Cheat Sheet (für Fachkollegen)

Moritz Schrom

15 Jänner 2023

Contents

Aufbau eines Berichts für Fachkollegen

Im Folgenden wird auf die Abschnitte die ein Bericht für Fachkollegen normalerweise enthält genauer eingegangen. Siehe Skriptum 15.2.1 Aufbau eines statistischen Berichts.

Ausgangssituation

Beschreibung der Ausgangssituation, inklusive Stichprobenumfang und Merkmalen samt Skalenniveau.

Kurze Beschreibung des Ziels der Analyse, und der dahinterliegenden Fragestellung.

Datenmanagement

In welcher Form liegen die Daten vor? Wie wurden diese eingelesen?

Gibt es fehlende Werte und wie wurden diese behandelt?

Visualisierung

Deskriptive Analyse der Stichprobe.

Diagramm(e) samt Beschreibung: Gibt es Auffälligkeiten?

Tabellen, Kennzahlen, samt verbaler Beurteilung dieser.

Eventuelle ableitung allfälliger Hypothesen

Überprüfung der Fragestellungen (Tests, Konfidenzintervalle, Modelle, etc.)

Bei Tests:

- Angabe von Testbezeichnung
- Begründung der Testauswahl (z.B. robuste Methoden bei Ausreißern etc.)
- Null- und Alternativhypothese
- Signifikanzniveau
- P-Wert
- Formale Schlussfolgerung (Vergleich P-Wert mit Signifikanzniveau)
- Inhaltliche Schlussfolgerung

Bei Modellen:

- Begründung für die Wahl des Modells (z.B. abhängig vom Skalenniveau der abhängigen und unabhängigen Variablen)
- Schätzung des Modells
- Bei Regressionsmodellen: Ableitung der Modellformel, sowie Interpretation des Einflusses der unabhängigen Variablen auf die abhängige Variable ("Effektstärke"). Unterstützung via Effektplot bzw. Modellplot.
- Beurteilung der Signifikanz der Parameter, sowie des Gesamtmodells.
- Beurteilung der Modellgüte (z.B. Rˆ2 bei linearer Regression)
- Modelldiagnostik (Normalverteilung der Residuen, Restzusammenhang der Residuen mit den geschätzten Werten)
- Diagramm mit erwarteten Werten (z.B. Regressionsgerade) samt Konfidenzbändern
- Vorhersagen (erwartete Werte, bzw. Prognoseintervalle für Einzelwerte)

Fazit

Zusammenfassung der Ergebnisse.

Diskussion alffälliger Probleme.

Generalisierbarkeit der Ergebnisse.

Einlesen von Daten

Daten können in R auf unterschiedliche Arten eingelesen werden, je nach vorliegendem Quellformat (CSV-Datei, Reintext, …) und gewünschtem Zielformat (Vektor, Dataframe, …).

Im Folgenden werden einige Möglichkeiten des Einlesens von Daten inklusive Beispielen demonstriert und erläutert.

Einlesen von Daten mit `scan()`

Siehe RDocumentation - scan: Read Data Values

Die `scan()` Funktion in *R* ließt Text (entweder in Form einer String-Variable oder in Form einer Textdatei) ein, und wandelt diese in eine Liste oder einen Vektor um.

Angenommen die Daten liegen in Form einer Textdatei `browser.txt` vor:

```
Liste an Browsern
Firefox Firefox IE Firefox IE IE IE Firefox IE
IE Chrome IE Chrome Firefox Firefox Chrome Firefox IE
Firefox Opera Chrome IE Firefox Firefox IE Chrome IE
IE IE IE IE Chrome Safari Safari Safari IE
Safari Safari Chrome Opera Opera Chrome Chrome Opera Firefox
Firefox Firefox Chrome Firefox Chrome IE Chrome Firefox Firefox
Firefox Chrome Chrome Chrome IE Chrome
```

Mit Hilfe der `scan()` Funktion kann diese simple Textdatei eingelesen werden. Der `what` Parameter den Datentyp der zu lesenden Daten an. Mit dem `skip` Parameter können Zeilen am Beginn der Textdatei beim Einlesen übersprungen werden.

```
daten = scan(file="browser.txt", what="A", skip=1)
head(daten)
```

```
## [1] "Firefox" "Firefox" "IE"      "Firefox" "IE"      "IE"
```

```
as.factor(daten)
```

```
##  [1] Firefox Firefox IE      Firefox IE      IE      IE      Firefox IE
## [10] IE      Chrome  IE      Chrome  Firefox Firefox Chrome  Firefox IE
## [19] Firefox Opera   Chrome  IE      Firefox Firefox IE      Chrome  IE
## [28] IE      IE      IE      IE      Chrome  Safari  Safari  Safari  IE
## [37] Safari  Safari  Chrome  Opera   Opera   Chrome  Chrome  Opera   Firefox
## [46] Firefox Firefox Chrome  Firefox Chrome  IE      Chrome  Firefox Firefox
## [55] Firefox Chrome  Chrome  Chrome  IE      Chrome
## Levels: Chrome Firefox IE Opera Safari
```

Alternativ kann der `scan()` Funktion auch Text übergeben werden.

```
daten = scan(text="1 5 3 2 1 6 3 2 5 3 1 2", what=1)
head(daten)
```

```
## [1] 1 5 3 2 1 6
```

Einlesen von Daten mit `read.table()`

Siehe RDocumentation - read.table: Data Input

Oft haben wir es mit tabellarischen Daten zu tun, welche mehrere Merkmale beinhalten. Um mit solchen Datenstrukturen zu arbeiten bietet *R* DataFrames an. Daten können mit Hilfe der `read.table()` Funktion als DataFrame eingelesen werden.

Angenommen, die Daten liegen in Form einer CSV-Datei `bugfixes.csv` vor:

```
"dauer" "programmierer" "bugtyp"
120 "Eckkrammer" "NA"
185 "Fischer" "Reporting"
174 "Meyer" "DB"
...
```

Mit Hilfe der `read.table()` Funktion kann die CSV-Datei als DataFrame eingelesen werden. Im Beispiel wird mit dem Paramter `header` angegeben ob die CSV-Datei Überschriften beinhaltet, der Paramter `sep` spezifiziert das Zeichen durch welches Felder begrenzt werden, mit dem Paramter `quote` werden gültige Quoting Zeichen angegeben. Bei den meisten CSV Files sind die Standardwerte für die Paramter, welche in der Dokumentation nachgelesen werden können, passend.

```
daten = read.table(file="bugfixes.csv", header=TRUE, sep=" ", quote="\"")
head(daten)
```

```
##   dauer programmierer    bugtyp
## 1   120    Eckkrammer      <NA>
## 2   185       Fischer Reporting
## 3   174         Meyer        DB
## 4   188         Meyer       GUI
## 5   161         Mandl       GUI
## 6   163       Fischer Reporting
```

Unterschiedliche Tests

Chi-squared Test

Mit dem Chi-squared Test können Daten auf ihre Gleichverteilung getestet werden.

Siehe RDocumentation - chisq.test: Pearson's Chi-squared Test for Count Data

t-Test

Überprüft Daten auf ihre zentrale Gleichverteilung ums arithmethische Mittel. Bei Daten mit großer Streuung (durch Ausreißer) sollte der Wilcoxon Test herangezogen werden, da er robuster ist.

Siehe RDocumentation - t.test: Student's t-Test

Wilcoxon Test

Überprüft Daten auf ihre zentrale Gleichverteilung um den Median und stellt damit eine robuste Alternative zum t-Test dar.

Siehe RDocumentation - wilcox.test: Wilcoxon Rank Sum and Signed Rank Tests

Medmed (Median Absolute Deviation)

Ein robustes Streuungsmaß. Der Median der Abweichungen zum Median.

Siehe RDocumentation - mad: Median Absolute Deviation

Levene Test

Überprüft die Varianzhomogenität, und sollte durchgeführt werden bevor eine Varianzanalyse angewendet wird. Ist er nicht signifikant, kann mit der Varianzanalyse fortgefahren werden.

Siehe RDocumentation - leveneTest: Levene's Test

Eine kategoriale Variable

Es liegt eine Stichprobe mit 60 Beobachtungen und einem kategorialen Merkmal mit den Ausprägungen: *Firefox*, *IE*, *Chrome*, *Opera* und *Safari* vor. Es soll untersucht werden, ob die Kateogiren gleichverteilt sind.

Einlesen der Daten als Vektor:

```
daten = scan(file="browser.txt", what="A")
browser = as.factor(daten)
head(browser)
```

```
## [1] Liste    an       Browsern Firefox  Firefox  IE
## Levels: an Browsern Chrome Firefox IE Liste Opera Safari
```

Überprüfen auf fehlende Werte:

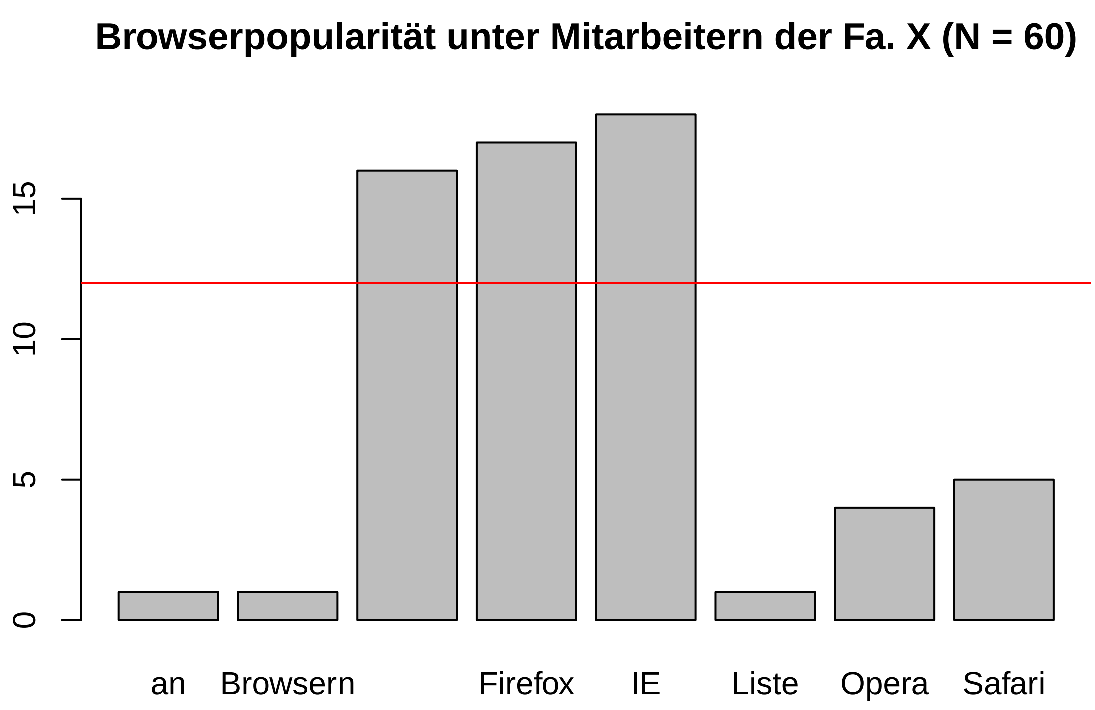
```r
sum_na = sum(is.na(browser))
sprintf("Es gibt %d fehlende Werte", sum_na)
```

```
## [1] "Es gibt 0 fehlende Werte"
```

Visualisieren der absoluten Häufigkeiten:

```r
browser_abs = table(browser)
browser_rel = round(100 * prop.table(browser_abs), 1)

barplot(browser_abs, main="Browserpopularität unter Mitarbeitern der Fa. X (N = 60)")
abline(h=12, col="red")
```



Durchführen des Chi-squared Tests:

```r
chisq.test(browser_abs)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  browser_abs
## X-squared = 52.937, df = 7, p-value = 3.816e-09
```

Die Nullhypothese lautet: Das Merkmal *Browser* ist gleichverteilt. Der Wert der Teststatistik beträgt 15.83, der dazugehörige P-Wert 3.816e-09. Der Wert liegt unter dem Signifikanzniveau von 0.05, die Nullhypothese muss damit verworfen werden.

Gesmatfazit: Die Umfragedaten sprechen gegen die Gleichverteilung der Browser unter allen Mitarbeitern.

Zwei kategoriale Variablen

Es liegt eine Stichprobe mit 60 Beobachtungen und drei Kategorialen Merkmalen *Browser*, *Email* und *Abteilung* vor. Untersucht werden soll ein möglicher Zusammenhang zwischen *Browser* und *Email*, sowie zwischen *Browser* und *Abteilung*. Die Daten liegen in Form einer CSV-Datei vor.

Einlesen der Daten als DataFrame:

```
umfrage = read.table(file="umfrage.csv", sep=",", header=TRUE)
summary(umfrage)
```

```
##    browser             email             abteilung
##  Length:60          Length:60          Length:60
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

Überprüfen auf fehlende Werte:

```
sum_na = sum(is.na(umfrage))
sprintf("Es gibt %d fehlende Werte", sum_na)
```

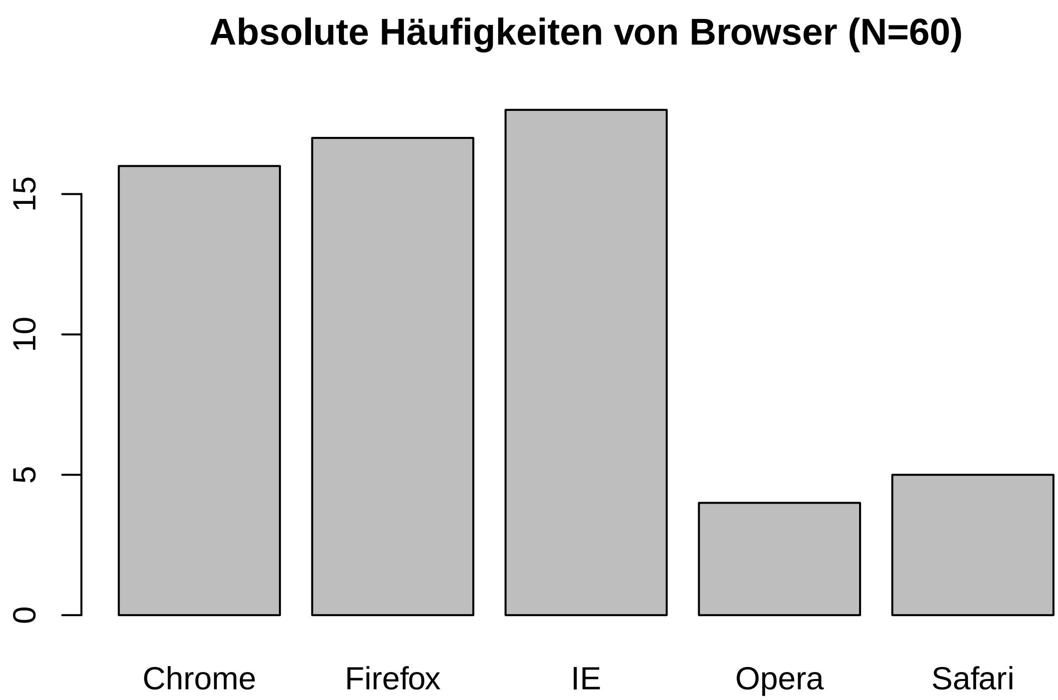
```
## [1] "Es gibt 0 fehlende Werte"
```

Visualisieren der Einzelmerkmale:

```
browser = table(umfrage$browser)
browser
```

```
##
##  Chrome Firefox      IE   Opera  Safari
##      16      17      18       4       5
```

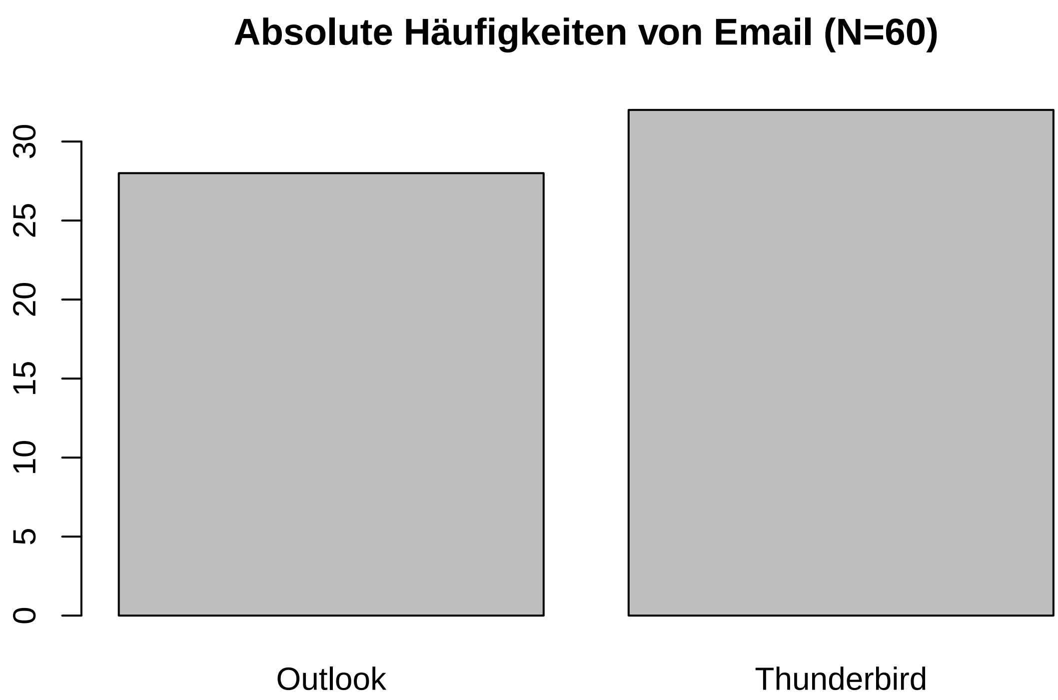
```
barplot(browser, main = "Absolute Häufigkeiten von Browser (N=60)")
```




```
email = table(umfrage$email)
email
```

```
##
##     Outlook Thunderbird
##          28          32
```

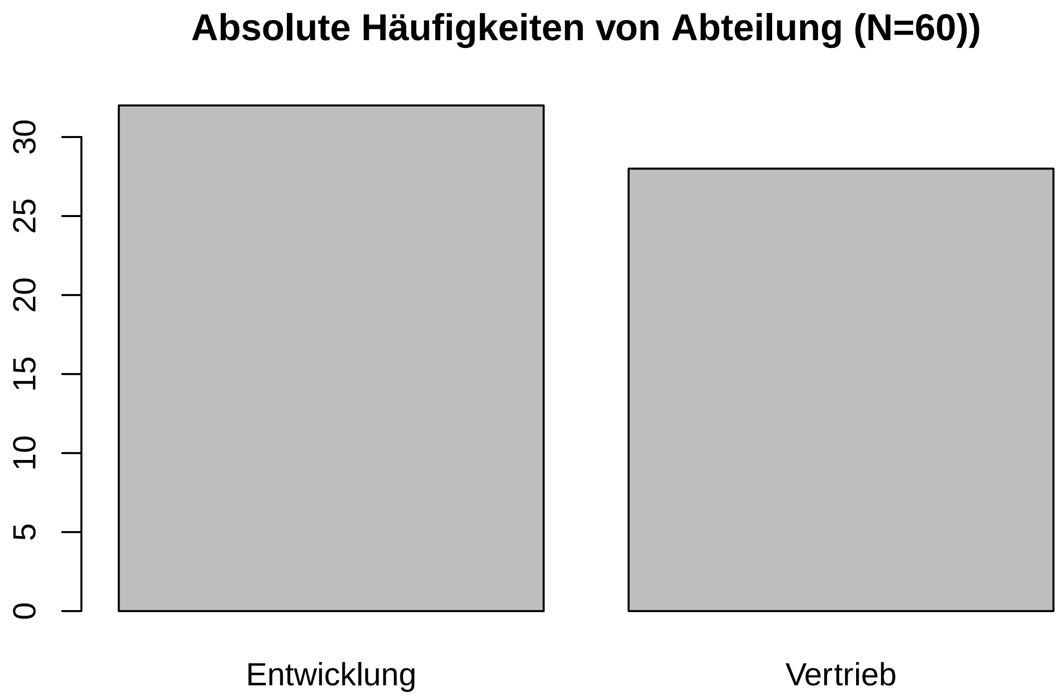
```
barplot(email, main = "Absolute Häufigkeiten von Email (N=60)")
```




