Fortgeschrittene Methoden der Biometrie

– Lineares Modell –

IMB
TU Dresden

WS 2021/22

Übungsblatt identisch zu unserem!

Inhaltsverzeichnis

1 Datenexploration und grafische Darstellung in R

Jede statistische Auswertung sollte mit der **Exploration der Daten** beginnen: man erkundet die Daten und versucht, ein Bild von den Verteilungen der Messgrößen und deren Korrelationen zu erhalten. Dazu eignen sich besonders grafische Darstellungen der Daten.

Einlesen der Daten. Starten Sie RStudio und setzen Sie ggf. das Arbeitsverzeichnis von R, falls Sie ein eigenes Verzeichnis für dieses R-Praktikum benutzen mittels Session › Set Working Directory. Beginnen Sie ein neues R-Skript mittels Ctrl + ⇧ + N.

Der R-Datensatz `airquality` enthält Messungen der Ozonkonzentration in New York über einen Zeitraum von fünf Monaten und enthält unter anderem die gemessene Ozonkonzentration, die Temperatur, den Monat und Tag, an dem die Messung stattfand. Schauen Sie sich den Hilfetext zum Datensatz mittels `?airquality` an, so dass Sie die Bedeutung der Spalten verstehen! Legen Sie nun eine Kopie des Datensatzes in Ihren Arbeitsbereich an, wobei Sie alle Beobachtungen (Zeilen) mit fehlenden Werten auslassen:

```
data(airquality)
airquality <- airquality[complete.cases(airquality),]
```

Aufgabe 1

Die Temperatur ist in Fahrenheit angegeben. Rechnen Sie die Temperatur in das uns geläufigere °C um und überschreiben Sie die ursprünglichen Werte. Die Umrechnung von Fahrenheit nach Celsius ist

$$°C = \frac{°F - 32}{1.8}$$

Fügen Sie dem Datensatz `airquality` eine neue Spalte `Ozone2` hinzu, die die Ozonkonzentration anhand des Schwellwertes 55 ppb in die zwei Klassen „niedrig“ und „hoch“ einteilt. Zur Einteilung einer metrischen Größe in Klassen steht Ihnen in R die Funktion `factor` zur Verfügung:

`factor(<Bedingung> , labels=c(<Name 1>,<Name 2>))`

Dabei geben Sie eine Bedingung an, anhand derer Sie die Daten unterteilen wollen. Zusätzlich können Sie mit der Option `labels=` den beiden Klassen Namen zuordnen.

Wir bearbeiten den Dataframe `airquality` folgendermaßen:

```
airquality$Ozone2 <- factor(airquality$Ozone > 55,
                            labels = c("niedrig", "hoch"))
airquality$Temp <- (airquality$Temp - 32)/1.8
```

Einen Überblick über die Verteilung der Daten bekommen Sie durch:

```
summary(airquality)
```

```
##      Ozone           Solar.R           Wind             Temp
##  Min.   :  1.0   Min.   :  7.0   Min.   : 2.30   Min.   :13.89
##  1st Qu.: 18.0   1st Qu.:113.5   1st Qu.: 7.40   1st Qu.:21.67
##  Median : 31.0   Median :207.0   Median : 9.70   Median :26.11
##  Mean   : 42.1   Mean   :184.8   Mean   : 9.94   Mean   :25.44
##  3rd Qu.: 62.0   3rd Qu.:255.5   3rd Qu.:11.50   3rd Qu.:29.17
##  Max.   :168.0   Max.   :334.0   Max.   :20.70   Max.   :36.11
##      Month            Day            Ozone2
##  Min.   :5.000   Min.   : 1.00   niedrig:80
##  1st Qu.:6.000   1st Qu.: 9.00   hoch   :31
##  Median :7.000   Median :16.00
##  Mean   :7.216   Mean   :15.95
##  3rd Qu.:9.000   3rd Qu.:22.50
##  Max.   :9.000   Max.   :31.00
```

Insbesondere sollten Sie der Zusammenfassung entnehmen können, dass von den insgesamt 111 Messungen 31 eine hohe Ozon-Belastung haben. Die mittlere Temperatur sollte 25.44 °C betragen.

1.1 Univariate Plots

Metrische Größen. Zur Darstellung von metrischen Größen eignen sich **Histogramme** und **Boxplots**. In R erzeugen Sie diese Diagramme beispielsweise mit:

```
boxplot(airquality$Ozone)
hist(airquality$Temp)
```

Bei wenigen Datenpunkten ist ein simpler **Dotplot** geeigneter, bei dem jeder Datenpunkt einzeln dargestellt wird:

```
stripchart(airquality$Temp, method="jitter")
```

Durch das Argument `method='jitter'` werden die Datenpunkte ein wenig gestreut, um gleiche Werte besser sichtbar zu machen.

Kategoriale Größen. Kategoriale Variablen können durch **Balkendiagramme** visualisiert werden:

```
plot(airquality$Ozone2)
```

Die `plot`-Funktion stellt die Daten nur dann als Balkendiagramm dar, wenn klar ist, dass es sich um eine kategoriale Variable handelt. Um Zahlenwerte als kategoriale Größe zu behandeln, verwenden Sie die Funktion `as.factor`, etwa:

```
plot(as.factor(airquality$Month))
```

1.2 Bivariate Plots

Metrisch vs metrisch. Bei zwei metrischen Messgrößen bietet sich ein **Streuplot** an, wobei Sie eine Messgröße auf der x-Achse und eine andere auf der y-Achse auftragen.

```
plot(Ozone ~ Temp, data=airquality)
```

Der linke Teil der **Formel** `Ozone~Temp` wird dabei auf der y-Achse dargestellt, der rechte auf der x-Achse.

