Existenz einer ersten Autokorrelation und nicht grundsätzlich ob Abhängigkeiten vorliegen.

Wir könnten nun versuchen, mit der Cochrane-Orcutt Methode die Korrelationsstruktur zu berücksichtigen. Da wir den Skiverkauf betrachten, könnte eine saisonale Abhängigkeit (Sommer/Winter) existieren. In derAbbildung unten sind die Quartalsdaten nach Saison (Winter: 1 und Sommer: 0) markiert.



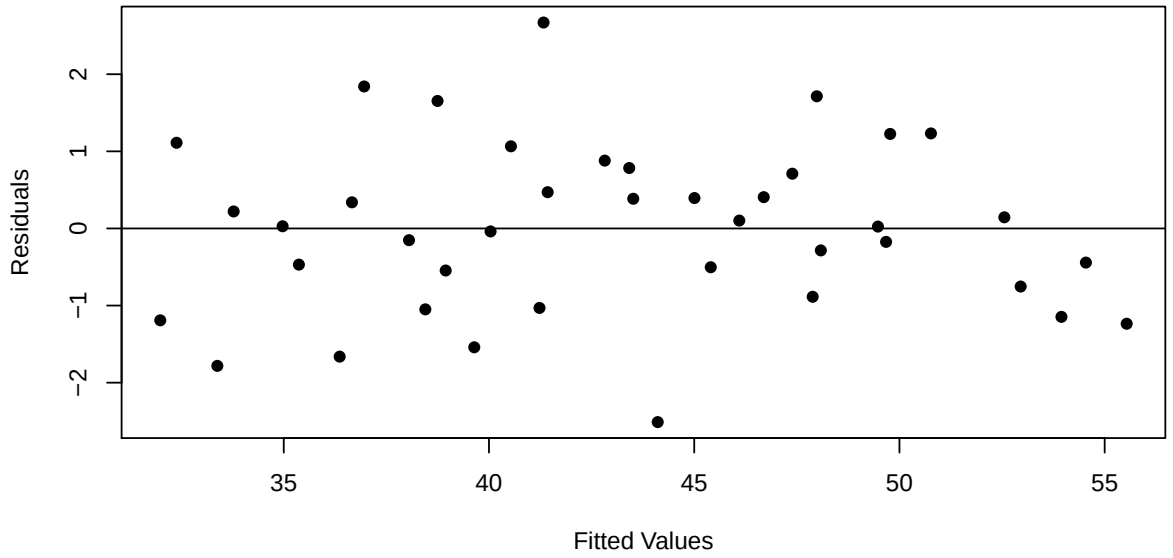
Wir erweitern deshalb unser Modell mit der zusätzlichen Saisonvariablen $\boldsymbol{z_t}$ ($\boldsymbol{z_t} = \mathbf{1}$ für die Winterquartale und $\boldsymbol{z_t} = \mathbf{0}$ für die Sommerquartale)

$$\boldsymbol{Y_t} = \boldsymbol{\beta_0} + \boldsymbol{\beta_1} \cdot \boldsymbol{x_t} + \boldsymbol{\beta_2} \cdot \boldsymbol{z_t} + \boldsymbol{E_t} \tag{61}$$