```
abteilung = table(umfrage$abteilung)
abteilung
```

```
##
## Entwicklung    Vertrieb
##          32          28
```

```
barplot(abteilung, main = "Absolute Häufigkeiten von Abteilung (N=60))")
```



Es handelt sich um zwei Merkmale,die pro Respondent erhoben wurden. Keines der Merkmale stellt ein offensichtliches Gruppierungsmerkmal dar. Es handelt sich daher um ein *Unabhängigkeitsproblem*.

Ermitteln der Kontingenztabelle, sowohl absolut als auch relativ:

```
tab_email_browser = with(umfrage, table(email, browser))
library(pander)
pander(tab_email_browser, justify = "right", emphasize.rownames = FALSE)
```

	Chrome	Firefox	IE	Opera	Safari
Outlook	11	9	5	1	2
Thunderbird	5	8	13	3	3

```
tab_anteile = round(prop.table(tab_email_browser), 2)
pander(tab_anteile, justify = "right", emphasize.rownames = FALSE)
```

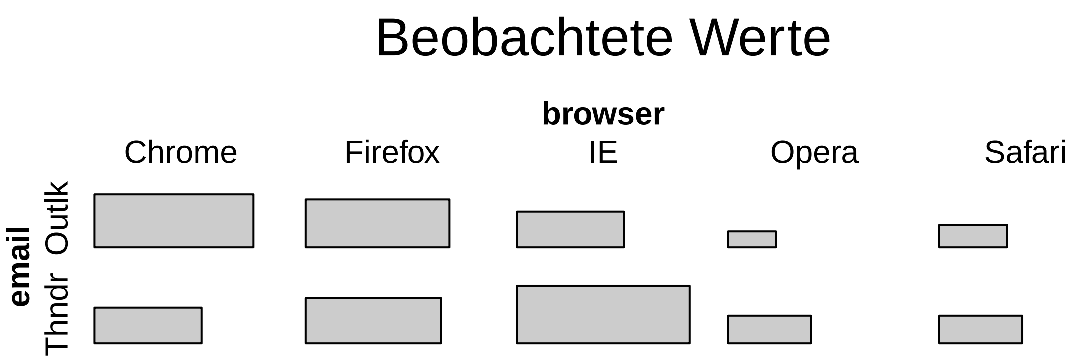
	Chrome	Firefox	IE	Opera	Safari
Outlook	0.18	0.15	0.08	0.02	0.03
Thunderbird	0.08	0.13	0.22	0.05	0.05

Visualisieren der beobachteten und erwarteten Werte mit einem Kacheldiagramm:

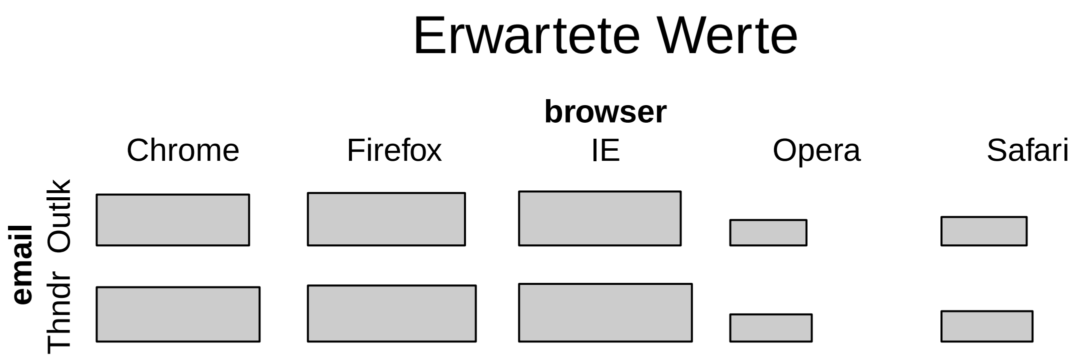
```
library(vcd)
```

```
## Loading required package: grid
```

```
tile(tab_email_browser, main = "Beobachtete Werte", abbreviate = c(email = 5))
```




```
tile(tab_email_browser, type = "expected", main = "Erwartete Werte", abbreviate = c(email = 5))
```



Zusammenhang zwischen Browser und Email

Um die Unabhängigkeit von *Browser* und *Email* zu testen, wird ein Chi-squared Test durchgeführt. Auch hier lautet die Nullhypothese: *Browser* und *Email* sind gleicherteilt. Das Signifikanzniveau wird mit 0.05 festgelegt.

```
chisq.test(tab_email_browser)
```

```
## Warning in chisq.test(tab_email_browser): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_email_browser
## X-squared = 6.8281, df = 4, p-value = 0.1453
```

```
# Aufgrund der Warnmeldung (zuwenig erwartete Werte in manchen Zellen) wird mittels simuliertem P-Wert
chisq.test(tab_email_browser, simulate.p.value = TRUE, B = 10000)
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 10000
##  replicates)
```

```
##
## data:  tab_email_browser
## X-squared = 6.8281, df = NA, p-value = 0.1437
```

Der P-Wert liegt mit 0.15 über dem Signifikanzniveau von 0.05, damit kann die Nullhypothese *nicht verworfen* werden. Mit den Daten ist kein Zusammenhang zwichen *Browser* und *Email* nachweisbar.

Zusammenhang zwischen Browser und Abteilung

Bei der zweiten Fragestellung ist zu untersuchen, ob die Browser in den beiden Abteilungen gleich populär sind. *Abteilung* stellt ein Gruppierungsmerkmal dar, es handelt sich daher um ein *Homogenitätsproblem*.

Ermitteln der Kontingenztabelle:

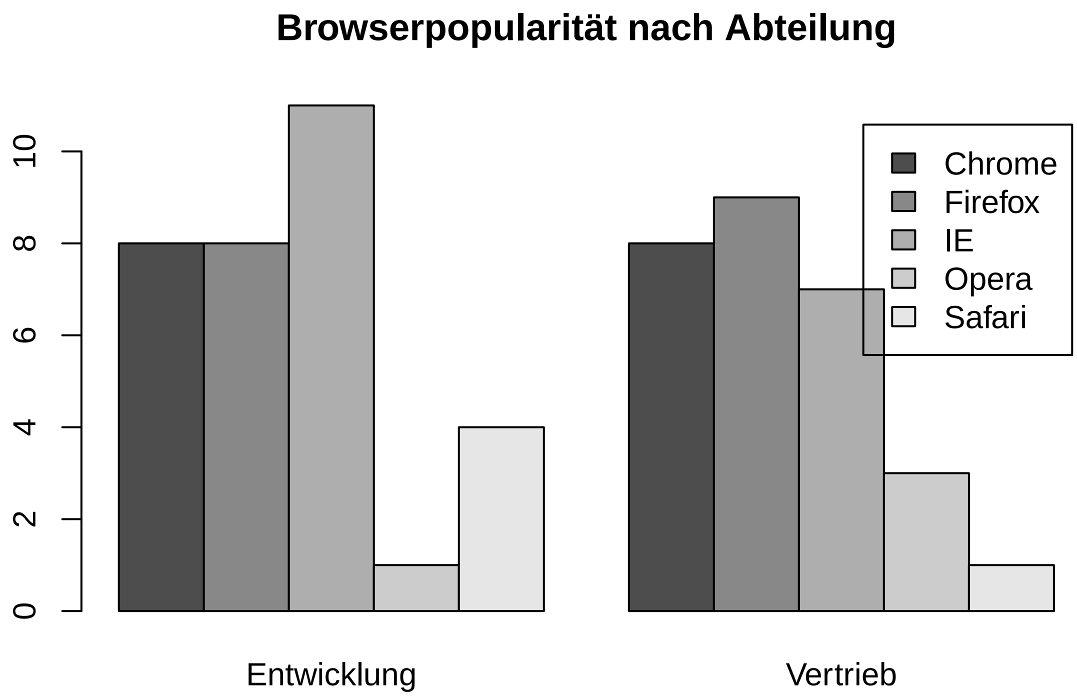
```
tab_abteilung_browser = with(umfrage, table(abteilung, browser))
pander(tab_abteilung_browser, justify="right", emphasize.rownames=FALSE)
```

	Chrome	Firefox	IE	Opera	Safari
Entwicklung	8	8	11	1	4
Vertrieb	8	9	7	3	1

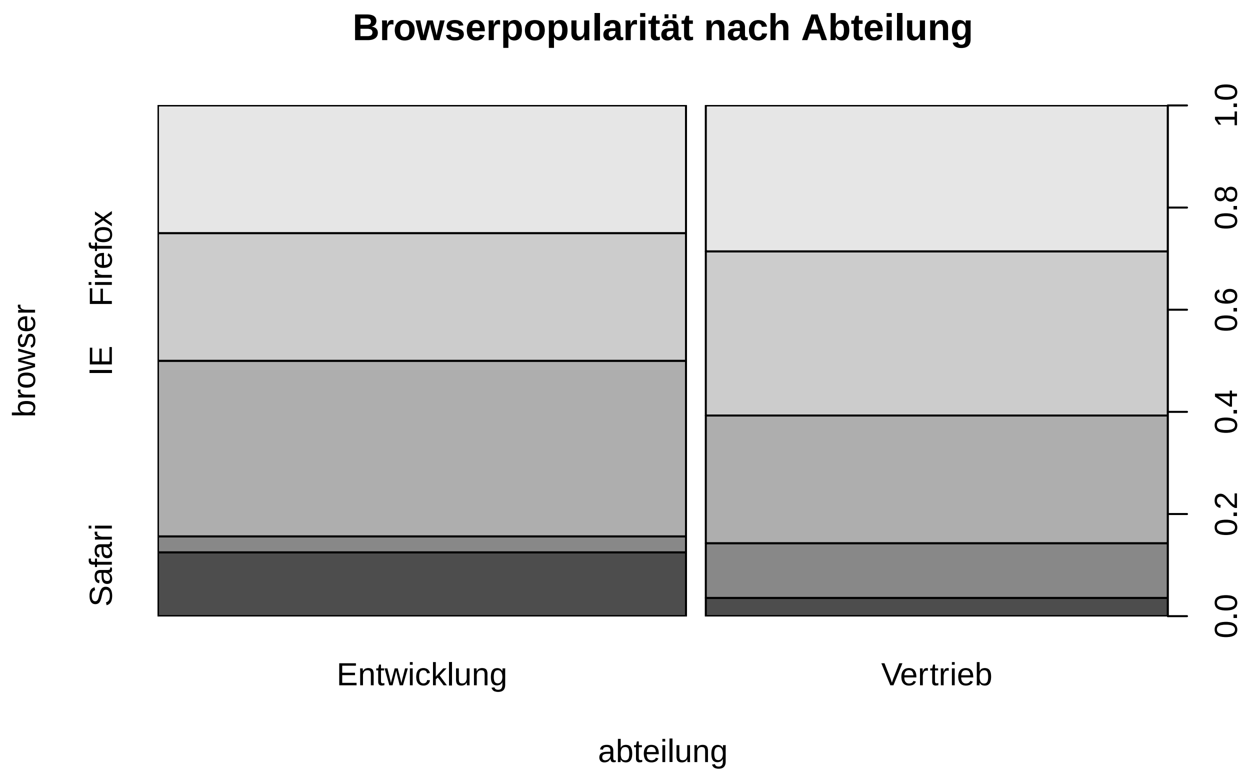
```
tab_abt_rel = round(prop.table(tab_abteilung_browser, 1), 2)
pander(tab_abt_rel, justify="right", emphasize.rownames=FALSE)
```

	Chrome	Firefox	IE	Opera	Safari
Entwicklung	0.25	0.25	0.34	0.03	0.12
Vertrieb	0.29	0.32	0.25	0.11	0.04

```
barplot(t(tab_abteilung_browser), main="Browserpopularität nach Abteilung", beside=TRUE, legend=TRUE)
```




```
spineplot(tab_abteilung_browser, main="Browserpopularität nach Abteilung")
```



Um zu überprüfen ob die in der Stichprobe beobachteten Unterschiede auch auf die Grundgesmtheit zutreffen, wird ein Chi-squared Test mit Signifikanzniveau 0.05 durchgeführt:

```
chisq.test(tab_abteilung_browser)
```

```
## Warning in chisq.test(tab_abteilung_browser): Chi-squared approximation may be
## incorrect
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab_abteilung_browser
## X-squared = 3.4966, df = 4, p-value = 0.4784
```

```
# Aufgrund der Warnmeldung (zuwenig erwartete Werte in manchen Zellen) wird mittels simuliertem P-Wert
chisq.test(tab_abteilung_browser, simulate.p.value = TRUE, B = 10000)
```

```
##
##  Pearson's Chi-squared test with simulated p-value (based on 10000
##  replicates)
##
## data:  tab_abteilung_browser
## X-squared = 3.4966, df = NA, p-value = 0.5014
```

Der P-Wert liegt weit über dem Signifikanzniveau von 0.05, die Homogenitätshypothese wird also beibehalten: mit den Vorliegenden Daten kann *kein* unterschiedliche Browser-Popularität in den beiden Abteilungen *Entwicklung* und *Vertrieb* nachgewiesen werden.

Eine metrische Variable

Es liegt eine Stichprobe mit 22 Beobachtungen, davon ein fehlender Wert. Erhoben wurde ein metrisches Merkmal, die Bugfixdauer in Minuten. Vermutet wird eine mittlere Dauer von 170 Minutne in der Population.

Die Daten liegen als Textdatei `bugfix.txt` vor:

```
66 174 188 161 157
178 143 184 159 178
157 183 152 174 189
176 174 177 176 159
207 ??
```

Daher werden sie mit `scan()` eingelesen:

```
dauer = scan("bugfix.txt", na.strings="??")
head(dauer)
```

```
## [1]  66 174 188 161 157 178
```

Überprüfen auf fehlende Werte:

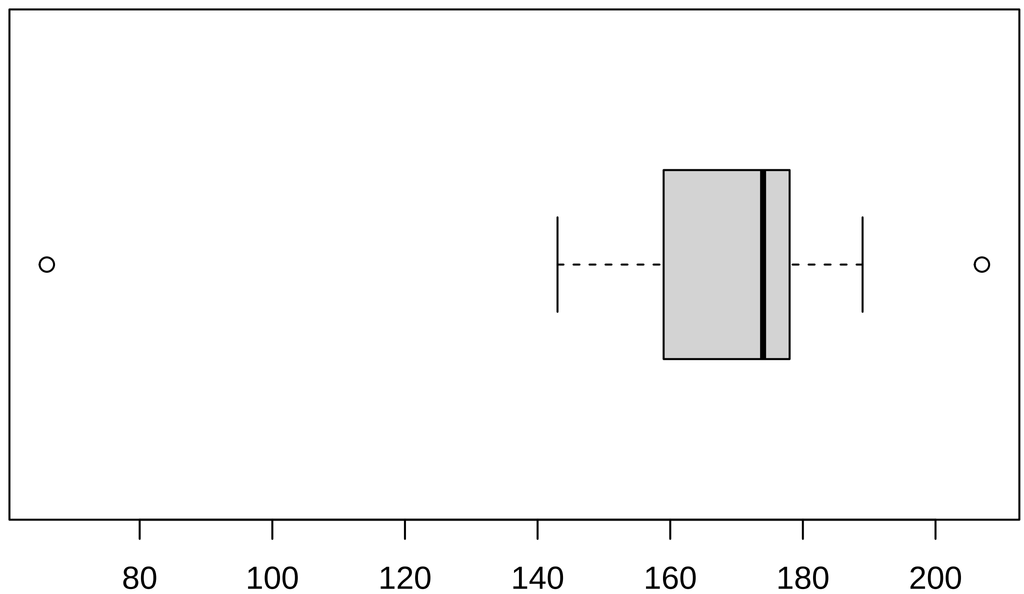
```
sum_na = sum(is.na(dauer))
sprintf("Es gibt %d fehlende Werte", sum_na)
```

```
## [1] "Es gibt 1 fehlende Werte"
```

Die *deskriptive Beurteilung* ist rein beschreibend. Im folgenden werden die Daten mittels Histogramm und Boxplot visualisiert:

```
hist(dauer, freq = F)
rug(dauer)
```


```
boxplot(dauer, horizontal=TRUE)
```



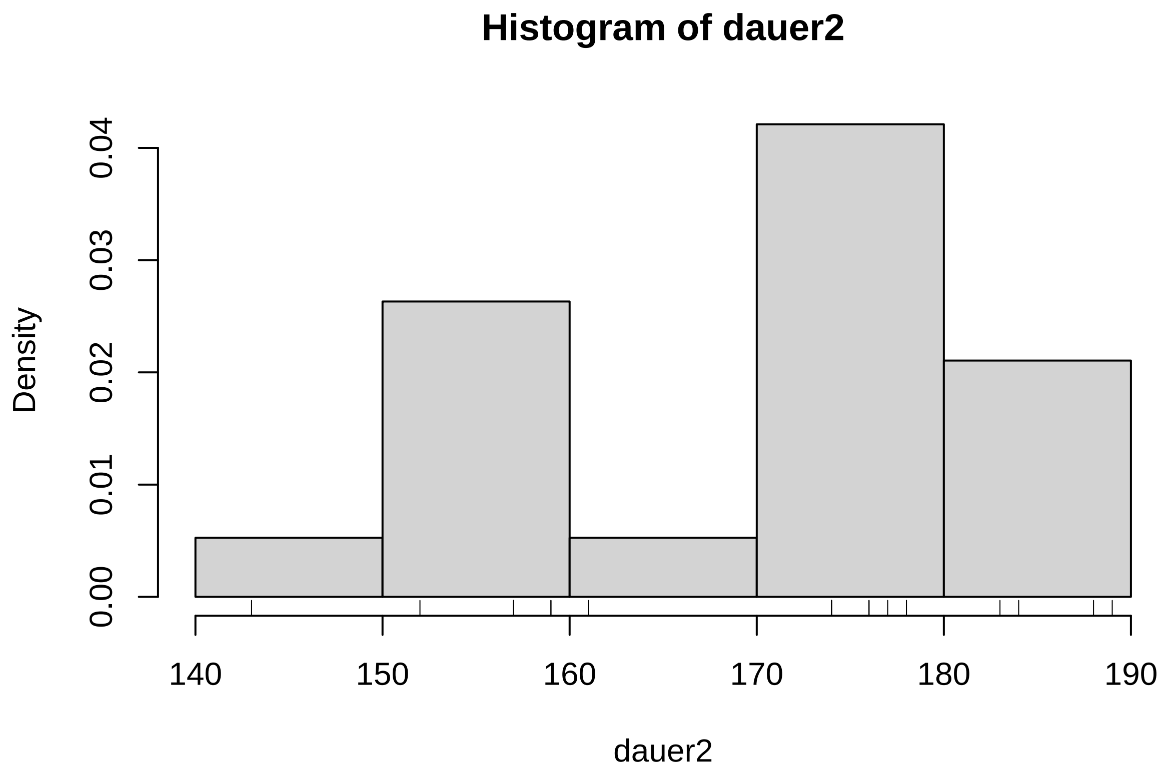
Die 5-Punkt Zusammenfassung liefert:

```
summary(dauer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    66.0   159.0   174.0   167.2   178.0   207.0       1
```

Im Boxplot sind Ausreißer zu sehen, welche unser Histogramm verzerren. Daher wird das Histogramm ohne Ausreißer erneut gezeichnet:

```
dauer2 = dauer[dauer > 70 & dauer < 200]
hist(dauer2, freq = FALSE)
rug(dauer2)
```



Die Form des Histogramms kann als *mehrgipfelig* beschrieben werden (Häufungspunkte in den Bereichen [150, 160] und [170, 190]). Die Daten sind eher im rechten Bereich (ab 170) konzentriert.

Dem Boxplot entnimmt man ebenfalls, dass die Verteilung **linksschief** ist (Median in den rechten Bereich der Box verschoben). 50% der Daten liegen im Bereich 159 bis 178 (Boxgrenzen = 1. und 3. Quartil), der Median bei 174. Aufgrund der Schiefe weicht er deutlich vom arithmetischen Mittel (167.2) ab. Die beiden Ausreißer bilden Minimum (66) und Maximum (207) der Verteilung, die Spannbreite daher 141.