Um zu dem Streuplot eine **Trendlinie** hinzuzufügen, können Sie die R-Funktion `lowess` verwenden, die allerdings *nicht* das Formel-Interface benutzt (d.h. die Funktion erwartet als Argumente zunächst die x-Variable, dann separat die y-Variable und Sie müssen für jede Spalte den Dataframe mit angeben):

```
lines(lowess(x = airquality$Temp, y = airquality$Ozone),
      col = "darkgreen", lty = "dashed")
```

Der Parameter `lty=` der Funktion `lines` bestimmt den Linientyp: `1` oder `solid` für durchgezogene, `2` oder `dashed` für gestrichelte und `3` oder `dotted` für gepunktete Linien.

Metrisch vs kategorial. Ist die metrische Größe die Zielvariable, dann eignen sich **Boxplot** oder – wenn man nur wenige Beobachtungen hat – ein **Dotplot**.

```
boxplot(Ozone ~ Month, data=airquality)
stripchart(Ozone ~ Month, data=airquality, vertical=TRUE, method="jitter")
```

Kategorial vs metrisch. Bei kategorialer Zielvariable sollte diese Größe auch auf der y-Achse abgebildet werden. Dazu eignet sich beispielsweise ein **konditionaler Dichteplot**:

```
cdplot(Ozone2 ~ Temp, data=airquality)
```

Kategorial vs kategorial. Bei zwei kategorialen Messgrößen können die Daten in einer **Kreuztabelle** (Kontingenztabelle) erfasst werden:

```
xtabs(~Month + Ozone2, data=airquality)
```

Beachten Sie, dass in diesem Fall beide Variablen, die dargestellt werden sollen, rechts vom Formelsymbol ~ mit einem + verbunden werden. Kreuztabellen können auch grafisch durch einen **Mosaikplot** dargestellt werden. Die Standardfunktion in R ist

```
mosaicplot(Month ~ Ozone2, data=airquality)
```

Eine Variante dieses Plots findet sich im R-Package `vcd` („visualizing categorical data“).[1] Laden Sie das Paket entweder in RStudio im Packages-Fenster rechts unten oder direkt in der Konsole durch den R-Aufruf:

```
library(vcd)
```

Nach dem Laden steht Ihnen die folgende Funktion zur Visualisierung von Kreuztabellen zur Verfügung:

```
doubledecker(Ozone2~Month, data=airquality)
```

Aufgabe 2

Visualisieren Sie mithilfe passender Diagramme den Zusammenhang zwischen:

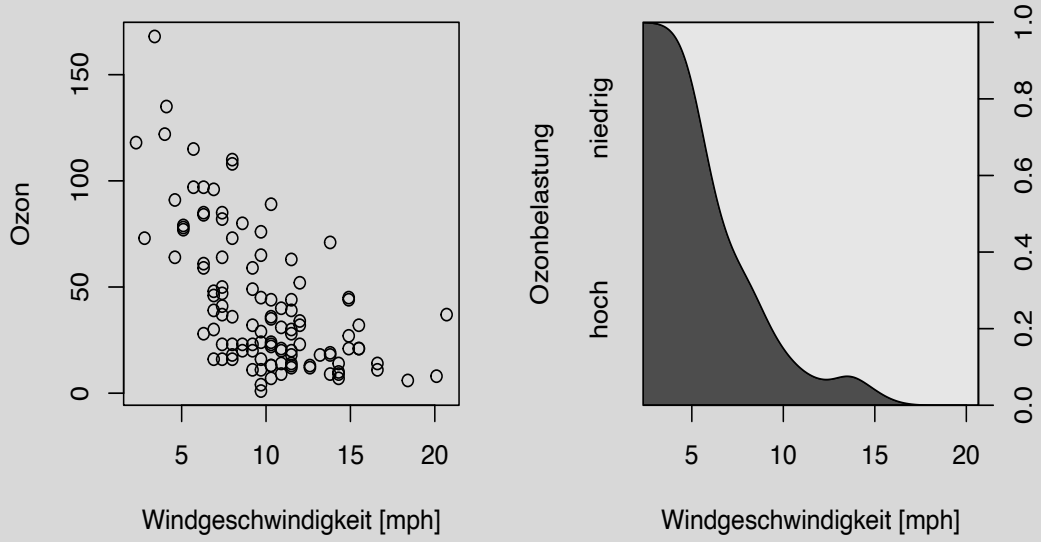
- Ozonwert vs Windgeschwindigkeit,
- Ozonbelastung (hoch/niedrig) vs Windgeschwindigkeit,
- Windgeschwindigkeit vs Temperatur und vs Monat,
- Ozonbelastung (hoch/niedrig) vs Sonneneinstrahlung.

Wie können Sie die beobachteten Zusammenhänge beschreiben und deuten?