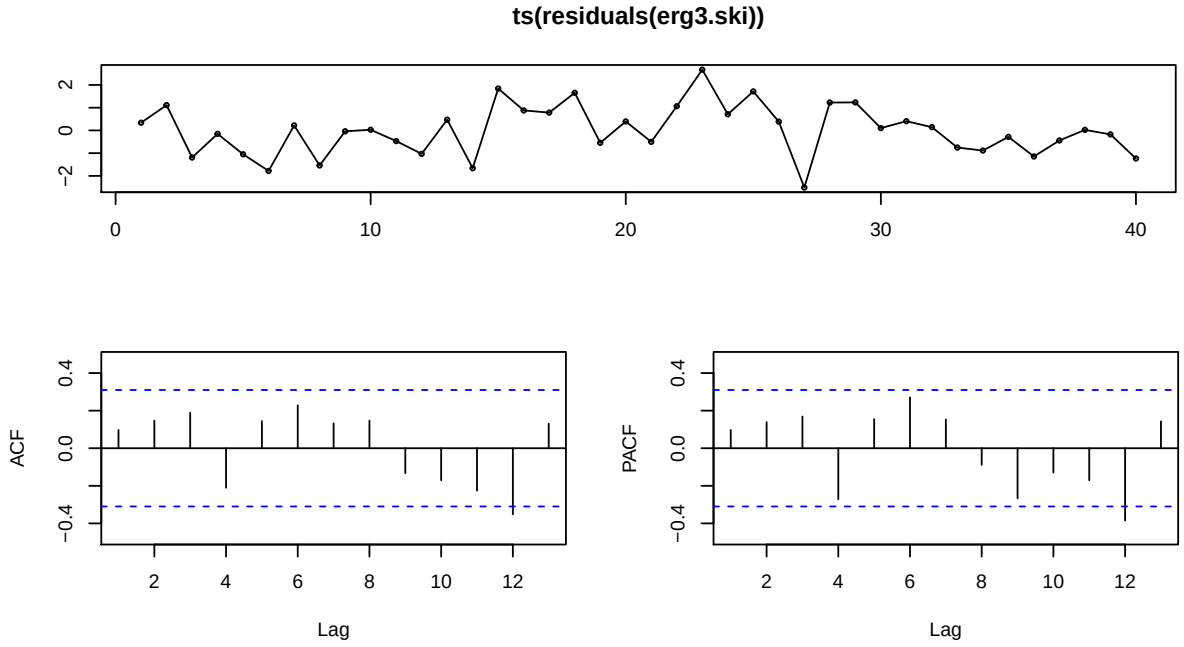
d.h. $\boldsymbol{\beta_2}$ schätzt den Unterschied zwischen dem Verkauf im Sommer und Winter.

```
> erg3.ski <- lm(Verkauf~PDI+Saison,data=Ski)
> summary(erg3.ski)

Call:
lm(formula = Verkauf ~ PDI + Saison, data = Ski)

Residuals:
    Min      1Q  Median      3Q     Max
-2.5112 -0.7864  0.0263  0.7284  2.6704

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.54020    0.97483    9.79  8.2e-12 ***
PDI          0.19868    0.00604   32.91  < 2e-16 ***
Saison       5.46434    0.35968   15.19  < 2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 1.14 on 37 degrees of freedom
Multiple R-Squared: 0.972,      Adjusted R-squared: 0.971
```

```
F-statistic:  653 on 2 and 37 DF,  p-value:    0
```



Die Residuenplots zeigen keine unerwünschten Strukturen. Auch die Autokorrelation in den Residuen ist verschwunden. In Abbildung sind die beiden Geraden eingezeichnet. Beachten Sie, dass unser Modell (61) für beide Geraden dieselbe Steigung $\boldsymbol{\beta_1}$ vorsieht. Dies erscheint aufgrund der Abbildung auch gerechtfertigt.



Die Hinzunahme der Saison als zusätzliche erklärende Variable hat die Autokorrelation der Residuen entfernt. Wir haben nun ein einfaches Modell, das die Daten gut beschreibt und sicherlich einfacher zu interpretieren ist (insbesondere für Vorhersagen), als ein Modell mit einer Korrelationsstruktur in den Residuen.

Allgemeine Bemerkung
Der Einbezug von korrelierten Fehlern sollte erst erfolgen, wenn vorher sorgfältig abgeklärt wurde, ob nicht wichtige Variablen im Modell fehlen oder ob eine erklärende Variable im Modell ungenügend berücksichtigt wurde, z.B. weitere lagged-Variables hinzufügen. Falls die Korrelationen in den Residuen durch Hinzunahme von erklärenden Variablen nicht entfernt werden kann, müssen wir mit den entsprechenden Methoden bessere Schätzer und korrekte Vertrauensintervalle berechnen.

8.5 Allgemeiner Fall, verallgemeinerte Kleinste-Quadrate

In diesem Kapitel wollen wir etwas näher auf die verallgemeinerte Kleinste-Quadrate-Lösung eingehen. Dabei wollen wir eine beliebige Korrelationsstruktur der Regressionsfehler zulassen, wie dies grundsätzlich auch in anderen Gebieten und Analysen (z.B. bei der Varianzanalyse) auftreten kann. Das methodische Vorgehen ist gleich wie bei der Cochrane-Orcutt-Methode, d.h. man transformiert das Regressionsmodell so, dass im neuen Modell die Fehler unkorreliert sind.