```
sd(dauer, na.rm = TRUE)
```

```
## [1] 27.46435
```

```
mad(dauer, na.rm = TRUE)
```

```
## [1] 19.2738
```

Das Merkmal hat eine Standardabweichung von 27.46 vom Mittel 167.2, das robuste Streumaß Medmed (Median der Abweichungen vom Median) beträgt hingegen nur 19.27. Die Standardabweichung ist durch die Ausreißer stark verzerrt.

Schlussfolgerung

Aufgrund der Ausreißer kann kein Konfidenzintervall für den Mittelwert ermittelt, oder t-Test durchgeführt werden. Stattdessen wird ein Wilcoxon-Test mit der Nullhypothese: "Der Median in der Population beträgt 170" auf dem Signifikanzniveau 0.05 durchgeführt:

```r
wilcox.test(dauer, mu = 170, conf.int = TRUE)
```

```
## Warning in wilcox.test.default(dauer, mu = 170, conf.int = TRUE): cannot compute
## exact p-value with ties
```

```
## Warning in wilcox.test.default(dauer, mu = 170, conf.int = TRUE): cannot compute
## exact confidence interval with ties
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  dauer
## V = 118.5, p-value = 0.9307
## alternative hypothesis: true location is not equal to 170
## 95 percent confidence interval:
##  163.0 178.5
## sample estimates:
## (pseudo)median
##          170.5
```

Der P-Wert ist mit 0.93 weit über dem Signifikanzniveau, und kann nicht verworfen werden. Die Warnung kann aufgrund der Deutlichkeit des Ergebnisses ignoriert werden. Ein Konfidenzintervall für den Median beträgt [164, 178.5] - in diesem kann der wahre Median vermutet werden, ohne dass die vorliegenden Daten dagegen sprechen. Es enthält insbesondere die vermuteten 170 Minuten.

Fazit: Die Daten sprechen nicht gegen die Annahme, dass der Median der Bugfixdauer in der Grundgesamtheit (alle Bugs) 170 Minuten beträgt.

Zwei metrische Variablen (Korrelation)

Für 21 Bugs wurden vier metrische Merkmale erhoben:

- Behebungsdauer in Minuten (*dauer*)
- Anzahl der Codezeilen (*codelines*)
- Anzahl der Use Cases (*usecases*)
- Alter des Programmierers (*alter*)

Ziel ist der Nachweis eines möglichen (linearen) Zusammenhangs zwischen der Behebungsdauer und der Programmgröße (Codezeilenanzahl)

Es liegt erneut ein Datensatz zum Thema Bugfixes als CSV-Datei `bugfixes2.csv` vor:

```
"dauer" "codelines" "usecases" "alter"
120 183000 49 35
174 386000 86 44
188 467000 95 37
...
```

Einlesen mit `read.table()`:

```r
bugfixes = read.table("bugfixes2.csv", header = TRUE)
head(bugfixes)
```

```
##   dauer codelines usecases alter
## 1   120    183000       49    35
## 2   174    386000       86    44
## 3   188    467000       95    37
## 4   161    309000       70    35
## 5   157    305000       71    21
## 6   178    243000       85    36
```

5-Punkt-Zusammenfassung:

```r
summary(bugfixes)
```

```
##      dauer         codelines         usecases         alter
##  Min.   :120.0   Min.   :183000   Min.   :49.00   Min.   :21.0
##  1st Qu.:159.0   1st Qu.:305000   1st Qu.:73.00   1st Qu.:31.0
##  Median :174.0   Median :364000   Median :79.00   Median :35.0
##  Mean   :169.8   Mean   :355286   Mean   :77.48   Mean   :34.9
##  3rd Qu.:178.0   3rd Qu.:401000   3rd Qu.:85.00   3rd Qu.:39.0
##  Max.   :207.0   Max.   :521000   Max.   :97.00   Max.   :47.0
```

Stichprobengröße:

```r
sum(complete.cases(bugfixes))
```

```
## [1] 21
```

Bevor ein möglicher Zusammenhang untersucht wird, sollten die Daten inkl. Merkmale immer vorab deskriptiv untersucht werden:

```
summary(bugfixes$dauer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   120.0   159.0   174.0   169.8   178.0   207.0
```

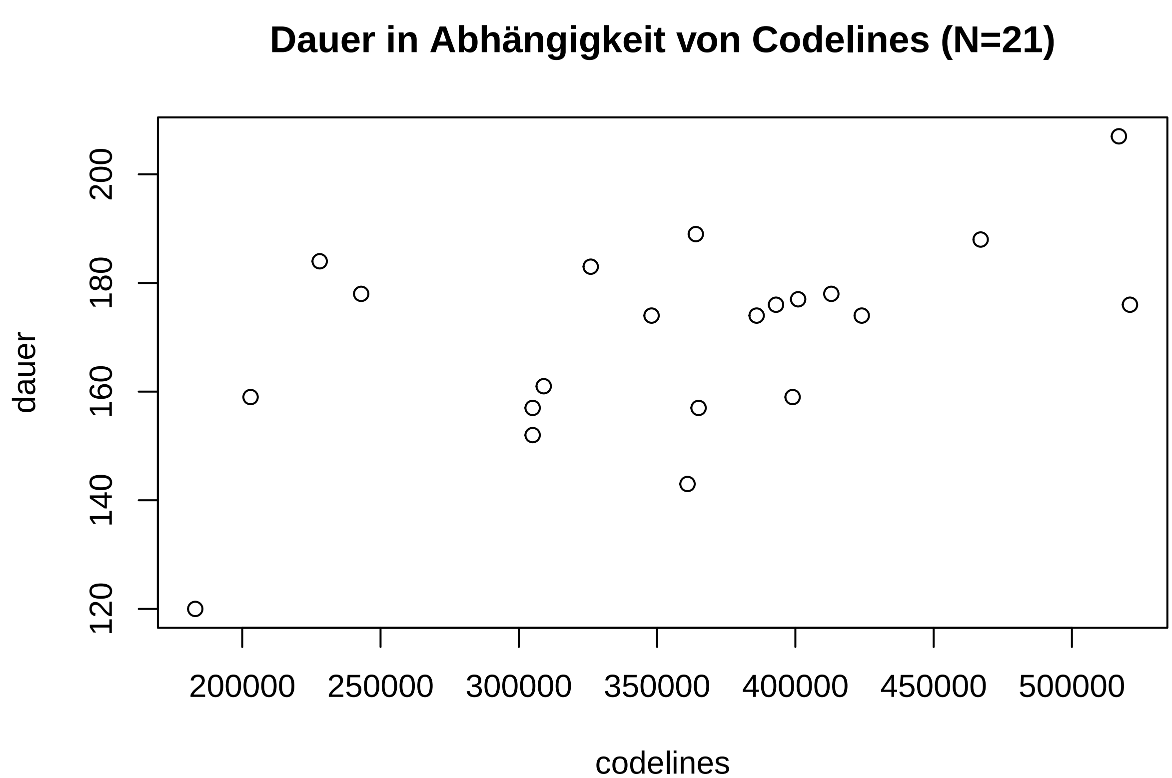
```
summary(bugfixes$codelines)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  183000  305000  364000  355286  401000  521000
```

Mit Hilfe von Boxplots kann auch ein emögliche mehrgipfeligkeit oder Schiefe der Daten (linksschief, rechtsschief, symmetrisch) untersucht werden. Darauf wird hier verzichtet, für mehr Informationen siehe *Eine metrische Variable*.

Die gemeinsame Verteilung der Variablen kann als Streudiagramm dargestellt werden:

```
plot(dauer ~ codelines, data=bugfixes, main="Dauer in Abhängigkeit von Codelines (N=21)")
```



Das Diagram lässt schon eine positive Korrelation vermutet. Um den Datensatz auf eine Korrelation zu untersuchen, werden Tests durchgeführt (zuerst nach Pearson, anschließend nach Spearman).

Die Nullhypothese lautet jeweils. Die Korrelation der Population beträgt 0, die Alternative: Sie ist ungleich 0. Das Signifikanzniveau betrgät 0.05.

Korrelation nach Pearson:

```
cor.test(~ dauer + codelines, data=bugfixes)
```

```
##
##  Pearson's product-moment correlation
##
## data:  dauer and codelines
## t = 2.8854, df = 19, p-value = 0.009477
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1579245 0.7943803
## sample estimates:
##       cor
## 0.5519806
```

Die P-Wert beträgt 0.0094 und liegt unter dem Signifikanzniveau von 0.05. Die Nullhypothese muss demnach verworfen werden: die Daten sprechen für eine Korrelation (nach Pearson). Das 95%-Konfidenzintervall beträgt [0.157, 0.794], die Schätzung von 0.55 ist damit sehr ungenau: der Bereich lässt Werte von "nicht korreliert" bis "stark korreliert" zu.

Korrelation nach Spearman:

Die Korrelation naach Spearman basiert auf Rängen, und ist bei einer so starken Streuung wie sie vorliegt ein robustes Maß.

```
cor.test(~ dauer + codelines, data = bugfixes, method = "spearman", )
```

```
## Warning in cor.test.default(x = mf[[1L]], y = mf[[2L]], ...): Cannot compute
## exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  dauer and codelines
## S = 898.62, p-value = 0.06038
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.4164778
```

Der P-Wert beträgt 0.06 und liegt knapp über dem Signifikanzniveau von 0.05. Die Nullhypothese kann demnach nicht verworfen werden. Die Daten sprechen nicht für eine Korrelation (nach Spearman).

Fazit: Bei Verwendung eines robusten Korrelationsmaßes (Spearman) kann kein Zusammenhang zwischen der Programmgröße und der Bugfixdauer nachgewiesen werden. Bei der Korrelation nach Pearson ist der Zusammenhang schwach und die Schätzung – aufgrund des geringen Stichprobenumfangs und der hohem Streuung – sehr ungenau.

Zwei metrische Variablen (Regresison)

Es liegt derselbe Datensaz `bugfixes2.csv` wie beim Beispiel für Korrelation vor (Siehe *Zwei metrische Variablen (Korrelation)*).

Ziel der Untersuchung ist die Ermittlung eines Prognosemodells für die Behebungsdauer in Abhängigkeit der Codezeilenzahl sowie der Erstellung von Vorhersagen für 200.000 und 600.000 Codezeilen.

Erneut werden die Daten mit `read.table()` eingelesen:

```
options(scipen = 9999)
bugfixes = read.table("bugfixes2.csv", header = TRUE)
head(bugfixes)
```

```
##   dauer codelines usecases alter
## 1   120    183000       49    35
## 2   174    386000       86    44
## 3   188    467000       95    37
## 4   161    309000       70    35
## 5   157    305000       71    21
## 6   178    243000       85    36
```

Auf die deskriptive Analyse, welche hier für die Einzelmerkmale folgen sollte wird verzichtet. Für mehr Informationen siehe *Zwei metrische Variablen (Korrelation)* oder *Eine metrische Variable*.

Modellschätzung

Mit `lm()` wird ein lineares Modell angepasst. Mit `summary()` kann eine Modellübersicht erstellt werden:

```
model = lm(dauer ~ codelines, data = bugfixes)
summary(model)
```

```
##
## Call:
## lm(formula = dauer ~ codelines, data = bugfixes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.639 -12.214   1.768   6.136  28.354
##
## Coefficients:
##                Estimate   Std. Error t value     Pr(>|t|)
## (Intercept) 130.27545317  14.13511560   9.216 0.0000000193 ***
## codelines     0.00011127   0.00003856   2.885      0.00948 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.92 on 19 degrees of freedom
## Multiple R-squared:  0.3047, Adjusted R-squared:  0.2681
## F-statistic: 8.326 on 1 and 19 DF,  p-value: 0.009477
```

Sowohl Achsenabschnitt, wie auch der Koeffizient *codelines* sind signifikant auf dem 0.05-Niveau (Der P-Wert beträgt 0.009477). Das Bestimmtheitsmaß *R-squared* beträgt 0.30, das bedeutet das Modell erklärt rund 30% der Varianz von *dauer*.

Die Regressionsgleichung lautet:

```
E(dauer|codelines) = 130.28 + 0.00011 codelines
```

Ermitteln deer 95%igen Konfidenzintervalle für die Paramter:

```r
round(confint(model), 5)
```

```
##                 2.5 %    97.5 %
## (Intercept) 100.69032 159.86059
## codelines     0.00003   0.00019
```

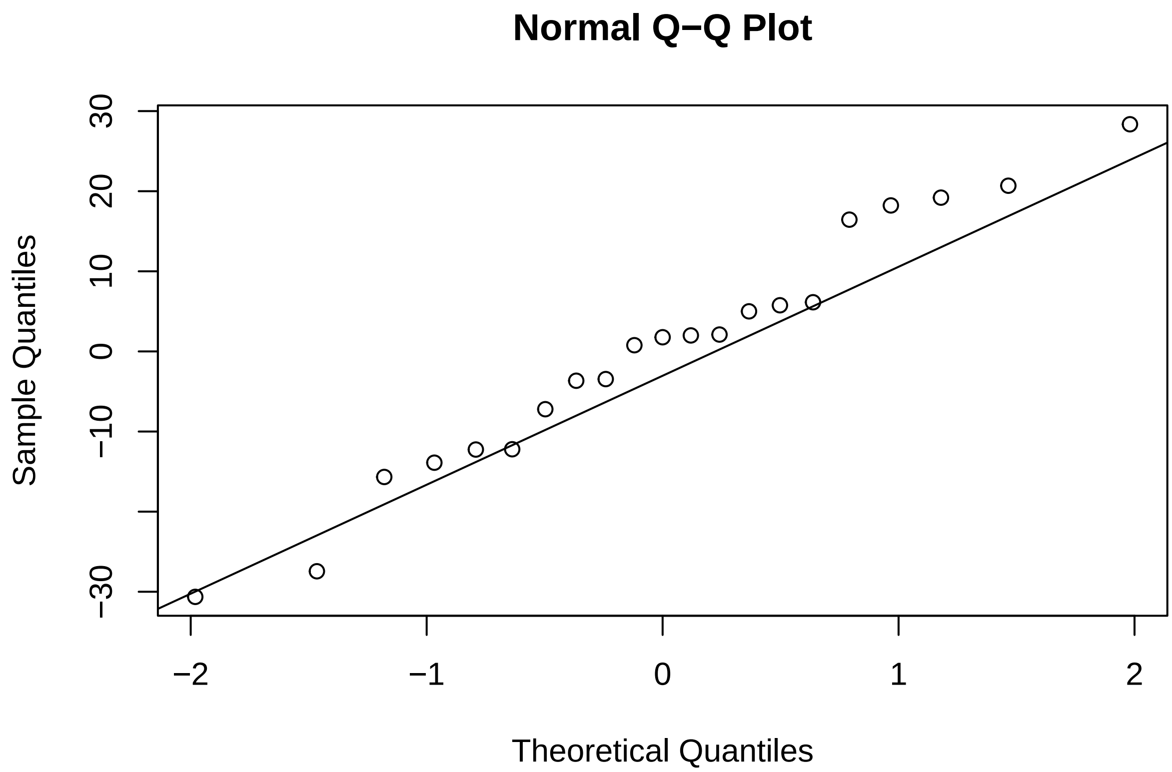
Die paramter weißen eine hohe Schwankungsbreite auf: Der Grundaufwand beträgt zwischen 100 und 160 Minuten, und für jeweils 100.000 Codezeilen können zwischen 3 und 19 Minuten hinzukommen. Die Schätzung ist also sehr ungenau.

Modelldiagnostik

Vor der Prognose ist die Gültigkeit der Modellannahme zu testen.

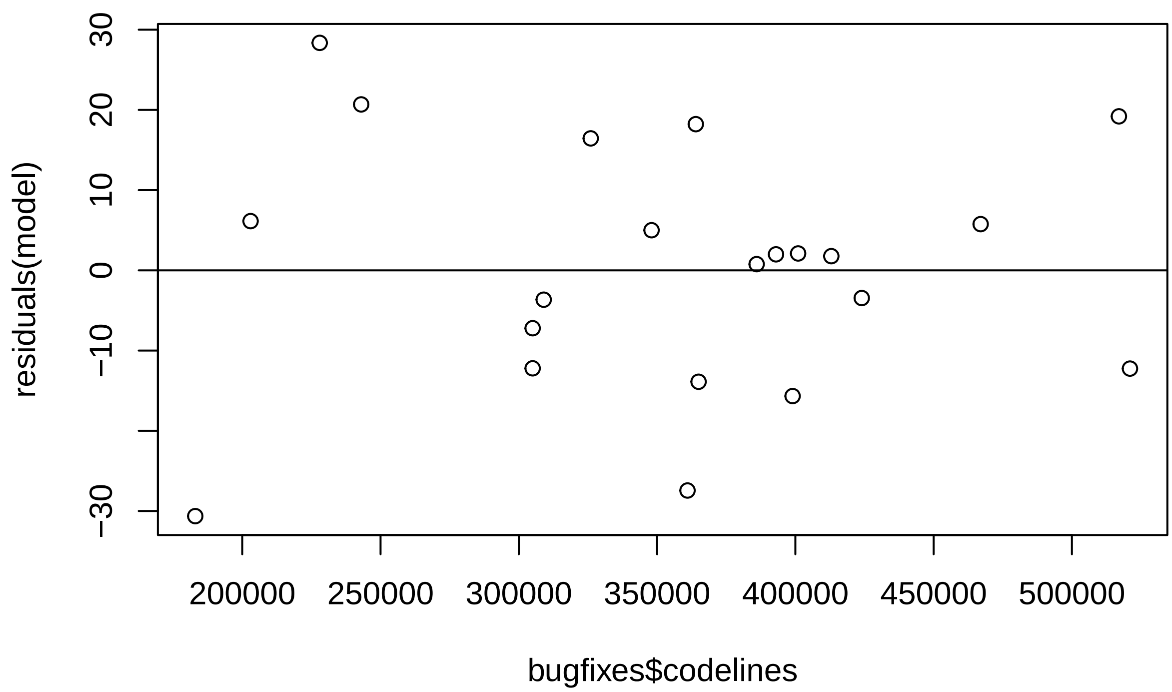
Kontrollieren der Normalverteilung der Residuen, mit einem Q-Q-Plot gegen die theorethischen Werte der Normalverteilung.