[1] Falls das Paket noch nicht installiert ist, installieren Sie es einmalig vor dem Laden durch `install.packages("vcd")` direkt in R oder über Tools › Install Packages... in RStudio

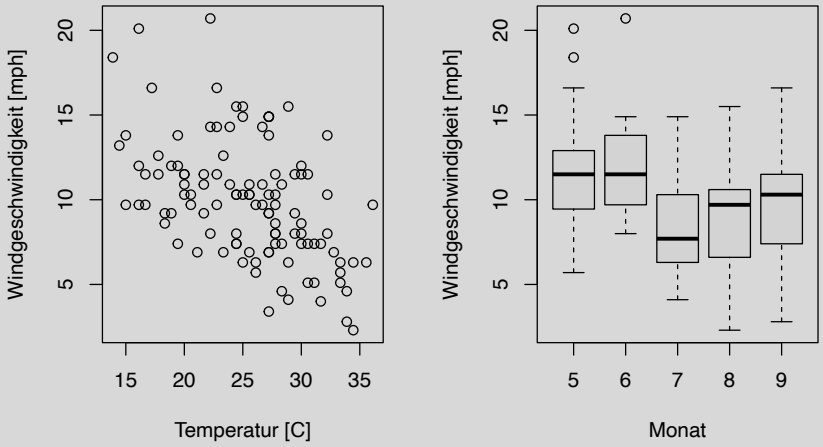
Ozonwert und Windgeschwindigkeit: Mit zunehmendem Wind nehmen die Ozonwerte im Mittel ab.

```
par(mfrow = c(1,2))
plot(Ozone ~ Wind, data=airquality,
     xlab="Windgeschwindigkeit [mph]", ylab="Ozon")
cdplot(Ozone2 ~ Wind, data=airquality,
       xlab="Windgeschwindigkeit [mph]",ylab="Ozonbelastung")
```



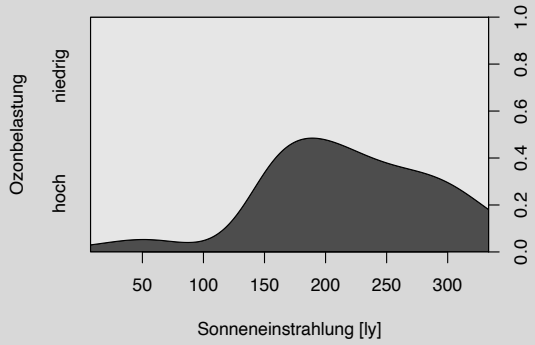
Windgeschwindigkeit vs Temp bzw. Monat: Bei höheren Temperaturen herrscht im Mittel auch eine niedrigere Windgeschwindigkeit.

```
par(mfrow = c(1,2))
plot(Wind ~ Temp, data = airquality, xlab = "Temperatur [C]",
     ylab = "Windgeschwindigkeit [mph]")
boxplot(Wind ~ Month, data=airquality,
        xlab="Monat", ylab="Windgeschwindigkeit [mph]")
```



Ozonbelastung und Sonneneinstrahlung: Tage mit niedriger Sonneneinstrahlung haben selten eine hohe Ozonbelastung. Allerdings zeigt sich kein eindeutiger linearer Trend über das Spektrum der Sonneneinstrahlungswerte. Bei Tagen mit mittlerer Sonneneinstrahlungswerten ist die Häufigkeit hoher Ozonbelastung am größten.

```
cdplot(Ozone2 ~ Solar.R, data=airquality,
       xlab="Sonneneinstrahlung [ly]", ylab="Ozonbelastung")
```



2 Einfache lineare Regression

Die einfache Regression heißt einfach, weil nur ein einziger Prädiktor verwendet wird – im Gegensatz zur multiplen Regression, in der mehrere Prädiktoren verwendet werden.

2.1 Explorative Datenanalyse

Hintergrund. In der Sportmedizin wurde ein Experiment an einer Gruppe von 21 Sportstudenten aus dem 1. Semester durchgeführt. Jedem Studenten wurde eine Leistung zwischen 0 Watt (Leerlauf) und 200 Watt vorgegeben, die er 20 Minuten lang auf dem Fahrradergometer treten sollte. Direkt nach Ablauf der 20 Minuten wurde bei jedem der 21 Sportstudenten die Herzfrequenz gemessen.

Bei der Regression geht es nun darum zu analysieren, inwieweit sich die am Ende gemessene Herzfrequenz (Y) durch die vorgegebene Belastung in Watt (X) erklären lässt.

Aufgabe 3

Lesen Sie den Datensatz `ergo.dat` ein (Tipp: in RStudio gibt es einen Knopf Import Dataset) und verschaffen Sie sich mithilfe des `summary`-Befehls einen ersten Überblick über die Daten.

Zeichnen Sie Histogramme und Boxplots der gemessenen Herzfrequenzen und Belastungen und schauen Sie sich die Beziehung zwischen den beiden Variablen in einem Streudiagramm an.