Zunächst schreiben wir das Modell (57) in Matrixform:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{E}. \tag{62}$$

wie Sie es vom Regressionsblock her kennen.
Dabei sind $\boldsymbol{Y}$ und $\boldsymbol{E}$ Spaltenvektoren der Länge $\boldsymbol{N}$ und $\boldsymbol{X} = (\boldsymbol{x_{t,j}})$ eine $\boldsymbol{N} \times \boldsymbol{q}$ Matrix.
Der Kleinste-Quadrate-Schätzer ist gegeben durch

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T y}.$$

Daraus folgt mit etwas linearer Algebra, dass er erwartungstreu ($\mathrm{E}(\widehat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$) ist und die Kovarianzen sind gleich

$$\mathrm{Var}(\widehat{\boldsymbol{\beta}}) = \mathrm{E}((\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})^{\boldsymbol{T}}) = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T V X} (\boldsymbol{X^T X})^{-1}, \tag{63}$$

wobei $\boldsymbol{V}$ die $\boldsymbol{N} \times \boldsymbol{N}$ Matrix der Kovarianzen der $\boldsymbol{E_t}$ ist, d.h. $\boldsymbol{V_{t,s}} = \mathrm{Cov}(\boldsymbol{E_t}, \boldsymbol{E_s})$. Im Falle von stationären Residuen können wir schreiben $\boldsymbol{V_{t,s}} = \boldsymbol{\sigma^2 \rho(t - s)}$.

Wenn die Fehler $\boldsymbol{E_t}$ **unkorreliert** sind, d.h. wenn $\boldsymbol{V} = \boldsymbol{\sigma^2 I}$ ist, dann erhalten wir für die Varianz von $\widehat{\boldsymbol{\beta}}$ die aus der Regression bekannte Form $\mathrm{Var}(\widehat{\boldsymbol{\beta}}) = \boldsymbol{\sigma^2} (\boldsymbol{X^T X})^{-1}$.

Die Formel (63) ist zwar nicht leicht verständlich, man sieht aber wenigstens, dass die Korrelationen der Fehler $\boldsymbol{E_t}$ in der Formel für die Varianzen und Kovarianzen des Schätzers $\widehat{\boldsymbol{\beta}}$ vorkommen. Bei Kenntnis der Korrelationsstruktur der Fehler können wir also mit (63) korrekte Standardfehler für $\widehat{\boldsymbol{\beta}}$ berechnen.

Dass der Effekt der Korrelationen beachtlich sein kann, sieht man schon im folgenden einfachen Beispiel des **Arithmetischen Mittels**

Für $\boldsymbol{q = 1}$ und $\boldsymbol{x_t^{(1)} = 1}$ hat man das Modell

$$\boldsymbol{Y_t = \beta_1 + E_t}$$

Der Kleinste-Quadrate-Schätzer für $\boldsymbol{\beta_1}$ ist dann $\widehat{\boldsymbol{\beta}}_{\boldsymbol{1}} = \frac{1}{N}\sum_{t=1}^{N} \boldsymbol{Y_t}$, also einfach das arithmetische Mittel der Beobachtungen. Für den Erwartungswert gilt

$$\mathrm{E}(\widehat{\boldsymbol{\beta}}_{\boldsymbol{1}}) = \frac{\boldsymbol{1}}{\boldsymbol{N}} \sum_{\boldsymbol{t=1}}^{\boldsymbol{N}} \mathrm{E}(\boldsymbol{Y_t}) = \boldsymbol{\beta_1}$$

und für die Varianz

$$\mathrm{Var}\widehat{\boldsymbol{\beta}} = \frac{\boldsymbol{\sigma^2}}{\boldsymbol{N^2}} \sum_{\boldsymbol{t=1}}^{\boldsymbol{N}} \sum_{\boldsymbol{s=1}}^{\boldsymbol{N}} \boldsymbol{\rho(t-s)} = \frac{\boldsymbol{\sigma^2}}{\boldsymbol{N}}(\boldsymbol{1} + \boldsymbol{2} \sum_{\boldsymbol{k=1}}^{\boldsymbol{N-1}} \boldsymbol{\rho(k)(1 - k/N)).}$$

Wir gehen nun zurück zum allgemeinen Fall und versuchen auch bessere Schätzer für die Koeffizienten zu finden. Wenn die Korrelationsmatrix $\boldsymbol{V}$ bekannt ist, können wir durch eine **geeignete Transformation des Modells** (62) die Korrelationen der Fehler zum Verschwinden bringen. Da $\boldsymbol{V}$ als Kovarianzmatrix symmetrisch ist, kann man eine Matrix $\boldsymbol{Z}$ finden, so dass man $\boldsymbol{V}$ schreiben kann als $\boldsymbol{V = ZZ^T}$ (Choleski-Zerlegung, siehe lineare Algebra, ohne Beweis). Multipliziert man in (62) auf beiden Seiten von links mit $\boldsymbol{Z^{-1}}$, so erhält man das Modell

$$\boldsymbol{Z^{-1}Y = Z^{-1}X\beta + Z^{-1}E}$$

resp.