```r
qqnorm(residuals(model))
qqline(residuals(model))
```



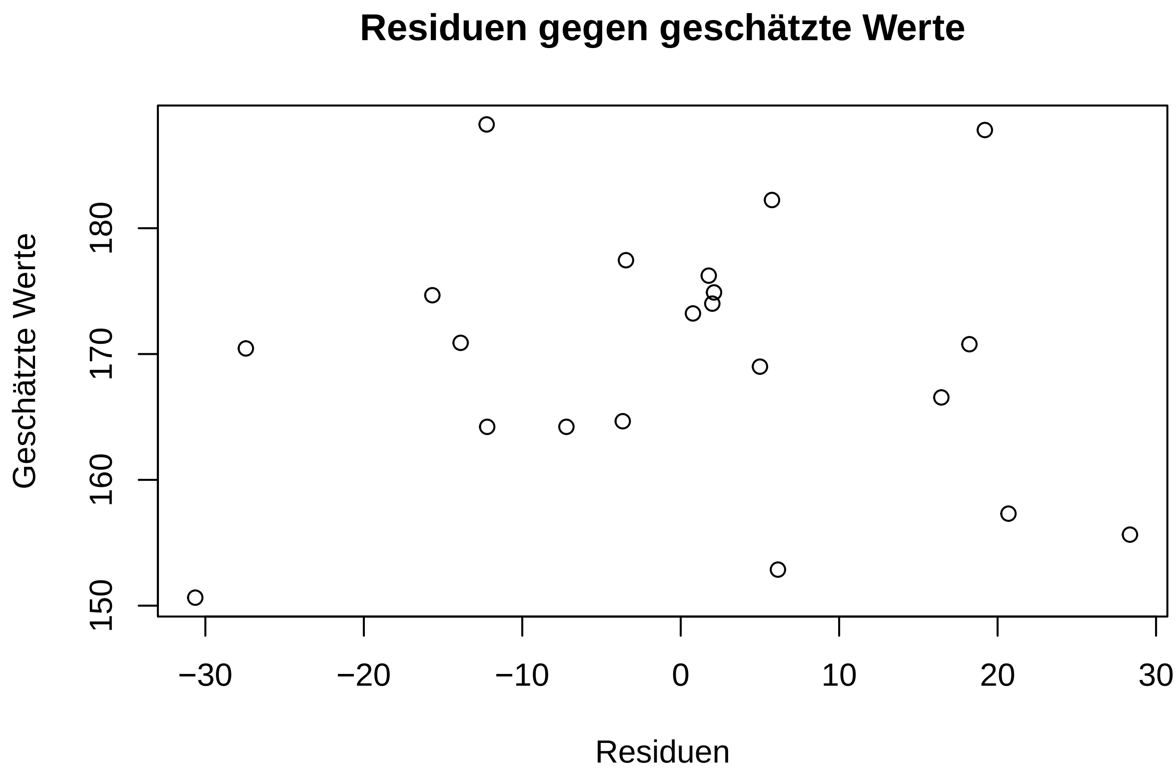
Untersuchung der Varianzhomogenität der Residuen: Sind die Varianzen gleich? Schwer zu beurteilen aufgrund des geringen Stichprobenumfangs.

```
plot(residuals(model) ~ bugfixes$codelines)
abline(h = 0)
```



Korrekte Modellspezifikation durch Streudiagramm gegen die geschätzen Werte: Gibt es eine Reststruktur in den Residuen? Es ist keine Struktur zu sehen, daher OK.

```
plot(fitted(model) ~ residuals(model),
     xlab = "Residuen", ylab = "Geschätzte Werte",
     main = "Residuen gegen geschätzte Werte")
```



Prognose

Es sind Vorhersagen für die 200.000 und 600.000 Codezeilen zu erstellen:

```
newdata = data.frame(codelines = c(200000, 600000))
newdata
```

```
##   codelines
## 1    200000
## 2    600000
```

Punkt- und Intervallprognosen der *erwarteten* Bugfixdauer-Werte:

```
pred = predict(model, newdata, interval = "confidence")
cbind(newdata, pred)
```

```
##   codelines      fit      lwr      upr
## 1    200000 152.5303 138.0389 167.0216
## 2    600000 197.0399 175.9910 218.0887
```

Die Intervallprognosen für die Einzelwerte lauten:

```r
pred = predict(model, newdata, interval = "prediction")[,-1]
cbind(newdata, pred)
```

```
##   codelines      lwr      upr
## 1    200000 116.1865 188.8740
## 2    600000 157.6200 236.4597
```

Das folgende Diagramm stellt das Streudiagramm samt Regressionsgerade (rot), Konfidenzbänder (blau) und Prognosekanal (violett) dar. Aufgrund der starken Streuung ist die Schätzung der Geraden, und besonders jene der Einzelprognosen, sehr ungenau.

```r
# 100 x-Werte erzeugen
x = seq(from = min(bugfixes$codelines),
        to = max(bugfixes$codelines),
        length = 100)
newdata = data.frame(codelines = x)

# Werte für Konfidenzband
konfband = predict(model, newdata, interval = "confidence")
head(konfband)
```

```
##        fit      lwr      upr
## 1 150.6386 134.9453 166.3319
## 2 151.0185 135.5688 166.4682
## 3 151.3984 136.1913 166.6055
## 4 151.7783 136.8127 166.7440
## 5 152.1582 137.4328 166.8836
## 6 152.5381 138.0517 167.0246
```

```r
# Werte für Prognosekanal
progkanal = predict(model, newdata, interval = "predict")
head(progkanal)
```

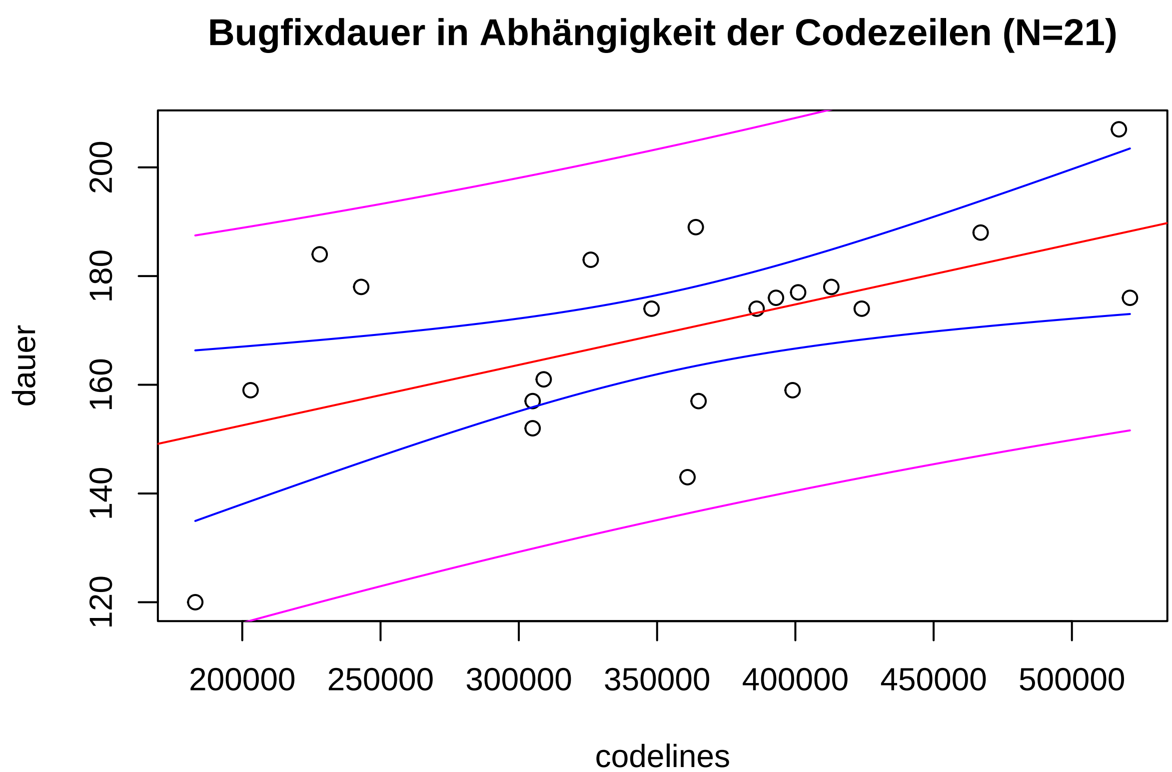
```
##        fit      lwr      upr
## 1 150.6386 113.7991 187.4781
## 2 151.0185 114.2821 187.7549
## 3 151.3984 114.7634 188.0335
## 4 151.7783 115.2428 188.3138
## 5 152.1582 115.7205 188.5960
## 6 152.5381 116.1963 188.8799
```

```r
# Streudiagramm
plot(dauer ~ codelines, data = bugfixes,
     main = "Bugfixdauer in Abhängigkeit der Codezeilen (N=21)")

# Regressionsgerade = erwartete Werte
abline(model, col = "red")

# Konfidenzband für erwartete Werte
lines(x, konfband[,"lwr"], col = "blue")
lines(x, konfband[,"upr"], col = "blue")
```

```
# Prognoskanal für Einzelwerte
lines(x, progkanal[,"lwr"], col = "magenta")
lines(x, progkanal[,"upr"], col = "magenta")
```

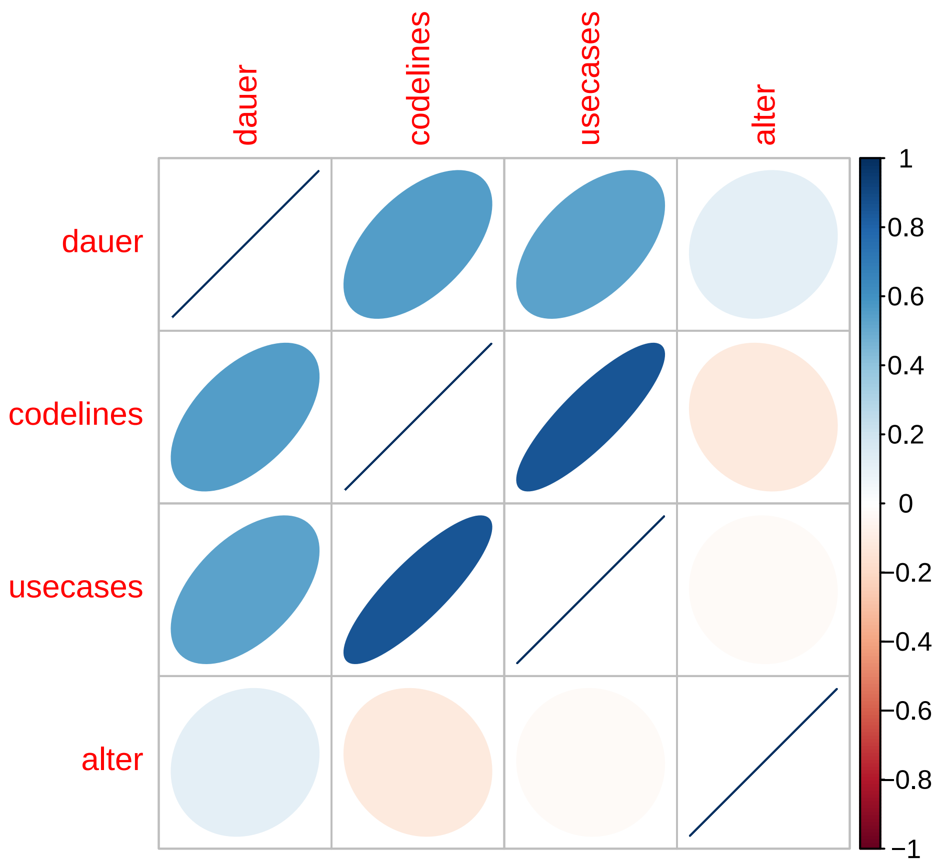


Visuelle Unterstützung für Korrelationen:

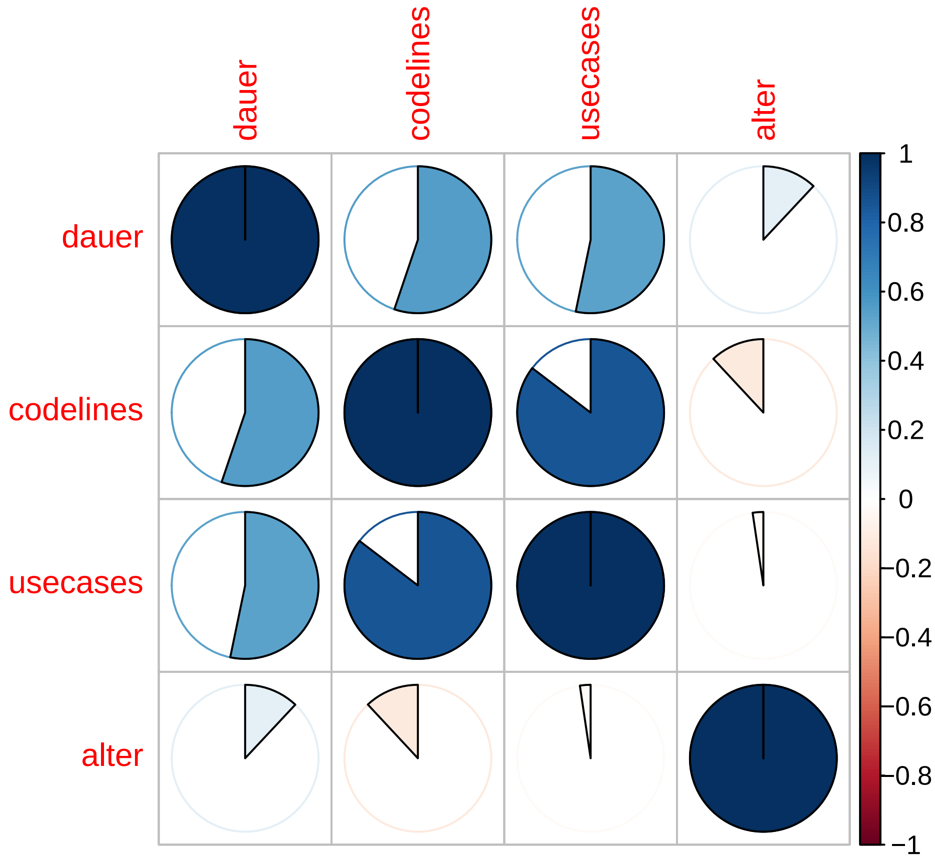
```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

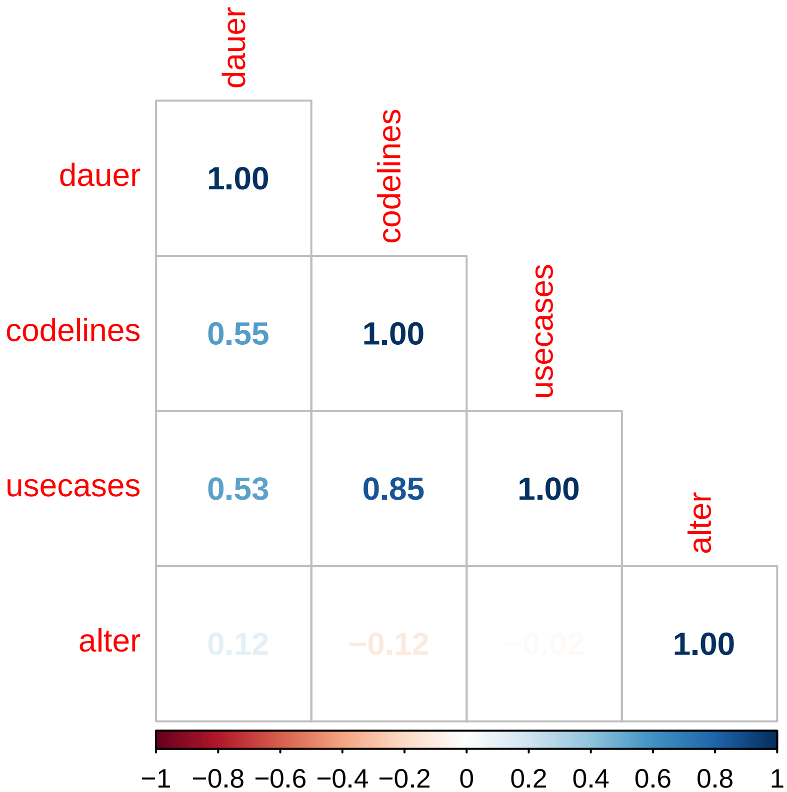
```
cor.daten <- cor(bugfixes)
corrplot(cor.daten, method = "ellipse")
```




```r
corrplot(cor.daten, method = "pie")
```




```
corrplot(cor.daten, method = "number", type = "lower")
```


dauer
codelines
usecases
alter
1.00
0.55
1.00
0.53
0.85
1.00
0.12
−0.12
−0.02
1.00
−1
−0.8
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
1

Metrische und kategoriale Variablen

Einfache ANOVA

In einer Softwareentwicklungsfirma wurden für 21 Bugs unter anderem zwei Merkmale erhoben:

- Behebungsdauer in Minuten (*dauer*)
- Programmierer (*programmierer*)

Es gilt zu untersuchen, ob es einen Unterschied zwischen den Programmieren gibt?

Die Daten liegen erneut als CSV-Datei `bugfixes3.csv` vor:

```
"dauer" "programmierer" "bugtyp"
120 "Eckkrammer" "GUI"
174 "Meyer" "DB"
188 "Meyer" "GUI"
...
```

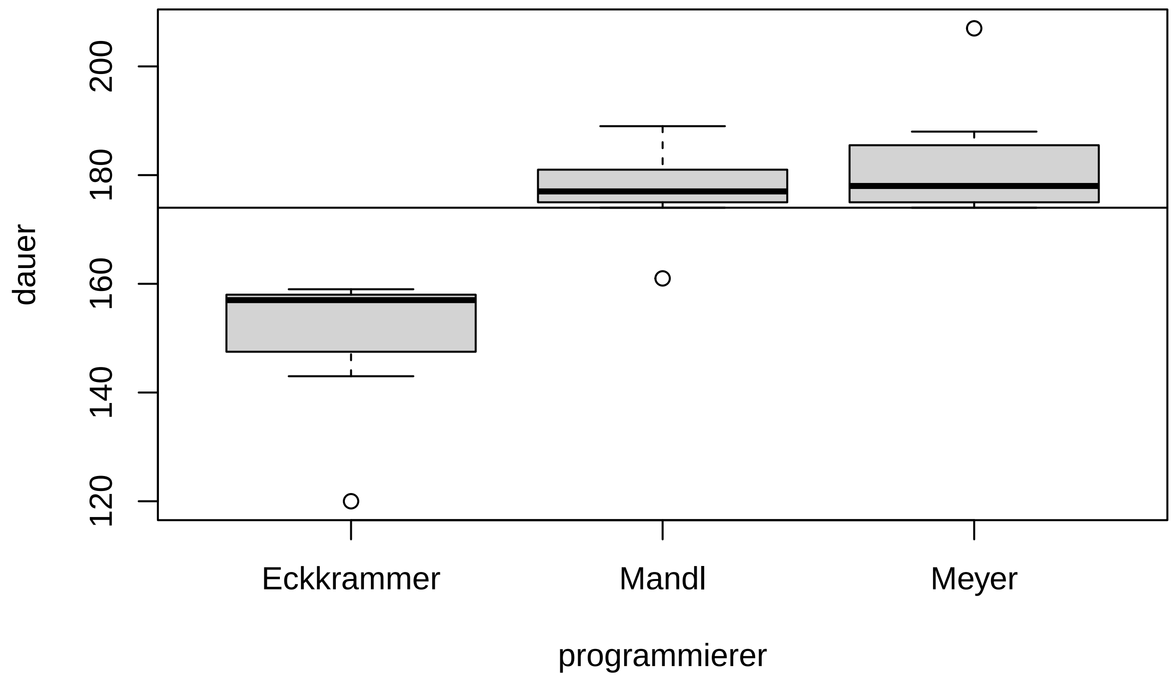
Einlesen mit `read.table()`:

```r
bugfixes = read.table("bugfixes3.csv", header = TRUE)
head(bugfixes)
```

```
##   dauer programmierer    bugtyp
## 1   120    Eckkrammer       GUI
## 2   174         Meyer        DB
## 3   188         Meyer       GUI
## 4   161         Mandl       GUI
## 5   157    Eckkrammer Reporting
## 6   178         Meyer Reporting
```

Die Verteilung der Bugfix-Dauer, gruppiert nach Programmierern wird mit parallellen Boxplots visualisiert:

```r
boxplot(dauer ~ programmierer, data = bugfixes)
abline(h = median(bugfixes$dauer))
```



Die 5-Punkt-Zusammenfassungen nach Programmierer lauten:

```
aggregate(dauer ~ programmierer, data = bugfixes, summary)
```

```
##   programmierer dauer.Min. dauer.1st Qu. dauer.Median dauer.Mean dauer.3rd Qu.
## 1    Eckkrammer   120.0000      147.5000     157.0000   149.5714      158.0000
## 2         Mandl   161.0000      175.0000     177.0000   177.0000      181.0000
## 3         Meyer   174.0000      175.0000     178.0000   182.8571      185.5000
##   dauer.Max.
## 1   159.0000
## 2   189.0000
## 3   207.0000
```

Programmierer Eckkrammer scheint deutlich schneller zu sein als seine beiden Kollegen.