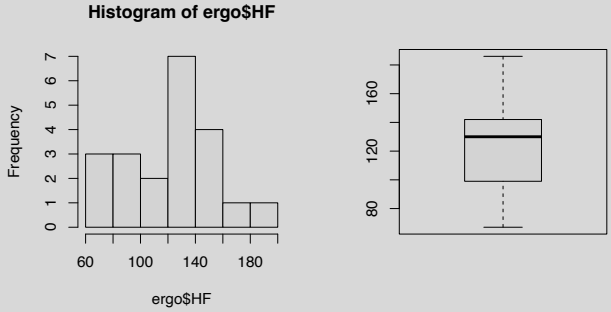
Datensatz einlesen:

```
ergo <- read.table("data/ergo.dat")
```

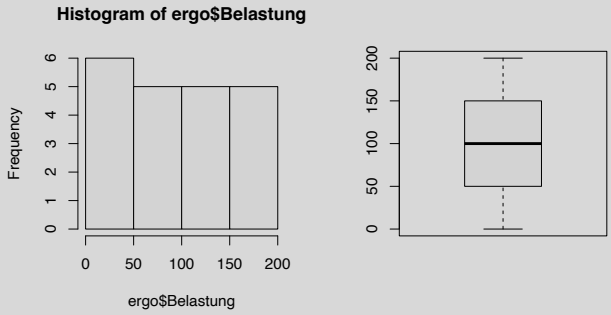
```r
summary(ergo)
```

```
##        HF        Belastung
##  Min.   : 67   Min.   :  0
##  1st Qu.: 99   1st Qu.: 50
##  Median :130   Median :100
##  Mean   :123   Mean   :100
##  3rd Qu.:142   3rd Qu.:150
##  Max.   :186   Max.   :200
```

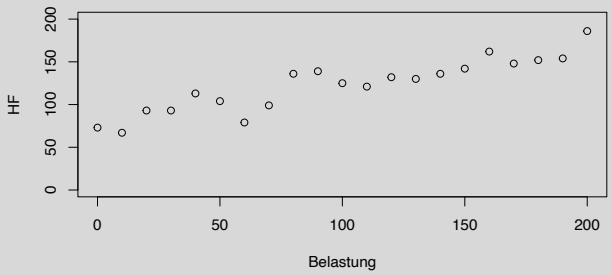
```r
par(mfrow = c(1,2))
hist(ergo$HF)      # Histogramm der Herzfrequenzen
boxplot(ergo$HF)   # Boxplot der Herzfrequenzen
```




```r
hist(ergo$Belastung)      # Histogramm der Belastungen
boxplot(ergo$Belastung)   # Boxplot der Belastungen
```



```r
par(mfrow = c(1,1))
plot(HF ~ Belastung, data = ergo, ylim = c(0, 200)) #Streudiagramm
```



Wie zu erwarten steigt die gemessene Herzfrequenz im Mittel, wenn die Belastung höher wird.

2.2 Visuelle Modellanpassung

Wir legen den Daten nun ein lineares statistisches Modell zugrunde, welches im systematischen Teil eine Geradengleichung beschreibt:

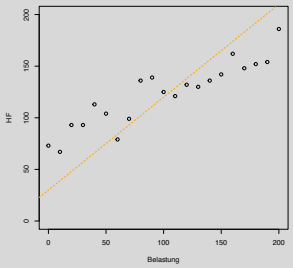
$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

y_i bezeichnet den i-ten Messwert der abhängigen Variable (Herzfrequenz), x_i den i-ten Wert des Prädiktors (Belastung).
Das Residuum ϵ_i ist die Abweichung der i-ten Beobachtung von der vom Modell beschriebenen Geraden.
β_0 und β_1 sind die Parameter im Modell. β_0 bezeichnet dabei den Schnittpunkt mit der y-Achse (engl.: intercept), β_1 ist der Anstieg der Regressionsgeraden. Dieser gibt an, um wie viel sich der Wert von y_i im Mittel erhöht, wenn x_i um eine Einheit erhöht wird.

Unter dem Begriff *Modellanpassung* versteht man, dass man Werte für β_0 und β_1 bestimmt, die die Daten möglichst gut beschreiben.

Zum Beispiel, mit folgendem R-Code sehen Sie die Gerade zu den Werten $\beta_0 = 30$ und Steigung $\beta_1 = 0.9$:

```r
plot(HF ~ Belastung, data = ergo, ylim = c(0, 200))
abline(a = 30, b = 0.9, col="orange", lty="dashed")
```



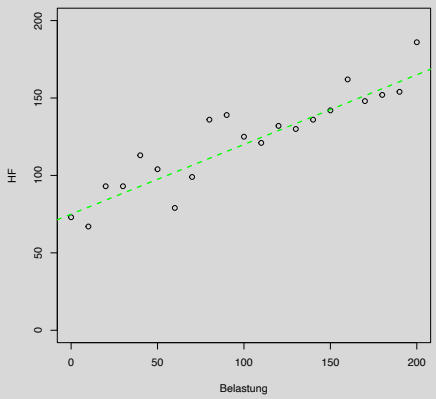
Rein visuell passt die Gerade noch nicht besonders gut zu den Datenpunkten.

Aufgabe 4

Versuchen Sie, bessere Werte für β_0 und β_1 zu finden, die nach Ihrem Geschmack den Trend in den Daten besser widerspiegeln!

Tipp: Um die Geraden auseinanderzuhalten, können Sie die Farben variieren. Wenn es trotzdem unübersichtlich wird, können Sie den `plot`-Befehl für den Streuplot `HF` $\sim$ `Belastung` erneut aufrufen!

```
plot(HF ~ Belastung, data = ergo, ylim = c(0, 200)) #Streudiagramm
abline(a = 75, b = 0.45, col="green", lty="dashed", lwd = 2)
```



2.3 Methode der kleinsten Quadrate – selbst gemacht

Die rein visuelle Methode ist natürlich stark abhängig vom subjektiven Empfinden. Sicherlich sind Sie nicht alle zum gleichen Ergebnis gekommen. Mit der Methode der kleinsten Quadrate werden die Schätzwerte für β_0 und β_1 so gewählt, dass die Summe der quadratischen Abweichungen der Daten von der Modellgerade minimal ist, und dies ist ein objektives Kriterium zur Schätzung der Parameter β_0 und β_1.