$$\widetilde{\boldsymbol{Y}} = \widetilde{\boldsymbol{X}}\boldsymbol{\beta} + \widetilde{\boldsymbol{E}} \tag{64}$$

mit $\widetilde{\boldsymbol{Y}} = \boldsymbol{Z^{-1}Y}$, $\widetilde{\boldsymbol{X}} = \boldsymbol{Z^{-1}X}$ und $\widetilde{\boldsymbol{E}} = \boldsymbol{Z^{-1}E}$. Für die transformierten Fehler $\widetilde{\boldsymbol{E}}$ gilt nun

$$\mathrm{Var}\widetilde{\boldsymbol{E}} = \mathrm{Var}\boldsymbol{Z^{-1}E} = \boldsymbol{Z^{-1}}\mathrm{Var}\boldsymbol{E}(\boldsymbol{Z^{-1}})^{\boldsymbol{T}} = \boldsymbol{Z^{-1}V}(\boldsymbol{Z^{-1}})^{\boldsymbol{T}} = \boldsymbol{I},$$

d.h. $\mathrm{Cov}(\widetilde{\boldsymbol{E}}_{\boldsymbol{t}}, \widetilde{\boldsymbol{E}}_{\boldsymbol{s}}) = \boldsymbol{1}$ für $\boldsymbol{t = s}$ und 0 für $\boldsymbol{t \neq s}$. Die Fehler $\widetilde{\boldsymbol{E}}$ sind also unkorreliert. Der Kleinste-Quadrate-Schätzer von $\boldsymbol{\beta}$ im Modell (64) ist somit gegeben durch

$$\widetilde{\boldsymbol{\beta}} = (\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\widetilde{\boldsymbol{X}})^{-1}\widetilde{\boldsymbol{X}}^{\boldsymbol{T}}\widetilde{\boldsymbol{y}} = (\boldsymbol{X^TV^{-1}X})^{-1}\boldsymbol{X^TV^{-1}y.}$$

$\widetilde{\boldsymbol{\beta}}$ hat minimale Varianz unter den linearen, erwartungstreuen Schätzern. Diese ist gegeben durch

$$\mathrm{Var}\widetilde{\boldsymbol{\beta}} = \mathrm{E}((\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta})(\widetilde{\boldsymbol{\beta}} - \boldsymbol{\beta})^{\boldsymbol{T}}) = (\boldsymbol{X^TV^{-1}X})^{-1}, \tag{65}$$

wie man leicht nachprüft.

Den Schätzer $\widetilde{\boldsymbol{\beta}}$ nennt man **verallgemeinerter Kleinste-Quadrate-Schätzer** oder **Generalized Least Squares Estimator (GLS)**. Im Falle von normalverteilten $\boldsymbol{E_t}$ kann man zeigen, dass dies gerade der Maximum-Likelihood Schätzer ist.

All diese Formeln haben den gewichtigen **Nachteil**, dass man die Korrelationsmatrix $\boldsymbol{V}$ praktisch nie kennt. Wir können also weder $\widetilde{\boldsymbol{\beta}}$ noch die Kovarianzmatrizen von $\widehat{\boldsymbol{\beta}}$ oder $\widetilde{\boldsymbol{\beta}}$ einfach berechnen. Grundsätzlich stehen zwei Methoden zur Verfügung.

Simultane Schätzung

Man optimiert die Likelihood-Funktion (wie dies die Funktion `gls` macht) simultan bezüglich der unbekannten Steigungskoeffizienten und der unbekannten Parameter der Korrelationsstruktur.

Schrittweises Vorgehen

Zuerst berechnet man den gewöhnlichen Kleinste-Quadrate-Schätzer $\widehat{\boldsymbol{\beta}}$ und bestimmt die Residuen $\boldsymbol{e_t} = \boldsymbol{y_t} - \sum_j \widehat{\boldsymbol{\beta}}_j \boldsymbol{x_{t,j}}$. Anschliessend passt man ein Modell an die Residuen $\boldsymbol{e_t}$ an (z.B. ARMA) und setzt die Kovarianzen des angepassten Modells in die obigen Formeln ein. Dabei macht man natürlich einen Fehler, da diese Kovarianzen nie exakt gleich den wahren Kovarianzen sind, aber man hofft, dass dieser Fehler klein ist.

Im Kapitel 8.2 haben wir dieses Verfahren im Fall, wo die Fehler $\boldsymbol{E_t}$ einem AR-Modell genügen, näher untersucht. Dies wird als **Cochrane-Orcutt-Methode** bezeichnet. Insbesondere war es dort für die Berechnung von $\widetilde{\boldsymbol{\beta}}$ nicht notwendig, die Matrix $\boldsymbol{V}$ explizit zu berechnen oder zu invertieren.