Varianzanalyse

Um die Hypothese in der Grundgesamtheit zu testen, wird eine einfache Varianzanalyse verwendet. Zunächst muss die Voraussetzung der Varianzhomogenität mit dem Levene-Test überprüft werden:

```
library(car)
```

```
## Loading required package: carData
```

```r
leveneTest(dauer ~ programmierer, data = bugfixes)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  2  0.1425 0.8681
##       18
```

Dieser ist nicht signifikant, die ANOVA kann somit durchgeführt werden. Zunächst wird ein Regressionsmodell (mit *dauer* als abhängige und *programmierer* als unabhängige Variable) angepasst:

```r
model = lm(dauer ~ programmierer, data=bugfixes)
```

Die ANOVA-Tabelle lautet:

```r
anova(model)
```

```
## Analysis of Variance Table
##
## Response: dauer
##               Df Sum Sq Mean Sq F value    Pr(>F)
## programmierer  2 4420.7 2210.33   15.86 0.0001068 ***
## Residuals     18 2508.6  139.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der F-Test für den Faktor *programmierer* verwirft die Nullhypothese: "Alle Gruppenmitglieder sind gleich" auf dem 0.05-Niveau. Die drei Programmierer arbeiten also nicht gleich schnell.

Die Summary des Regressionsmodells lautet:

```r
summary(model)
```

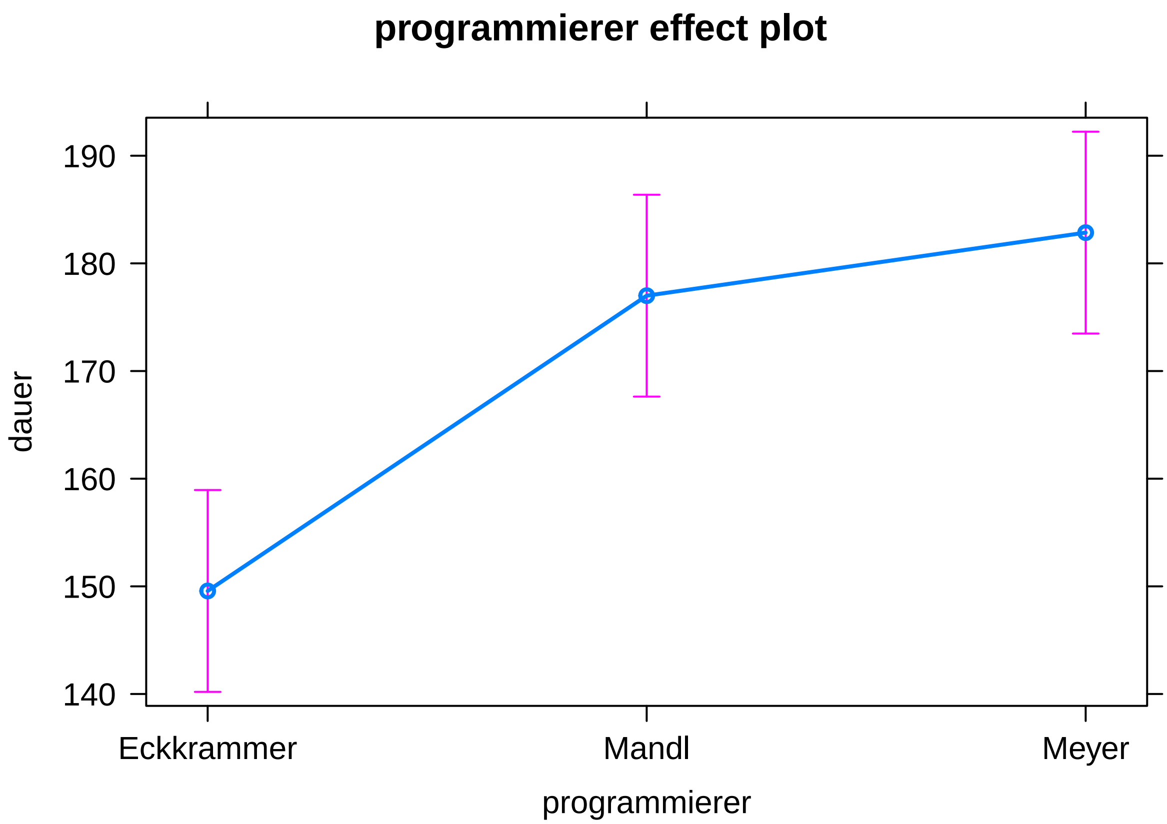
```
##
## Call:
## lm(formula = dauer ~ programmierer, data = bugfixes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.5714  -6.5714   0.1429   7.4286  24.1429
##
## Coefficients:
##                    Estimate Std. Error t value             Pr(>|t|)
## (Intercept)         149.571      4.462  33.521 < 0.0000000000000002 ***
## programmiererMandl   27.429      6.310   4.347             0.000389 ***
## programmiererMeyer   33.286      6.310   5.275            0.0000514 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.81 on 18 degrees of freedom
## Multiple R-squared:  0.638,  Adjusted R-squared:  0.5977
## F-statistic: 15.86 on 2 and 18 DF,  p-value: 0.0001068
```

Der Erklärungswert des Modells ist 63.8% und ist signifikant auf dem 0.05-Niveau. Der Achsenabschnitt repräsentiert die Bugfix-Dauer von Programmierer "Eckkramer" und ist signifikant, ebenso wie die Koeffizienten für die anderen beiden Programmierer. Diese sind signifikant langsamer als Programmierer Eckkramer. Der Effekt-Plot fasst dies grafisch zusammen:

```
library(effects)
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
plot(allEffects(model))
```



Auch hier sollte die Normalverteilung der Residuen mittels Q-Q-Plot visualisiert werden. Dies wird bewusst ausgelassen. Siehe *Zwei metrische Variablen (Korrelation)*.

Zweifache ANOVA

In einer Softwareentwicklungsfirma wurden für 21 Bugs (keine fehlenden Werte) folgende drei Merkmale erhoben:

- Behebungsdauer in Minuten (*dauer*)
- Programmierer (*programmierer*)
- Bugtyp (*bugtyp*)

Gibt es einen Einfluss der jeweiligen Fakoren? Gibt es eine Wechselwirkung?

Die Daten liegen in derselben CSV-Datei `bugfixes3.csv` vor (Siehe *Einfache ANOVA*).

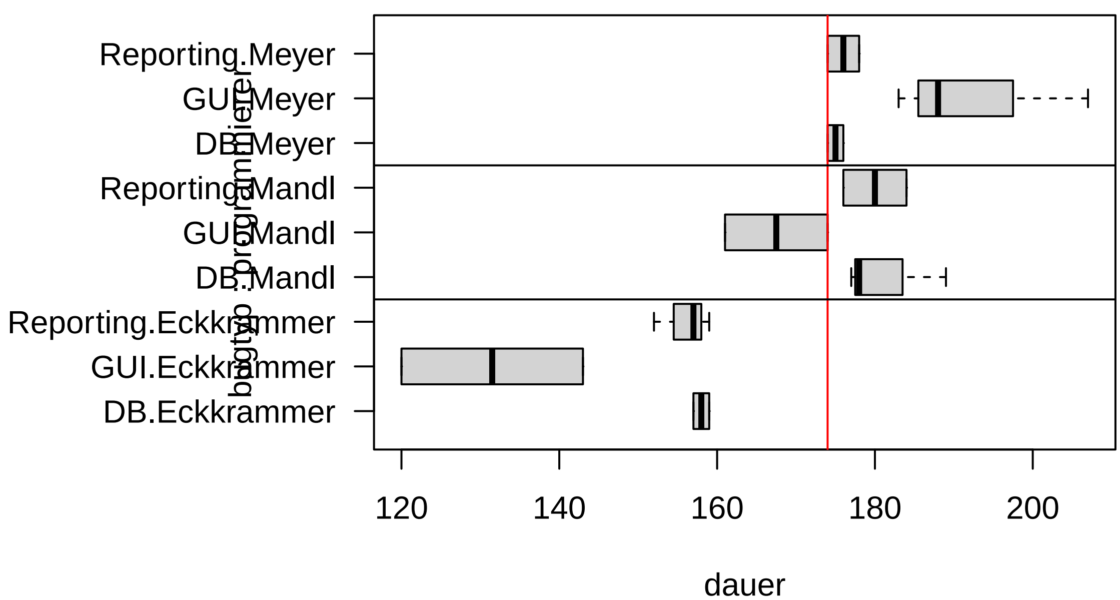
Einlesen mit `read.table()`:

```
bugfixes = read.table("bugfixes3.csv", header = TRUE)
head(bugfixes)
```

```
##   dauer programmierer    bugtyp
## 1   120     Eckkrammer       GUI
## 2   174          Meyer        DB
## 3   188          Meyer       GUI
## 4   161          Mandl       GUI
## 5   157     Eckkrammer Reporting
## 6   178          Meyer Reporting
```

Visualisieren der Verteilung der Bugfix-Dauer mit parallellen Boxplots:

```
par(oma = c(1,6,1,1))
boxplot(dauer ~ bugtyp + programmierer, bugfixes, las = 1, horizontal = TRUE)
abline(v = median(bugfixes$dauer), col = "red")
abline(h = c(3.5, 6.5))
```



Varianzanalyse

Um die Hypothesen: "Alle Programmierer sind gleich schnell" sowie: "Bugs verschiedenen Typs werden gleich schnell gefixt" in der Grundgesamtheit zu testen, wird eine zweifache Varianzanalyse ohne Wechselwirkungen verwendet.

Vorab wird erneut die Homogenität der Varianzen mit dem Levene-Test überprüft:

```
library(car)
leveneTest(dauer ~ programmierer * bugtyp, data = bugfixes)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  8  1.1735 0.3873
##       12
```

Dieser ist nicht signifikant, daher kann mit der ANOVA fortgefahren werden.

Dazu wird ein lineares Modell erstellt:

```
model = lm(dauer ~ programmierer + bugtyp, data = bugfixes)
```

Die ANOVA-Tablle lautet:

```
library(car)
Anova(model) # ACHTUNG: Anova(model) != anova(model)
```

```
## Anova Table (Type II tests)
##
## Response: dauer
##               Sum Sq Df F value    Pr(>F)
## programmierer 4467.1  2 15.1207 0.0002055 ***
## bugtyp         145.1  2  0.4912 0.6208037
## Residuals     2363.4 16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Faktor *programmierer* ist signifikant, *bugtyp* jedoch nicht.

Wie immer wird auch eine Zusammenfassung des Modells ausgegeben:

```
summary(model)
```

```
##
## Call:
## lm(formula = dauer ~ programmierer + bugtyp, data = bugfixes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -25.5804  -7.0804   0.9821   5.9821  27.3571
##
## Coefficients:
##                      Estimate Std. Error t value          Pr(>|t|)
```

```
## (Intercept)         151.393      6.149  24.621 0.000000000000038 ***
## programmiererMandl   27.375      6.564   4.171          0.000722 ***
## programmiererMeyer   34.062      6.564   5.189 0.000089495757407 ***
## bugtypGUI            -5.812      6.564  -0.886          0.388982
## bugtypReporting      -0.375      6.564  -0.057          0.955148
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.15 on 16 degrees of freedom
## Multiple R-squared:  0.6589, Adjusted R-squared:  0.5736
## F-statistic: 7.727 on 4 and 16 DF,  p-value: 0.001149
```

Es hat eien Erklärungswert von 65.9% und ist signifikant auf dem 0.05-Niveau.

Prüfen wir nun die Frage nach einer möglichen *Wechselwirkung* zwischen *programmierer* und *bugtyp*. Hierzu wird ein Regressionsmodell mit zusätzlichem Interaktionsterm geschätzt:

```
model = lm(dauer ~ programmierer * bugtyp, data = bugfixes)
```

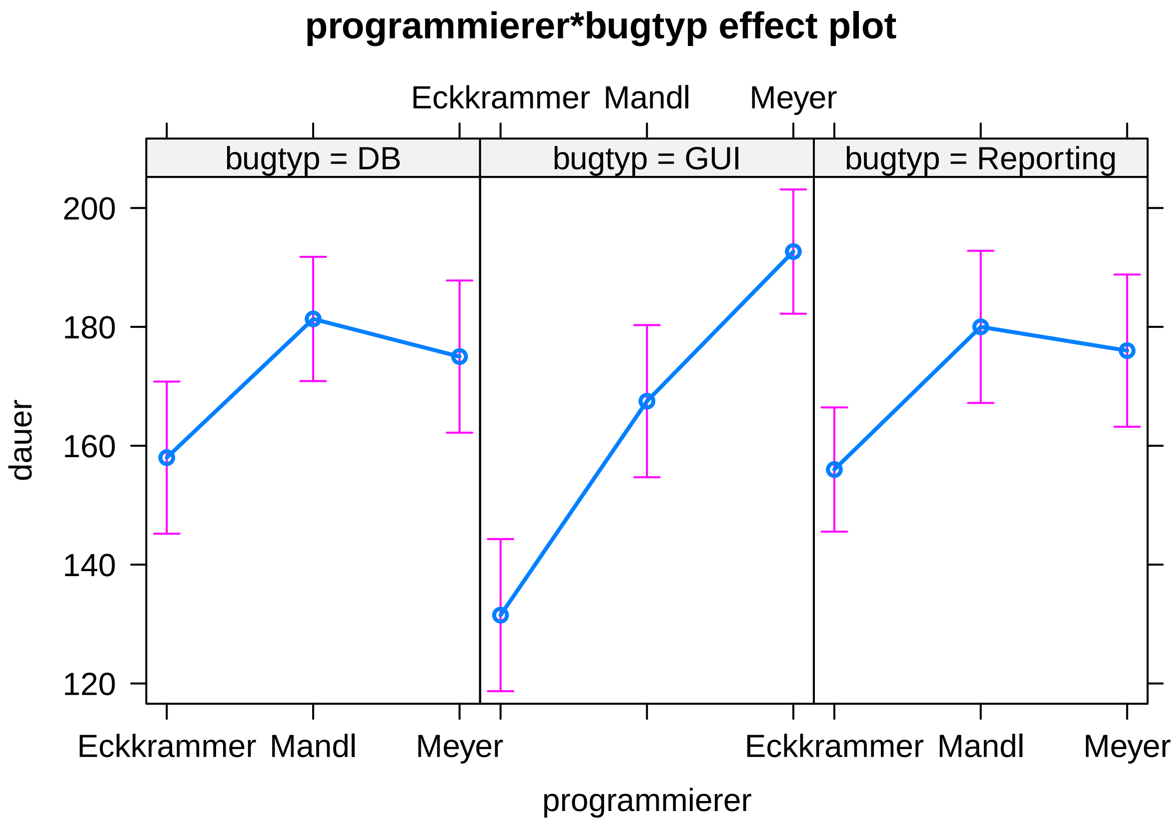
Die ANOVA-Tabelle lautet:

```
Anova(model)
```

```
## Anova Table (Type II tests)
##
## Response: dauer
##                      Sum Sq Df F value     Pr(>F)
## programmierer        4467.1  2 32.3574 0.00001465 ***
## bugtyp                145.1  2  1.0512   0.379600
## programmierer:bugtyp 1535.1  4  5.5598   0.009076 **
## Residuals             828.3 12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Signifikanzen der Haupteffekte ändern sich nicht (Programmierer signifikant, Bugtyp nicht), der Interaktionsterm ist signifikant. Es besteht also ein gemeinsamer Einfluss von Programmierer und Bugtyp auf die Bugfix-Dauer. Die Auswirkungen der jeweiligen Faktorkombinationen sind im Effekt-Plot grafisch dargestellt:

```
library(effects)
plot(allEffects(model))
```



Es fällt zunächst auf, dass für alle drei Bugtypen Programmierer Eckkrammer signifikant schneller als die Kollegen arbeitet (in Einklang mit dem signifikanten Haupteffekt). Bei GUI-Bugs unterscheiden sich alle drei Kollegen signifkant voneinander: hier sind die Unterschiede somit besonders stark ausgeprägt (Wechselwirkungseffekt).

Zeitreihenanalyse

Langfristprognose

Für einen IT-Händler sind die monatlichen Absatzzahlen von PCs für die Jahre 2015–2017 erhoben worden. Es ist ein Vorhersagemodell für Langfrist-Prognosen zu erstellen und die Entwicklung im Jahr 2018 vorherzusagen.

Die Daten liegen als Text vor, und werden daher mit `scan()` eingelesen und in eine Zeitreihe umgewandelt:

```
x = scan(text = "779  703  772  781  768  765 599 403  712  707  770  777
                1014  929 1008 1025 1010  996 738 448 1053 1042 1010 1022
                1048  973 1042 1059 1044 1030 572 382 1087 1076 1044 1056")
ist = ts(x, start = c(2015,1), freq = 12)
ist
```

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2015  779  703  772  781  768  765  599  403  712  707  770  777
## 2016 1014  929 1008 1025 1010  996  738  448 1053 1042 1010 1022
## 2017 1048  973 1042 1059 1044 1030  572  382 1087 1076 1044 1056
```

Stichprobengröße, fehlende Werte:

```
length(x)
```

```
## [1] 36
```

```
sum(is.na(x))
```

```
## [1] 0
```

Numerische Beschreibung:

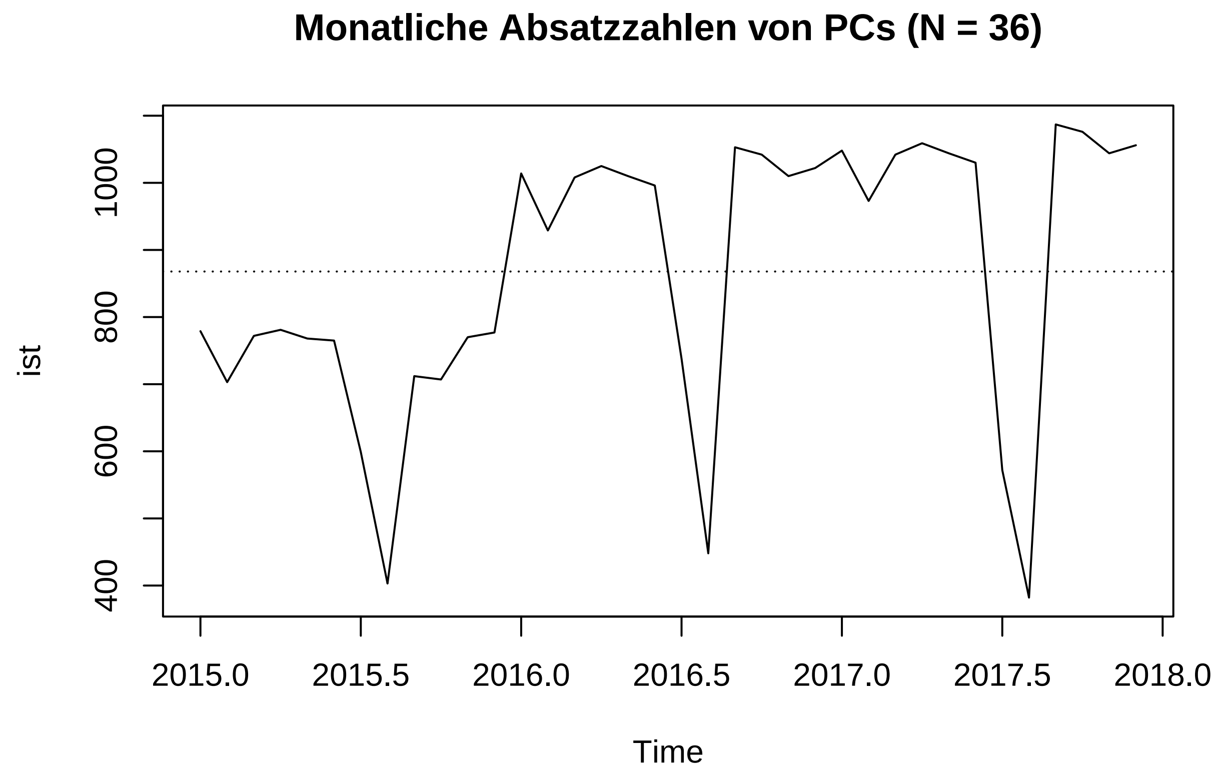
```
summary(ist)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   382.0   758.2   984.5   867.9  1042.0  1087.0
```

Der mittlere Absatz über den Beobachtungszeitraum beträgt rd. 868 Stück; das Minimum lag bei 382, das Maximum bei 1087 Stück.