Für die Summe der quadratischen Abstände benötigen Sie für jeden Datenpunkt den vertikalen Abstand zur Geraden. Die vertikalen Abstände lassen sich wie folgt berechnen:

```
predicted <- 30 + 0.9 * ergo$Belastung # Modellvorhersage
diff <- ergo$HF - predicted # Residuen
```

Die Summe der quadratischen Abweichungen lässt sich folgendermaßen berechnen:

```
sum(diff^2)
```

Dies ist ein Beispiel für vektorisiertes Rechnen in R: Die Quadrierung wird automatisch auf jedes Element des Vektors `diff` angewendet. Der `sum`-Befehl schließlich summiert alle Zahlen des Vektors.

Die Summe der quadratischen Abstände wird meist mit RSS (residual sum of squares) abgekürzt. Die folgende Funktion fasst die vorherigen Schritte zusammen. Sie berechnet die RSS für gegebenes β_0 und β_1 und visualisiert die Abstände:

```
drawLine <- function(beta0, beta1) {
   predicted <- beta0 + beta1 * ergo$Belastung
   diff <- ergo$HF - predicted
   rss <- sum(diff^2)
   yRange <- range(0, predicted, ergo$HF)

   plot(HF ~ Belastung, ergo, ylim = yRange)
   abline(beta0, beta1, col = "red", lty = "dashed")
   segments(ergo$Belastung, predicted, ergo$Belastung, ergo$HF)

   rss
}
```

Die Funktion wird einmal in R ausgeführt und kann dann mit `drawLine(beta0,beta1)` für beliebige Werte `beta0` und `beta1` aufgerufen werden.

Aufgabe 5

Berechnen Sie mithilfe der gegebenen Funktion die Summe der quadratischen Abweichungen für $\beta_0 = 30$ und $\beta_1 = 0.9$.

Probieren Sie weitere Werte für β_0 und β_1 aus, und notieren Sie sich jeweils die zugehörige Summe der quadratischen Abweichungen. Welches der von Ihnen gewählten Wertepaare liefert die geringste Summe der quadratischen Abweichungen? Vergleichen Sie mit Ihrem Nachbarn!

Die Funktion wird mit `drawLine(30,0.9)` aufgerufen. Sie zeichnet die Ausgleichsgerade und gibt die Summe der quadratischen Abweichungen (RSS, engl. *residual sum of squares*) aus. $RSS_{30,0.9} = 18102.0$.

Eine bessere Anpassung wird zum Beispiel erreicht durch $\beta_0 = 75$ und $\beta_1 = 0.45$ mit $RSS_{75,0.45} = 3265.5$.

2.4 Methode der kleinsten Quadrate – durch R

R bietet eine Funktion, die die Werte für β_0 und β_1 berechnet, für welche die Summe der quadratischen Abweichungen minimal ist – den `lm`-Befehl. Die so geschätzten Werte für β_0 und β_1 werden mit $\hat{\beta}_0$ und $\hat{\beta}_1$ bezeichnet.

Heute sind für Sie nur die ersten beiden Argumente des `lm`-Befehls relevant. Das erste Argument spezifiziert das statistische Modell, das angepasst werden soll. Das zweite gibt an, welche Daten (in Ihrem Fall der `ergo`-Datensatz) benutzt werden sollen.

Das statistische Modell, das angepasst werden soll, wird in R über eine **Formel** spezifiziert. Eine Formel ist wie folgt aufgebaut:

`<abhängige Variable>` $\sim$ `<Prädiktoren, z.B. mit + verknüpft>`

Der y-Achsenabschnitt β_0 sowie ein Fehlerterm sind standardmäßig schon mit dabei. Im vorliegenden Fall gibt es nur einen Prädiktor, sodass das Modell $y_i = \beta_0 + \beta_1 \cdot x_i + \epsilon_i$ durch die Formel `HF` $\sim$ `Belastung` spezifiziert wird.

```
model1 <- lm(HF ~ Belastung, data = ergo)
model1 # zeigt Basisinformationen zum angepassten Modellobjekt


##
```

```
## Call:
## lm(formula = HF ~ Belastung, data = ergo)
##
## Coefficients:
## (Intercept)    Belastung
##     76.9567       0.4609
```

Wie nahe sind Sie mit Ihrer manuellen Geraden-Anpassung den optimalen Parametern gekommen?

Aufgabe 6

Interpretieren Sie die Zusammenfassung zum Modell `summary(model1)`. Geben Sie ein 95%-Konfidenzintervall zu den Modell-Parametern an. Welche Herzfrequenz-Werte sagt Ihr Modell für eine Belastung von 125W bzw. 1000W vorher? Können Sie ein Konfidenzintervall für diese Vorhersagen angeben? Tipp: siehe Vorlesung.

Das lineare Modell schätzt bei Leerlauf (0W Belastung) eine mittlere Herzfrequenz von 76.95671. Gemäß des geschätzten Modells steigt die mittlere Herzfrequenz pro 1W Belastung um 0.46. Der Anstieg ist statistisch signifikant (auf dem 5%-Signifikanzniveau):

```
summary(model1)
```

```
##
## Call:
## lm(formula = HF ~ Belastung, data = ergo)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.611  -7.311  -3.957   6.825  22.171
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 76.95671    5.34763   14.39 1.14e-11 ***
## Belastung    0.46091    0.04574   10.08 4.66e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.69 on 19 degrees of freedom
## Multiple R-squared:  0.8424,Adjusted R-squared:  0.8341
## F-statistic: 101.5 on 1 and 19 DF,  p-value: 4.659e-09
```

95%-Konfidenzintervall zu den Modell-Parametern mit `confint`. Das KI zum Anstieg enthält nicht die 0. Dies entspricht dem signifikanten Testergebnis in der `summary`-Ausgabe.

```
confint(model1)
```

```
##                  2.5 %     97.5 %
## (Intercept) 65.7639934 88.1494265
## Belastung    0.3651667  0.5566515
```

Modell-Vorhersagen können mittels der `predict`-Funktion zu einem Dataframe mit den vorgegebenen Prädiktorwerten angegeben werden. Vorsicht bei Extrapolation: 1000W wurden nicht beobachtet und liefern einen unglaubhaften Wert für die Herzfrequenz zurück. In diesem Bereich der Belastung ist das lineare Modell so also nicht einfach übertragbar.

```
vorhersage <- data.frame(Belastung = c(125, 1000))
vorhersage$HF <- predict(model1, newdata = vorhersage)
vorhersage
```

```
##   Belastung       HF
## 1       125 134.5703
## 2      1000 537.8658
```

Für diese vorhergesagten mittleren Herzfrequenzen bei gegebener Belastung können auch Konfidenzintervalle angegeben werden, um die Unsicherheit der Schätzung abzubilden:

```
predict(model1, newdata = vorhersage,interval = "confidence")
```

```
##        fit      lwr      upr
## 1 134.5703 128.2982 140.8425
## 2 537.8658 451.5029 624.2288
```

Mithilfe der `predict`-Funktion kann zusätzlich zum Mittelwert bei gegebender Belastung und seinem Konfidenzintervall auch ein Intervall vorhergesagt werden, in welchem 95% der **Einzelwerte** liegen:

```
predict(model1, newdata = vorhersage,interval = "prediction")
```

```
##        fit      lwr      upr
## 1 134.5703 107.2726 161.8681
## 2 537.8658 447.5088 628.2228
```

Dieses Intervall berücksichtigt nun die Varianz zwischen den Probanden.

Ergänzung: Modellvorhersagen

Fortgeschrittene Methoden der Biometrie

– Lineares Modell –

IMB
TU Dresden

WS 2023/24

Inhaltsverzeichnis

3 Modellvorhersagen und Modellgüte

3.1 Vorhersagen und Konfidenzintervalle

Wir arbeiten weiterhin mit dem `ergo`-Datensatz, für den Sie in der letzten Übung ein lineares Regressionsmodell angepasst haben.

Aufgabe 1

Welche Herzfrequenz-Werte sagt Ihr Modell für eine Belastung von 125W bzw. 1000W vorher? Sind diese Werte realistisch? Können Sie Konfidenzintervalle für diese Vorhersagen angeben?

Aufgabenstellung „vorher“ >> ich muss eine Vorhersage machen!

neueBelastung <- data.frame(Belastung=c(125,1000))

3.2 Modellgüte: Bestimmtheitsmaß B

Um die Anpassungsgüte des linearen Modells für die gegebenen Daten einschätzen zu können, hilft ein Blick auf das Bestimmtheitsmaß B, welches für die einfache lineare Regression dem Quadrat des Pearsonschen Korrelationskoeffizienten R^2 entspricht. Das Bestimmtheitsmaß B beschreibt den Anteil der durch das Modell erklärten Varianz an der Gesamtvarianz.

Aufgabe 2

Rufen Sie die Zusammenfassung Ihres linearen Modells auf und versuchen Sie, das Bestimmtheitsmaß B in der Ausgabe zu finden.

Bei unserem Übungsblatt ist noch Modellvorhersagen mit drauf
- Daher ist jetzt die Skalierung anders- Aufgaben/Inhaltlich aber gleich

Fortgeschrittene Methoden der Biometrie

– Lineares Modell –

IMB
TU Dresden

WS 2021/22

Inhaltsverzeichnis

3 Diagnostik im linearen Modell

Um aus einem angepassten Modell nützliche Schlussfolgerungen ziehen zu können, müssen wir zunächst prüfen, ob es die Daten adäquat beschreibt. Die Modelldiagnostik dient dazu, Verletzungen der Modellannahmen zu detektieren, die so gravierend sind, dass sie die Gültigkeit der Schlussfolgerungen in Frage stellen.

Die folgenden Punkte sollten überprüft werden:

1. Annahmen an Verteilung der Residuen
   - Die Residuen sind normalverteilt.
   - Die Varianz der Residuen ist konstant - unabhängig davon, wie groß der vorhergesagte Wert ist, streuen die tatsächlichen Beobachtungen mit der gleichen Amplitude nach oben und unten.
   - Die Residuen sind unabhängig voneinander.
2. Ungewöhnliche Beobachtungen (Ausreißer, Datenpunkte mit starkem Einfluss?)
3. Struktur des Modells (linearer Einfluss der Prädiktoren auf die Zielgröße?)