Das Zeitreihendiagramm sieht wie folgt aus:

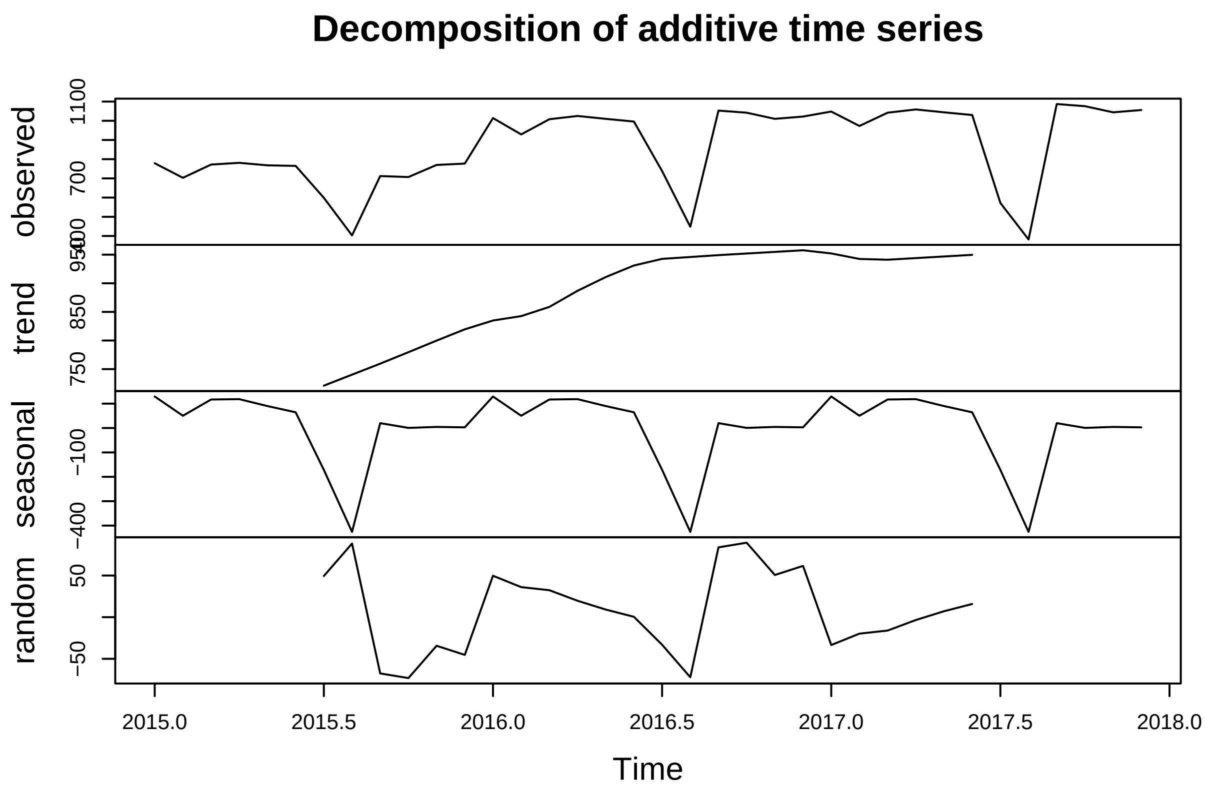
```
plot(ist, main = "Monatliche Absatzzahlen von PCs (N = 36)")
abline(h = mean(ist), lty = "dotted")
```



Die gestrichelte Gerade stellt den durchschnittlichen Absatz in den Jahren dar. Betrachtet man nun den Verlauf der Zeitreihe, fallen die periodisch auftretenden, starken Absatzeinbrüche jeweils zu Jahresmitte auf. Dies ist vermutlich auf das typische *Sommerloch* im Handel zurückzuführen. Ein kleinerer Einbruch könnte im Jänner zu beobachten sein (möglicherweise in Folge der vorgezogenen Käufe zu Weihnachten?). Des weiteren dürfte es generell einen Aufwärtstrend in den Absatzzahlen geben, mit einem stärkeren Sprung von 2015 auf 2016.

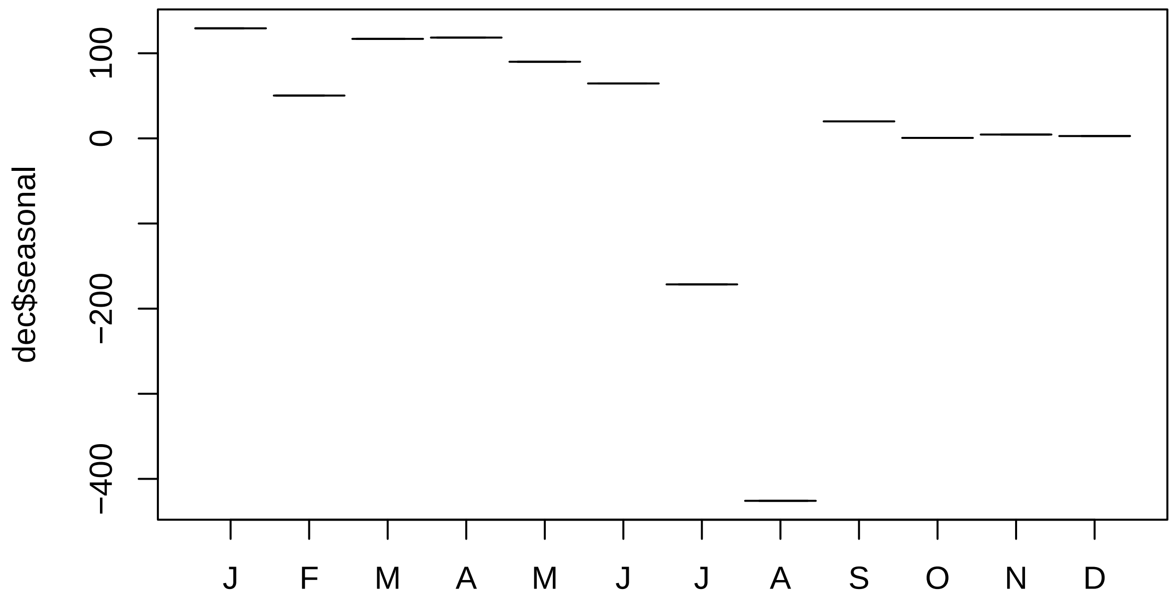
Für eine nähere Betrachtung wird die Zeitreihe in Trend- und Saisonkomponente zerlegt und diese dargestellt:

```
dec = decompose(ist)
plot(dec)
```



Der vermutete Aufwärtstrend (mit Sprung vom 1. zum 2. Jahr) ist gut in der Trendkomponente zu erkennen, ebenso wie der Saisoneffekt. Betrachten wir die Saisonfigur genauer:

```
monthplot(dec$seasonal)
```



Prognosemodell

Für die Vorhersage wird ein Regressionsmodell geschätzt:

```
t = time(ist)
s = as.factor(cycle(ist))
model = lm(ist ~ t + s)
```

Die Modellzusammenfassung lautet:

```
summary(model)
```

```
##
## Call:
## lm(formula = ist ~ t + s)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -184.21  -48.29  -15.71   67.67  111.88
##
## Coefficients:
##                  Estimate Std. Error t value     Pr(>|t|)
## (Intercept) -240721.00   37835.00  -6.362 0.0000017147 ***
## t                119.87      18.77   6.387 0.0000016166 ***
```

```
## s2              -88.66      75.09  -1.181       0.250
## s3              -26.31      75.13  -0.350       0.729
## s4              -21.97      75.22  -0.292       0.773
## s5              -46.29      75.33  -0.615       0.545
## s6              -66.61      75.48  -0.883       0.387
## s7             -370.60      75.65  -4.899 0.0000599735 ***
## s8             -605.93      75.86  -7.987 0.0000000441 ***
## s9              -76.25      76.10  -1.002       0.327
## s10             -95.24      76.38  -1.247       0.225
## s11            -105.56      76.68  -1.377       0.182
## s12            -105.22      77.02  -1.366       0.185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 91.94 on 23 degrees of freedom
## Multiple R-squared:  0.8691, Adjusted R-squared:  0.8008
## F-statistic: 12.73 on 12 and 23 DF,  p-value: 0.0000001919
```

Das Modell hat einen sehr hohen Erkärungswert von fast 87%. Achsenabschnitt und Trend sind signifikant, wie auch die Saisoneffekte im Juli und August.

Die Regressionsgleichung für den erwarteten Absatz, gegeben Zeit und Saison, lautet:

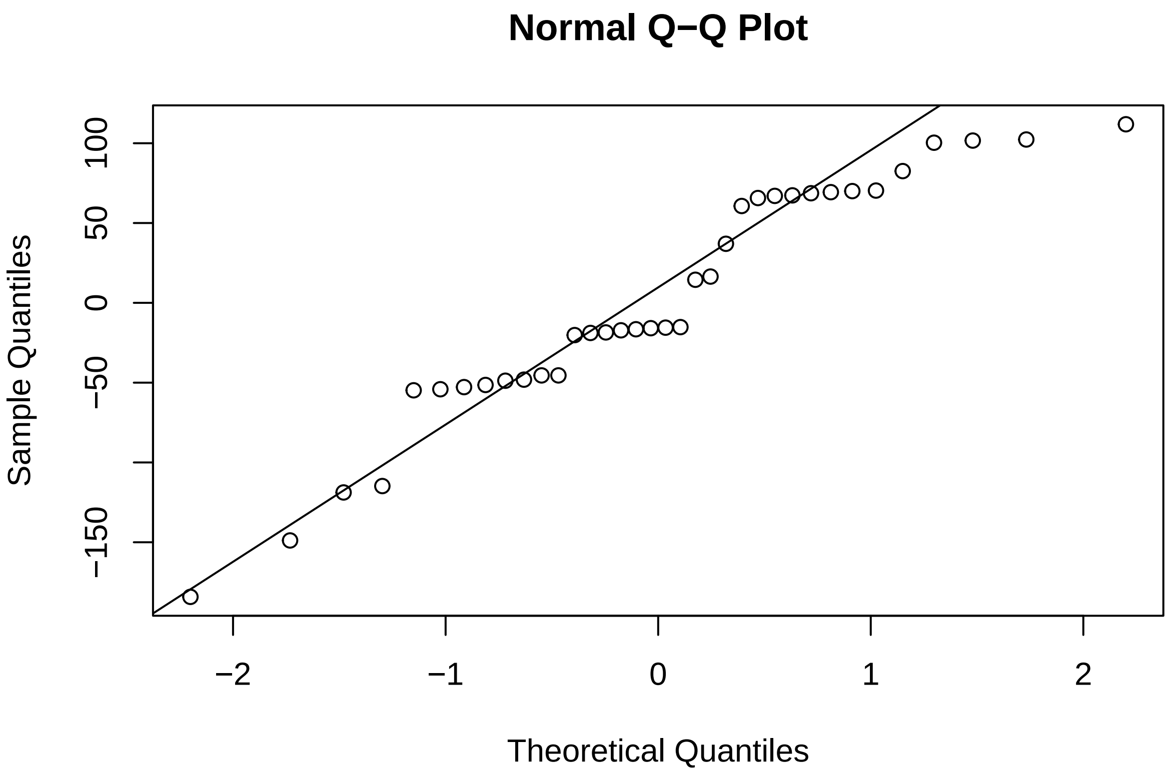
```
E(Ist|t,s) = -240721 + 119.87 t -88.66 (s == "Februar") -26.31 (s == "März") ...
```

(Zahlencodierung der Monate zwecks Klarheit durch Text ersetzt.)

Diagnostik

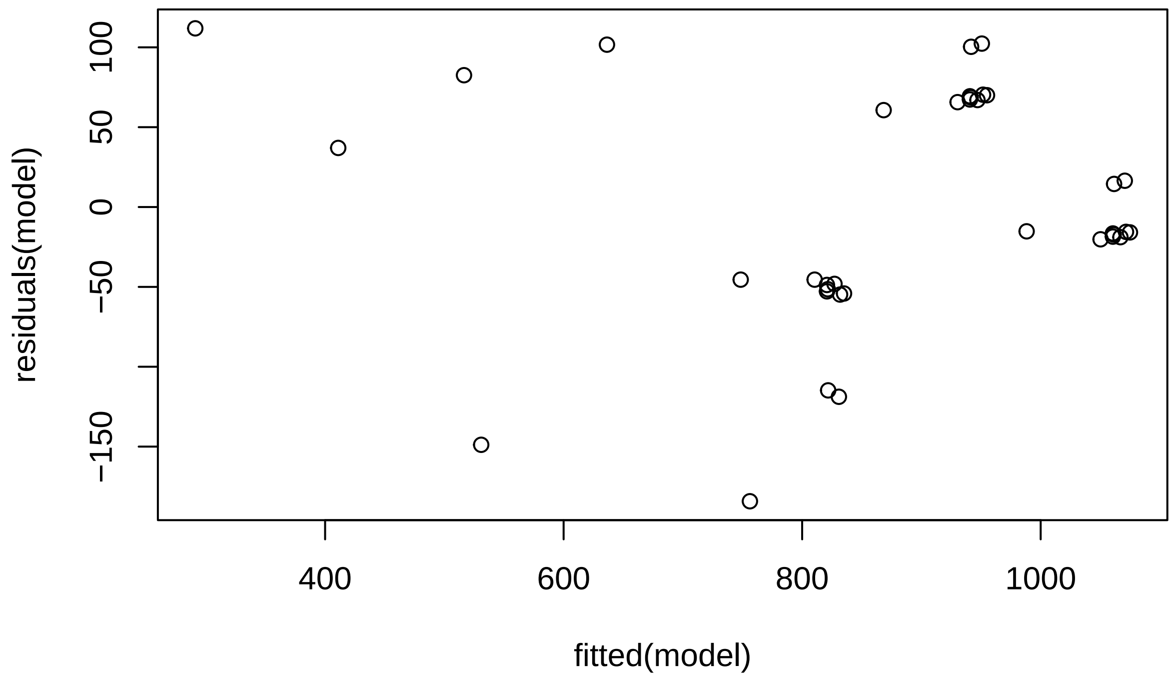
Vor einer Prognose ist die Gültigkeit der Modellannahmen zu überprüfen. Aufgrund des geringen Stichprobenumfangs ist zunächst jedenfalls die Normalverteilung der Residuen zu kontrollieren, dies erfolgt mit einem Q-Q-Plot gegen die theoretischen Werte der Normalverteilung:

```
qqnorm(residuals(model))
qqline(residuals(model))
```



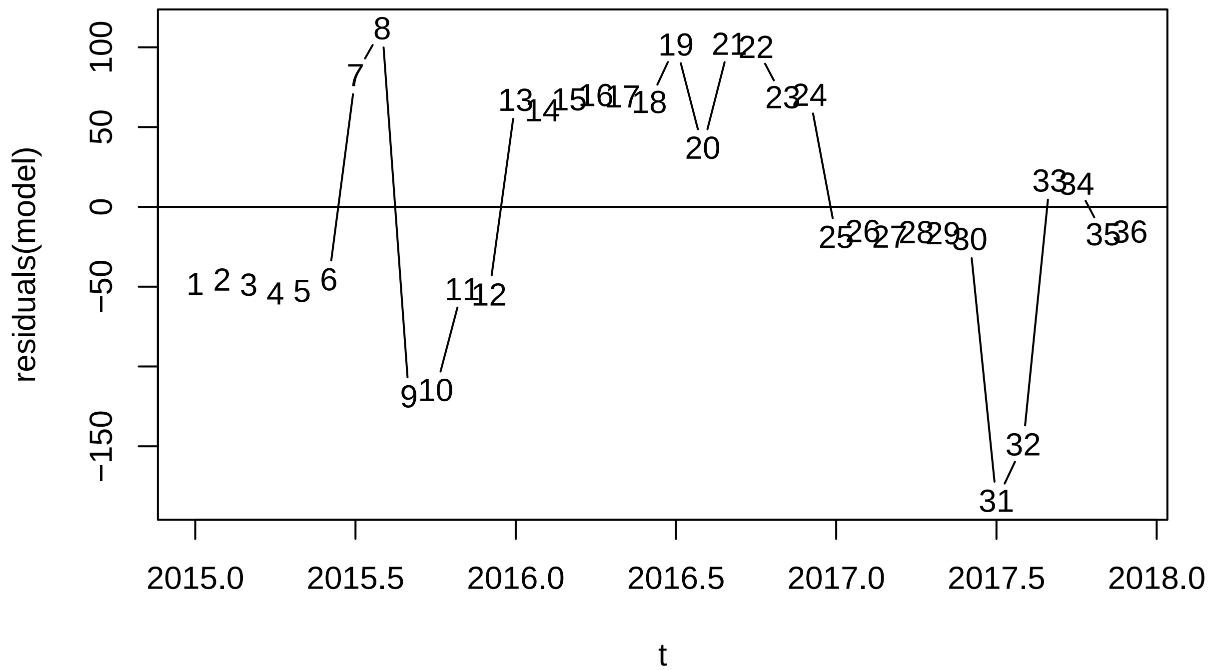
Es ist eine starke Korrelation der Residuen mit den theoretischen normalverteilten Werten zu erkennen. Ein paar Punkte oben rechts weichen jedoch vom Schema ab:

```
plot(residuals(model) ~ fitted(model))
```



Des weiteren ist die korrekte Modellspezifikation durch ein Streudiagramm der Residuen gegen die geschätzten Werte zu überprüfen. Bis auf ein paar Häufungspunkte sind keine auffälligen Strukturen im Streudiagramm zu erkennen, das lineare Modell dürfte die in den Daten vorhandene Information gut abdecken.

```
plot(residuals(model) ~ t)
abline(h = 0)
```



Drittens ist die Varianzhomogenität der Residuen zu überprüfen. Dies erfolgt mit einem Diagramm der Residuen gegen die Zeit (siehe Abbildung **??**). Die Residuen weisen auffällig hohe Veränderungen in den Sommemonaten 2015 und 2017 auf, die Abweichungen des Modells von den historischen Daten sind hier größer als im übrigen Zeitraum. Das Modell fängt den Saisoneffekt offenbar nicht ideal ein. Im übrigen scheint sich die Streuung nicht systematisch mit der Zeit zu verändern.

Prognose

Für die Prognose des Folgejahres werden zunächst die Zukunftswerte der unabhängigen Variablen generiert:

```
newdata = data.frame(t = seq(from = 2018, by = 1/12, length = 12),
                     s = as.factor(1:12))
newdata
```

```
##           t  s
## 1  2018.000  1
## 2  2018.083  2
## 3  2018.167  3
## 4  2018.250  4
## 5  2018.333  5
## 6  2018.417  6
## 7  2018.500  7
## 8  2018.583  8
## 9  2018.667  9
## 10 2018.750 10
```

```
## 11 2018.833 11
## 12 2018.917 12
```

und für diese die Prognose erstellt:

```
tmp = predict(model, newdata)
pred = ts(tmp, start = c(2018, 1), frequency = 12)
pred
```

```
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2018 1186.7500 1108.0833 1180.4167 1194.7500 1180.4167 1170.0833  876.0833
##            Aug       Sep       Oct       Nov       Dec
## 2018  650.7500 1190.4167 1181.4167 1181.0833 1191.4167
```

In der grafischen Darstellung wird zusätzlich der Prognosekanal eingezeichnet:

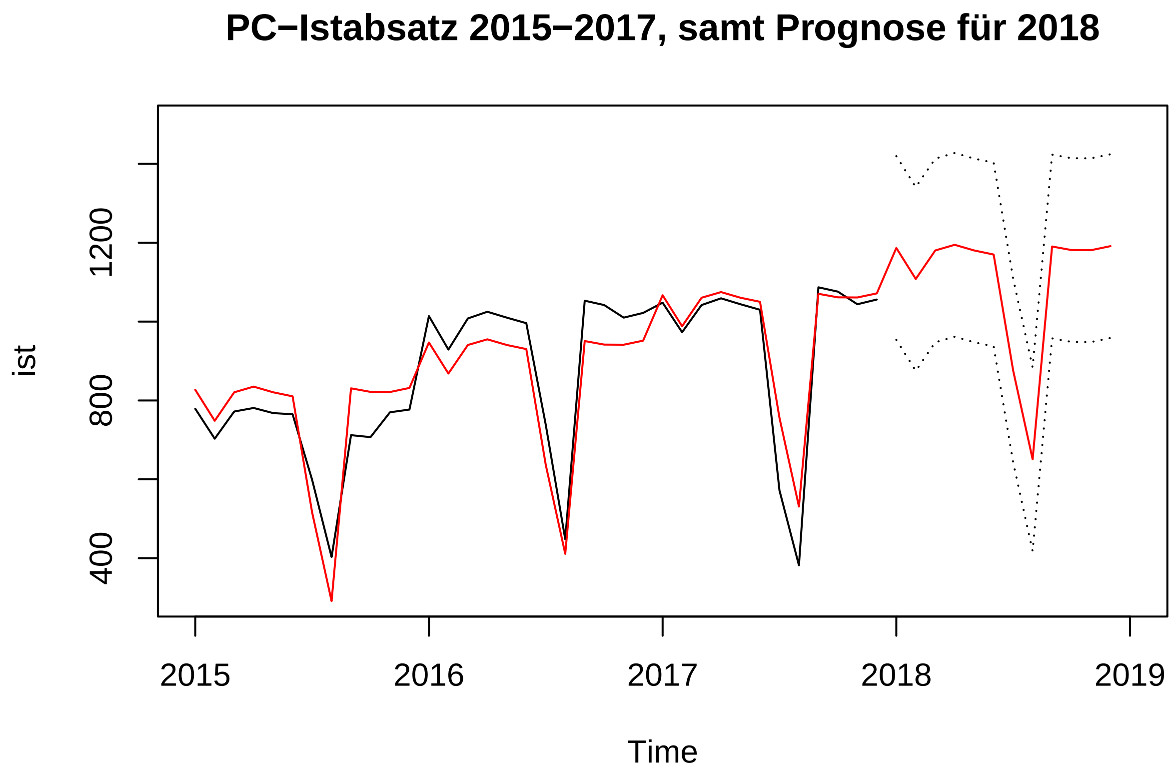
```
## Ist-Werte
plot(ist,
     main = "PC-Istabsatz 2015-2017, samt Prognose für 2018",
     xlim = c(2015, 2019), ylim = c(300, 1500))

## Erwartete Werte
fit = ts(c(fitted(model), pred), start = start(ist), freq = frequency(ist))
lines(fit, col = "red")

## Prognoseintervalle
conf = predict(model, newdata, interval = "prediction")

## Unteres Band
lwr = ts(conf[,2], start = 2018, freq = 12)
lines(lwr, col = "black", lty = "dotted")

## Oberes Band
upr = ts(conf[,3], start = 2018, freq = 12)
lines(upr, col = "black", lty = "dotted")
```



Die geschätzten Werte (rot) folgen in etwa dem Verlauf der Daten, mit deutlichen Abweichungen besonders in den ersten 2 Jahren. Durch den Sprung von Jahr 2015 zum Jahr 2016 ist die Schätzung des Trends verzerrt. Das Prognoseband zeigt eine große Schwankungsbreite und entsprechende Ungenauigkeit der Vorhersagen.

Variante:

Für die Prognose von Juli 2018 bis Juni 2019 werden zunächst die Zukunftswerte der unabhängigen Variablen generiert:

```
newdata = data.frame(t = seq(from = 2018.5, by = 1/12, length = 12),
                     s = as.factor(c(7:12, 1:6)))
newdata
```

```
##           t  s
## 1  2018.500  7
## 2  2018.583  8
## 3  2018.667  9
## 4  2018.750 10
## 5  2018.833 11
## 6  2018.917 12
## 7  2019.000  1
## 8  2019.083  2
## 9  2019.167  3
## 10 2019.250  4
## 11 2019.333  5
## 12 2019.417  6
```

und für diese die Prognose erstellt:

```r
tmp = predict(model, newdata, interval = "predict")
pred = ts(tmp, start = c(2018, 7), frequency = 12)
pred
```

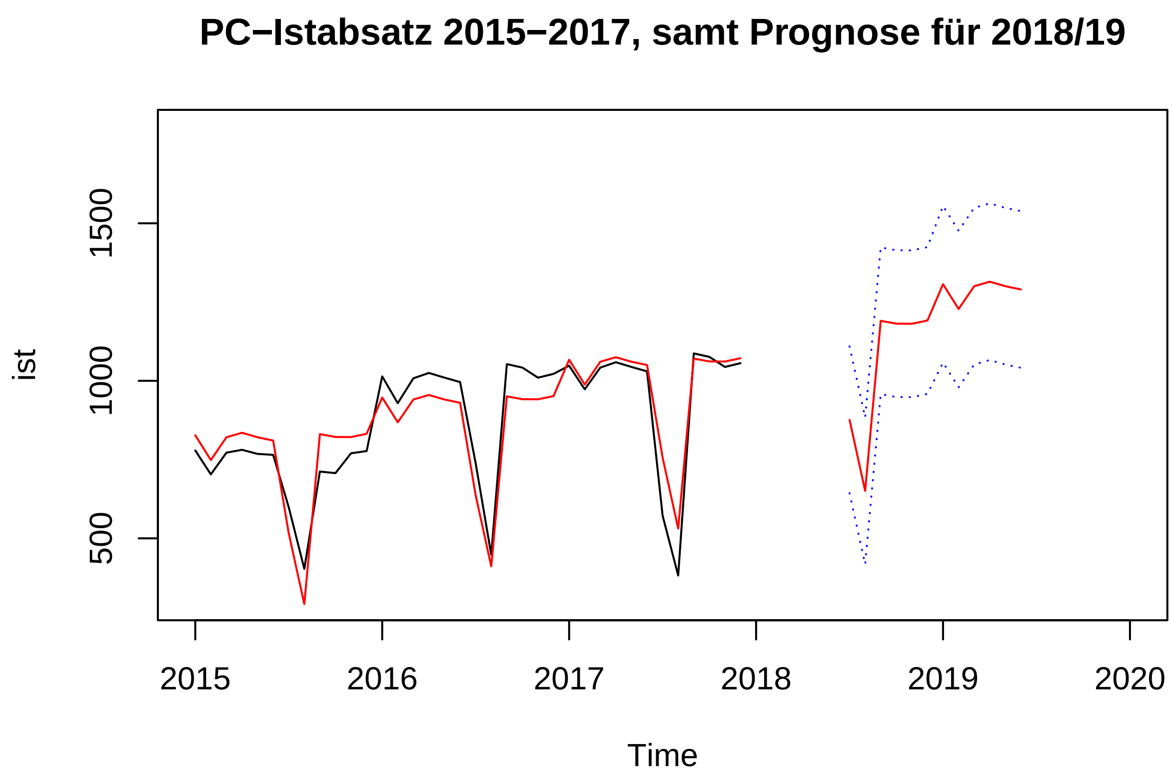
```
##                fit       lwr       upr
## Jul 2018  876.0833  643.1441 1109.0226
## Aug 2018  650.7500  417.8108  883.6892
## Sep 2018 1190.4167  957.4774 1423.3559
## Oct 2018 1181.4167  948.4774 1414.3559
## Nov 2018 1181.0833  948.1441 1414.0226
## Dec 2018 1191.4167  958.4774 1424.3559
## Jan 2019 1306.6250 1058.0352 1555.2148
## Feb 2019 1227.9583  979.3685 1476.5481
## Mar 2019 1300.2917 1051.7019 1548.8815
## Apr 2019 1314.6250 1066.0352 1563.2148
## May 2019 1300.2917 1051.7019 1548.8815
## Jun 2019 1289.9583 1041.3685 1538.5481
```

In der grafischen Darstellung wird zusätzlich der Prognosekanal eingezeichnet:

```r
## Ist-Werte
plot(ist,
     main = "PC-Istabsatz 2015-2017, samt Prognose für 2018/19",
     xlim = c(2015, 2020), ylim = c(300, 1800))

## Erwartete Werte
fit = ts(fitted(model), start = start(ist), freq = frequency(ist))
lines(fit, col = "red")

## Prognose
lines(pred[,"fit"], col = "red")
lines(pred[,"lwr"], col = "blue", lty = "dotted")
lines(pred[,"upr"], col = "blue", lty = "dotted")
```



Kurzfristprognose

Für einen PC-Assemblierer soll der tägliche Bedarf an Schrauben, die in der Montageabteilung benötigt werden, für die Beschaffungsplanung vorhergesagt werden. Es stehen die Werte der letzten vier Wochen zur Vefügung (Freitags wird nur bis Mittag gearbeitet).

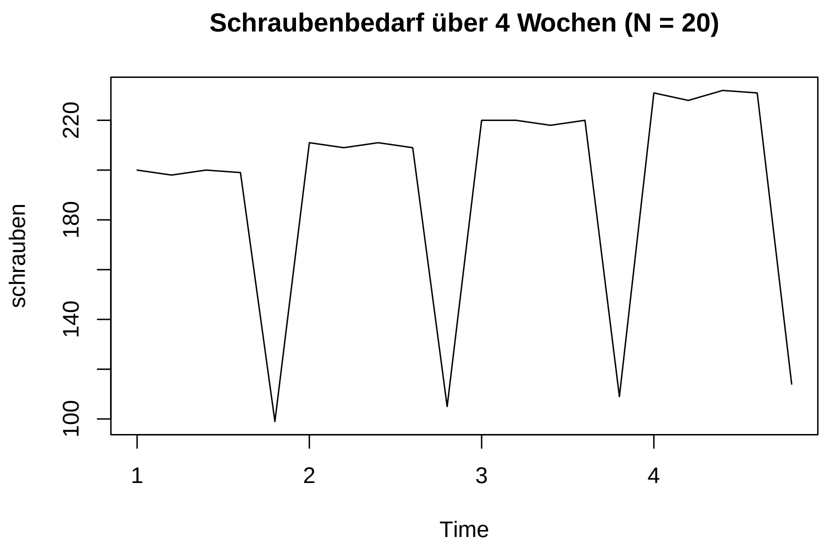
Die Daten wurden mit `scan()` eingelesen und in eine Zeitreihe umgewandelt:

```
x = scan(text = "200 198 200 199  99
                 211 209 211 209 105
                 220 220 218 220 109
                 231 228 232 231 114")

schrauben = ts(x, frequency = 5)
```

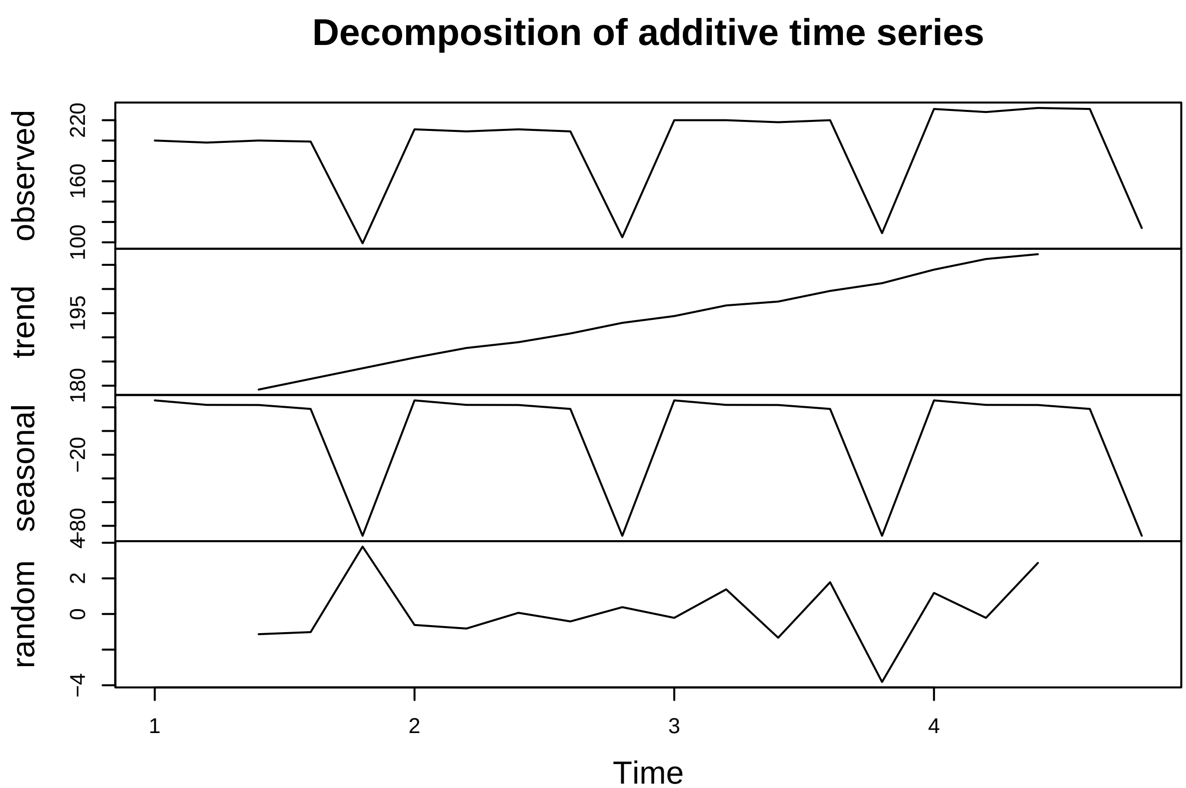
Das Zeitreihendiagramm sieht wie folgt aus:

```
plot(schrauben, main = "Schraubenbedarf über 4 Wochen (N = 20)")
```



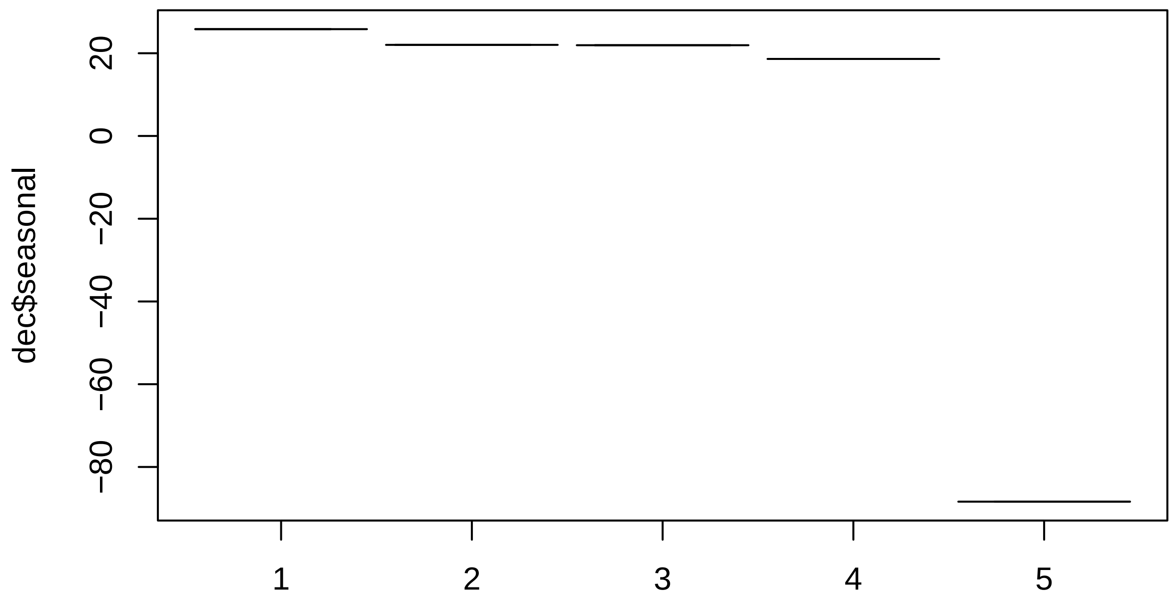
Für eine nähere Betrachtung wird die Zeitreihe in Trend- und Saisonkomponente zerlegt und diese dargestellt:

```
dec = decompose(schrauben)
plot(dec)
```



Der vermutete Aufwärtstrend ist gut in der Trendkomponente zu erkennen, die nahezu konstant ansteigt, ebenso wie der Saisoneffekt. Betrachten wir die Saisonfigur genauer:

```
monthplot(dec$seasonal)
```



Im wesentlichen erfolgt am Freitag eine deutliche Reduktion (minus 80 Stück) des Schraubenbedarfs gegenüber dem Basisniveau – bedingt durch die geringere Arbeitsleistung aufgrund des früheren Arbeitsschlusses.

Modellschätzung

Für die Kurzfrist-Prognose der kommenden Tage wird ein *HoltWinters-Modell* mit Saison- und Trendkomponente verwendet:

```
model = HoltWinters(schrauben)
model
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = schrauben)
##
## Smoothing parameters:
##  alpha: 0.2503222
##  beta : 0
```

```
##  gamma: 1
##
## Coefficients:
##          [,1]
## a  209.367738
## b    2.011429
## s1  29.068301
## s2  24.367791
## s3  25.246393
## s4  21.985071
## s5 -95.367738
```

Bei der automatischen Anpassung der Glättungsparameter wurde für α (Parameter für Achsenabschnitt) 25% gewählt (Berücksichtigung der aktuellen Werte nur zu 25%); für β (Parameter für Trend) der Wert 0 (nur die vergangenen Werte waren für die Schätzung relevant); für γ (Parameter für Saison) der Wert 1 (nur die aktuellsten Werte waren jeweils für die Schätzung ausschlaggebend).

Das Prognosemodell lautet:

$$Y_{t+k} = 209.36 + 2.011k + 29.07(s = 1) + 24.37(s = 2) + 25.25(s = 3) + 21.98(s = 4) - 95.36(s = 5)$$

wobei k den k-ten Messpunkt (also Tag) in der Zukunft darstellt. Der Basisbedarf beträgt somit rd. 209 Stück, erhöht sich pro Tag um 2 Stück, und wird je nach Wochentag um 22 bis 29 Stück nach oben, aber am Freitag um rd. 95 Stück nach unten korrigiert.