3.1 Diagnostische Plots

R bietet standardmäßig vier Diagnostikplots an, um die Annahmen zu überprüfen. Diese können Sie durch folgenden Befehl aufrufen:

```
plot(model1)
```

wobei in der Variable `model1` ein lineares Modell-Objekt gespeichert ist. Nun können Sie durch Drücken der Enter-Taste im Eingabefenster die Diagramme nach und nach abrufen. Wenn Sie alle vier Diagramme gleichzeitig betrachten wollen, können Sie das Bildfenster folgendermaßen erweitern:

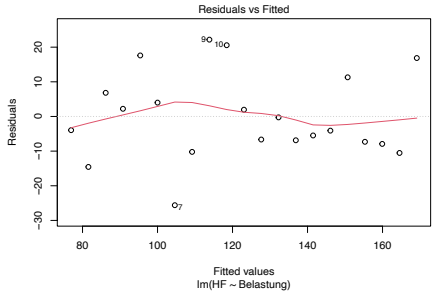
```
par(mfrow = c(2,2))
```

Jetzt werden die Diagramme in einem 2x2-Gitter angezeigt. Um das Bildfenster wieder in den Ursprungszustand zurück zu versetzen:

```
par(mfrow = c(1,1))
```

Im Folgenden sehen Sie die vier Diagnostikplots für das in der letzten Übung besprochene Beispiel des *ergo*-Datensatzes.

3.1.1 Residuals vs Fitted



Aufgetragen sind die angepassten Werte der Zielgröße gegen die Residuen, d.h. die Abweichung der angepassten von den gemessenen Werten.

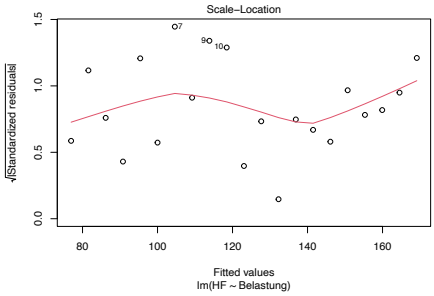
In diesem Plot kann man auf 2 Dinge achten:

- Ist die Struktur des Modells korrekt? Wenn für Teilintervalle der fitted values die Residuen auffällig oft größer oder kleiner 0 sind, so ist dies ein Hinweis auf ein Problem mit der Struktur des Modells. Dies könnte zum Beispiel der Fall sein, wenn der Einfluss eines Prädiktors eigentlich eher quadratisch ist als linear.

  Die rote Linie sollte ungefähr waagerecht in der Nähe der Nulllinie laufen.

- Ist die Varianz konstant? Typische Annahmenverletzung besteht zum Beispiel darin, dass sich die Varianz trichterförmig erweitert - oft ist bei größerem vorhergesagtem Wert die Streuung größer

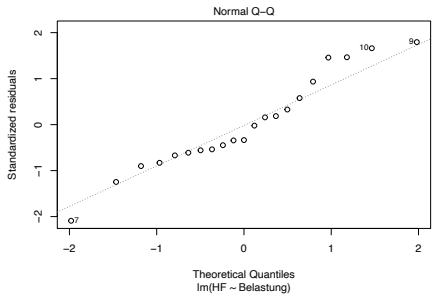
3.1.2 Scale location



Bei diesem Diagramm sind auf der x-Achse wieder die angepassten Werte aufgetragen. Auf der y-Achse ist nun die Wurzel des Betrages der standardisierten Residuen aufgetragen. Der Erwartungswert dieser Größe ist etwa 0.8. Deswegen sollten die y-Werte über den gesamten Wertebereich der vorhergesagten Werte im Mittel etwa 0.8 sein. Die

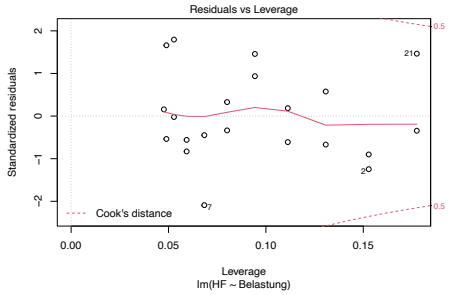
rote Linie ist eine Schätzung dieses Mittelwertes, und sollte deshalb mehr oder weniger parallel zur x-Achse auf Höhe $y = 0.8$ verlaufen. Steigt die Linie hingegen an, ist das ein Zeichen dafür, dass die Annahme der Varianzgleichheit verletzt ist.

3.1.3 Normal Q-Q



In Q-Q-Plot kann man schauen, ob etwas gegen die Annahme spricht, dass die Residuen normalverteilt sind. Es werden die bei einer Normalverteilung erwarteten gegen die tatsächlich beobachteten Residuen dargestellt. Die Werte sollten ungefähr eine Gerade bilden.

3.1.4 Residuals vs Leverage



In diesem Plot kann man nach ungewöhnlichen/einflussreichen Beobachtungen Ausschau halten. Einflussreich in dem Sinne, dass sie die Schätzung der Modellparameter besonders stark beeinflussen.