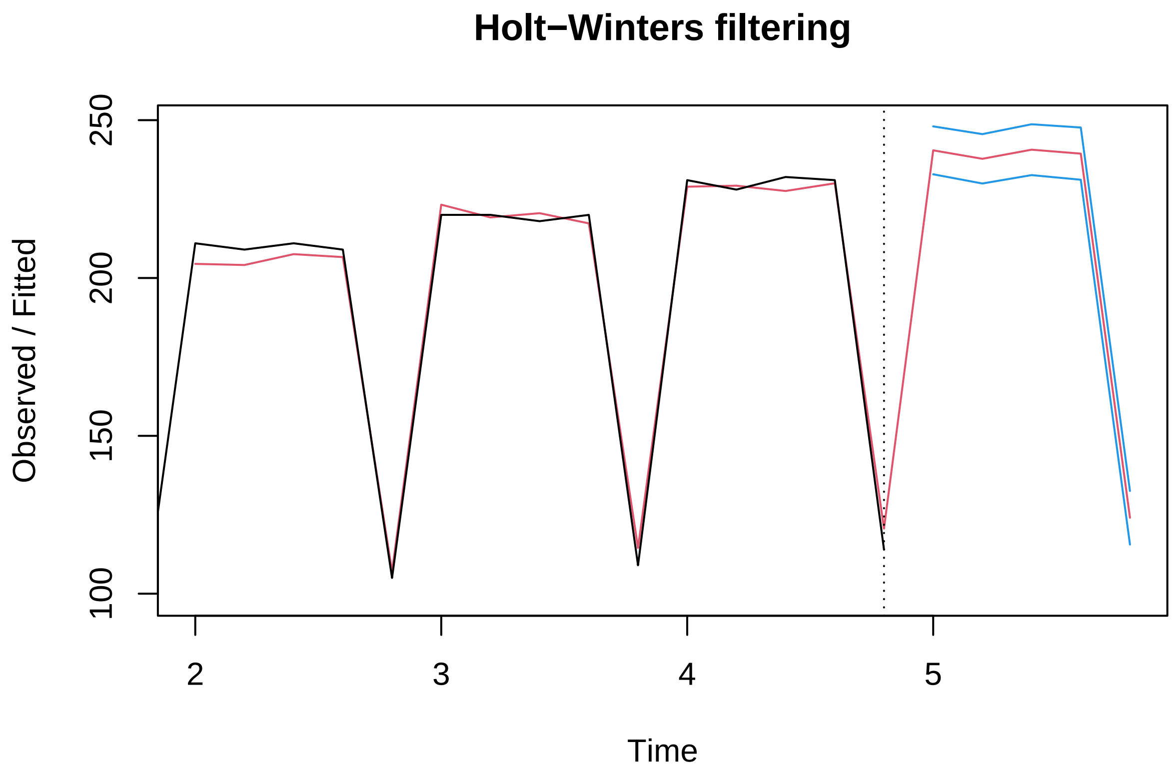
Die Prognose für die nächsten 5 Tage (samt Prognoseintervalle) lautet:

```
pred = predict(model, n.ahead = 5, prediction.interval = TRUE)
pred
```

```
## Time Series:
## Start = c(5, 1)
## End = c(5, 5)
## Frequency = 5
##          fit      upr      lwr
## 5.0 240.4475 248.0368 232.8581
## 5.2 237.7584 245.5819 229.9348
## 5.4 240.6484 248.6993 232.5975
## 5.6 239.3985 247.6705 231.1265
## 5.8 124.0571 132.5445 115.5698
```

und ist samt den Ursprungsdaten in folgendem Diagramm dargestellt:

```
plot(model, predicted.values = pred)
```



Holt−Winters filtering
Observed / Fitted
250
200
150
100
2
3
4
5
Time

Logistische Regression

In einer Firma wurden in einem Serverraum Lüfter verschiedenen Alters (in Monaten) auf ihre Funktionsfähigkeit überprüft:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Alter	2	10	12	16	17	18	19	19	20	21	25	27	29	30	34	36	36
Ausfall	F	F	F	F	F	T	F	F	T	T	F	T	T	T	T	T	T

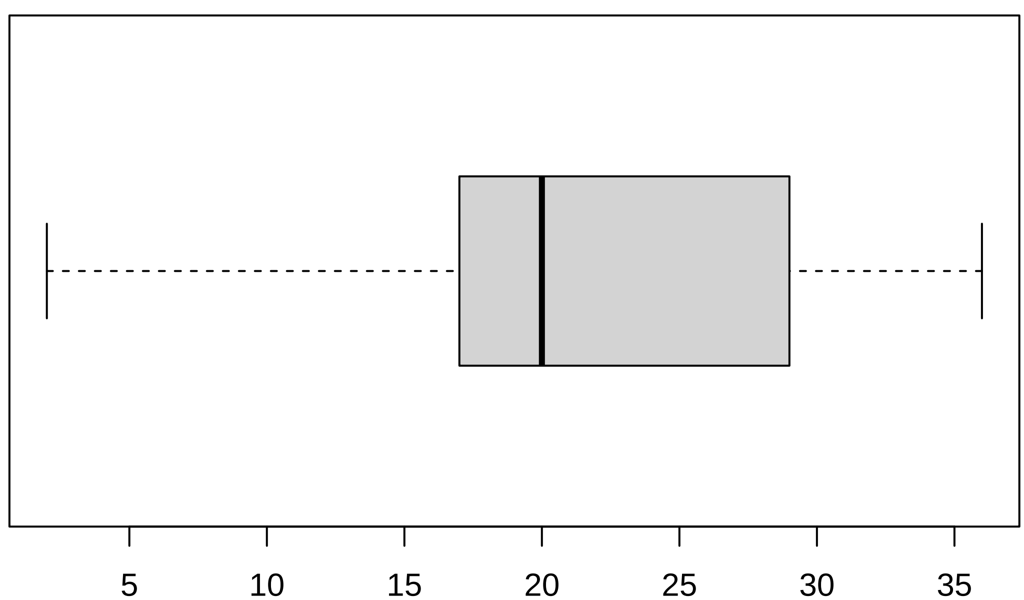
Kann man die Ausfallswahrscheinlichkeit durch das Alter vorhersagen? Wie lautet diese für 25, 30 und 35 Monate?

Einlesen der Daten mittels `scan()`. Die binäre Variable `defekt` wird als Faktor mit 2 Ausprägungen (`Ja`, `Nein`) kodiert:

```
alter = scan(text = "2 10 12 16 17 18 19 19 20 21 25 27 29 30 34 36 36")
defekt = scan(text = "F  F  F  F  F  T  F  F  T  T  F  T  T  T  T  T  T", what = TRUE)
ausfall  = factor(defekt, labels = c("Nein","Ja"))
luefter = data.frame(Alter = alter, Ausfall = ausfall)
```

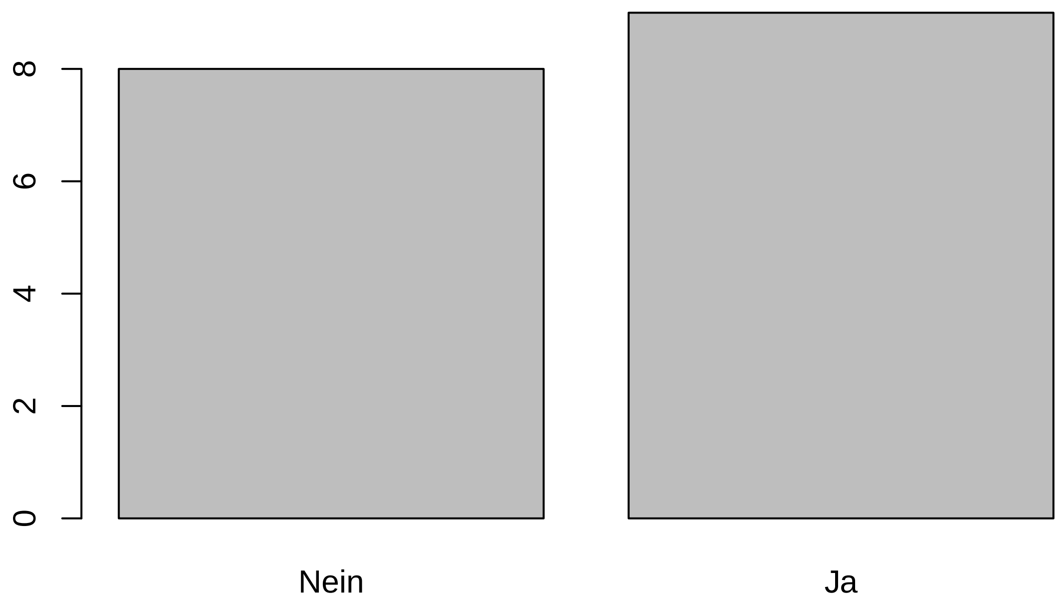
Die Verteilung von *Alter* wird mit einem Boxplot dargestellt. Die Verteilung ist weder symmetrisch noch eindeutig links- oder rechtsschief, Ausreißer sind nicht zu erkennen. Die Daten befinden sich im Bereich $[2, 36]$ Monate, das arithmetische Mittel beträgt 21.8, der Median 20 Monate. Der Mittelwert wird also durch die Schiefe der Verteilung etwas nach oben verzerrt:

```
boxplot(luefter$Alter, horizontal = TRUE)
```



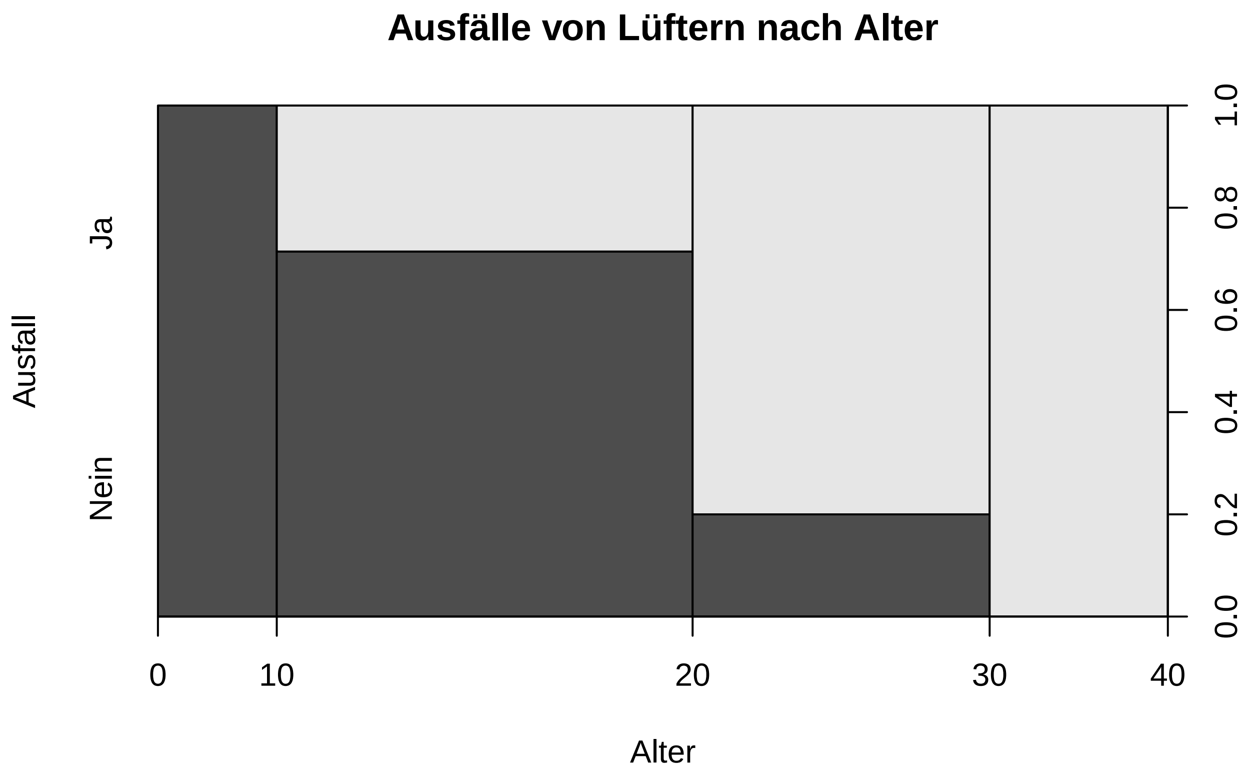
Der Boxplot von *Ausfall* wird mit einem Balkendiagramm (siehe Abbildung **??**) dargestellt - 9 von 17 Lüftern sind insgesamt ausgefallen:

```
barplot(table(luefter$Ausfall))
```



Zur Visualisierung der Anteile in Abhängigkeit vom Alter wird ein "Spinogram" verwendet:

```
spineplot(Ausfall ~ Alter, data = luefter,
          breaks = 3, ylevels = c("Ja","Nein"),
          main = "Ausfälle von Lüftern nach Alter")
```



- Die Balkenbreite ist proportional zur Dichte der Daten im Bereich: die meisten Lüfter befinden sich in der Gruppe von 10-20 Monaten.
- Die schwarzen Balken "highlighten" den Anteil der binären Variablen innerhalb jedes Bereichs: der Anteil der ausgefallenen Lüfter steigt an: Keiner der Lüfter, die weniger als 10 Monate alt sind, ist zum Zeitpunkt der Untersuchung ausgefallen. Sind die Lüfter zwischen 10 und 20 Monate alt, beträgt die Ausfallswahrscheinlichkeit ca. 30%, bei einer Betriebsdauer zwischen 20 und 30 Monaten sind bereits vier von fünf Geräten ausgefallen. Von den Geräten, die älter als 30 Monate sind, funktioniert in der Stichprobe keines mehr.

Modell

Da ein binäres Merkmal durch ein metrisches erklärt werden soll, wird ein logistisches Regressionsmodell geschätzt:

```
model = glm(Ausfall ~ Alter, data = luefter, family = "binomial")
pander(model, caption = "Parameterschätzung des Modells: Ausfall ~ Alter")
```

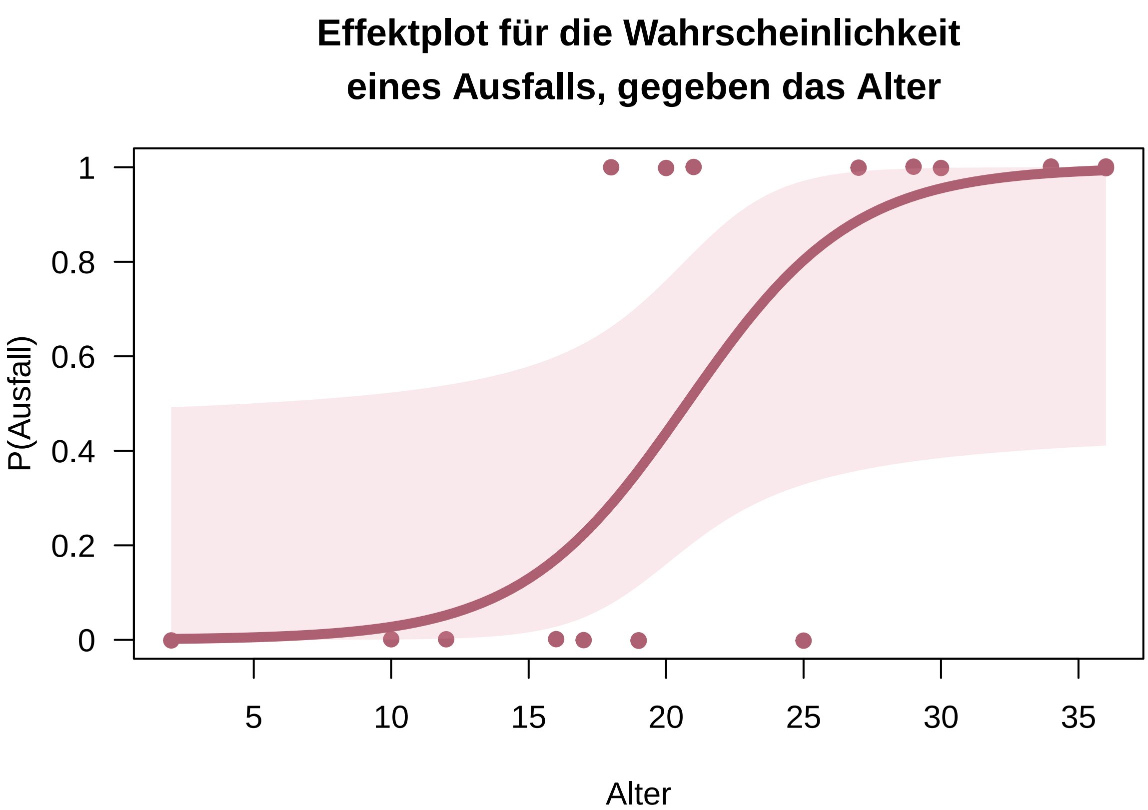
Table 6: Parameterschätzung des Modells: Ausfall ~ Alter

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-6.874	3.483	-1.974	0.04841
Alter	0.3312	0.1687	1.964	0.04957

Beide Parameter sind knapp signifikant auf dem 0.05–Niveau.

Die Effektstärke wird mittels "binärem Regressionsplot" dargestellt:

```
binreg_plot(model, pred_range ="xlim", jitter = 0.01,
            main = "Effektplot für die Wahrscheinlichkeit\n eines Ausfalls, gegeben das Alter")
```



Der Einfluss von `Alter` auf Wahrscheinlichkeit von `Ausfall` ist am stärksten im mittleren Wertebereich von Alter (ca. 10–30 Monate). Bei neueren Lüftern (unter 10 Monate) geht die Ausfallswahrscheinlichkeit gegen 0, bei älteren (über 30 Monate) gegen 1. Auffällig ist die große Schwankungsbreite - das Modell liefert also sehr ungenaue Vorhersagen.

Modelldiagnostik

Die Devianzanalyse liefert:

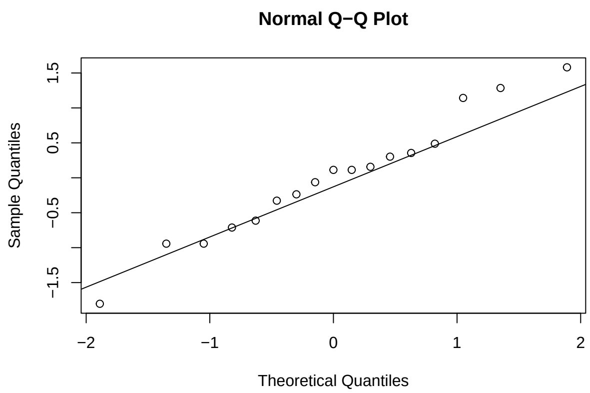
```
pander(Anova(model), caption = "")
```

	LR Chisq	Df	Pr(>Chisq)
Alter	11.47	1	0.0007088

Das Modell hat demnach einen signifikant höheren Erklärungswert als das konstante Modell.

Überprüfung der Normalverteilung der Residuen:

```r
qqnorm(residuals(model))
qqline(residuals(model))
```



Es sind kaum Auffälligkeiten zu erkennen (ein wenig an den Rändern).

Prognose

Die Ausfallsprognose für Lüfter mit einem Alter von 25, 30 und 35 Monaten lautet:

```r
tmp = c(25, 30, 35)
pred = predict(model, newdata = data.frame(Alter = tmp), type = "response")
pander(cbind(Alter = tmp, `P(Ausfall)` = pred))
```

Alter	P(Ausfall)
25	0.8032
30	0.9553
35	0.9912

Laut dem Modell ist damit zu rechnen, dass vier von fünf Lüftern ausgefallen sind, wenn ihr Alter 25 Monate erreicht. Nach 30 Monaten beträgt die geschätzte Ausfallswahrscheinlichkeit schon über 95%, nach 35 Monaten funktionieren Lüfter nur noch in Ausnahmefällen.