Auf x- und y-Achse sind Größen aufgetragen, die jeweils den Einfluss eines Falls widerspiegeln, und die relativ unabhängig voneinander sind. Auf der x-Achse ist die Leverage eines jeden Datenpunktes aufgetragen. Das ist ein Maß dafür, wie stark der Fall aufgrund seiner Prädiktorkonstellation die Parameterschätzung beeinflussen kann (Beispiel: Wenn in einer Varianzanalyse in einer Gruppe deutlich weniger Fälle sind als in den anderen

Gruppen, so beeinflussen die Fälle in der kleinen Gruppe pro Fall die Schätzung des Gruppenmittelwerts stärker). Auf der y-Achse sind die standardisierten Residuen aufgetragen. Cook's Distance ist eine Statistik, die sich für jede Beobachtung berechnen lässt. In sie gehen das standardisierte Residuum sowie die Leverage ein. Die Cook's Distance ist höhenlinienartig in dem Plot durch die gestrichelten roten Linien dargestellt. Punkte mit einer großen Cook's Distance sollten noch einmal kritisch geprüft werden!

3.2 Beispiele

Aufgabe 1

Laden Sie den Datensatz `modelldiagnostik.dat`. Die Zielgröße wird mit `y`, der Prädiktor mit `x` bezeichnet.

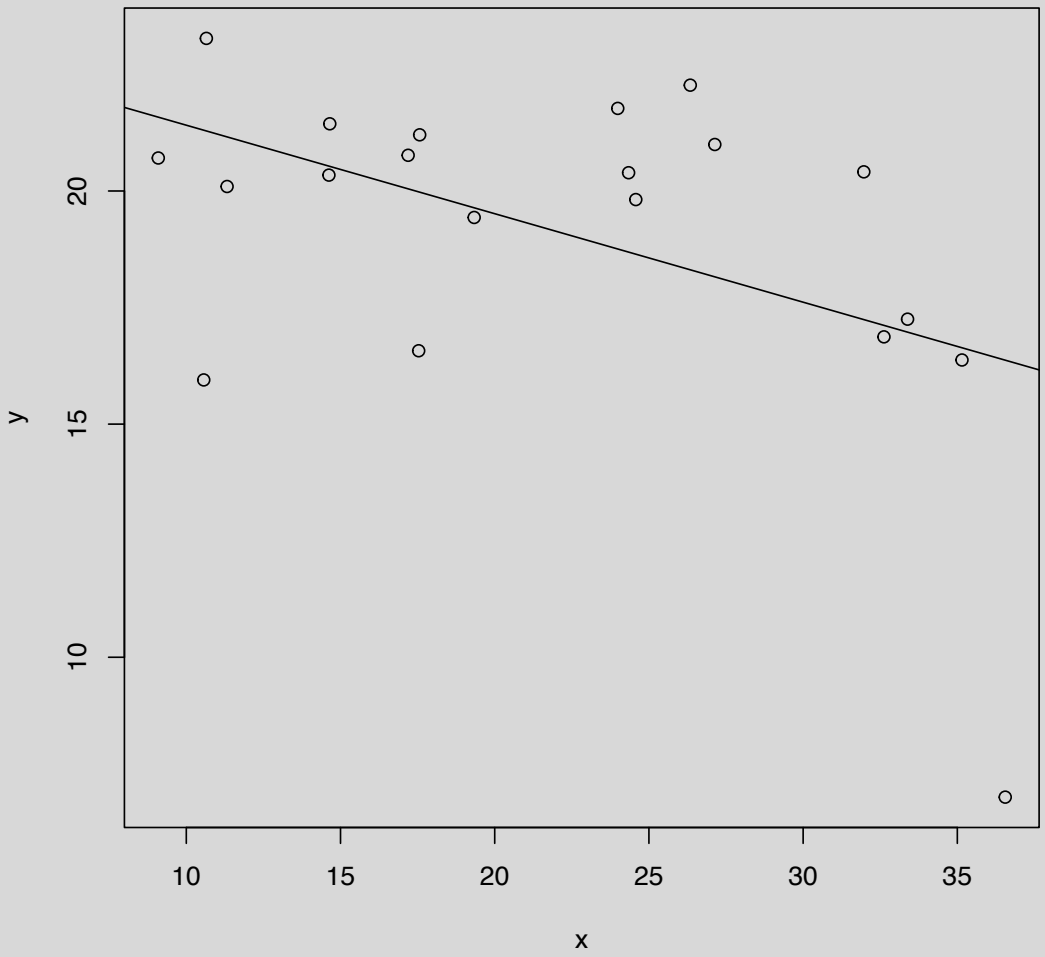
Passen Sie ein lineares Modell an die Daten an. Gibt es einen signifikanten Einfluss des Prädiktors? Stellen Sie die Daten in einem Streudiagramm dar und zeichnen Sie die angepasste Gerade mittels `abline` ein. Gibt es Auffälligkeiten in den Diagnostikplots zum einfachen linearen Modell?

Die geschätzte Steigung ist negativ und stat. signifikant von 0 verschieden (auf dem 5%-Signifikanzniveau). Eine Beobachtung erscheint als ein Ausreißer, dessen y-Wert nach unten hin von dem allgemeinen linearen Trend abweicht. Dieser Ausreißer lässt sich mithilfe der Diagnostikplots identifizieren, er ist bspw. an seiner großen *Cook's distance* zu erkennen.

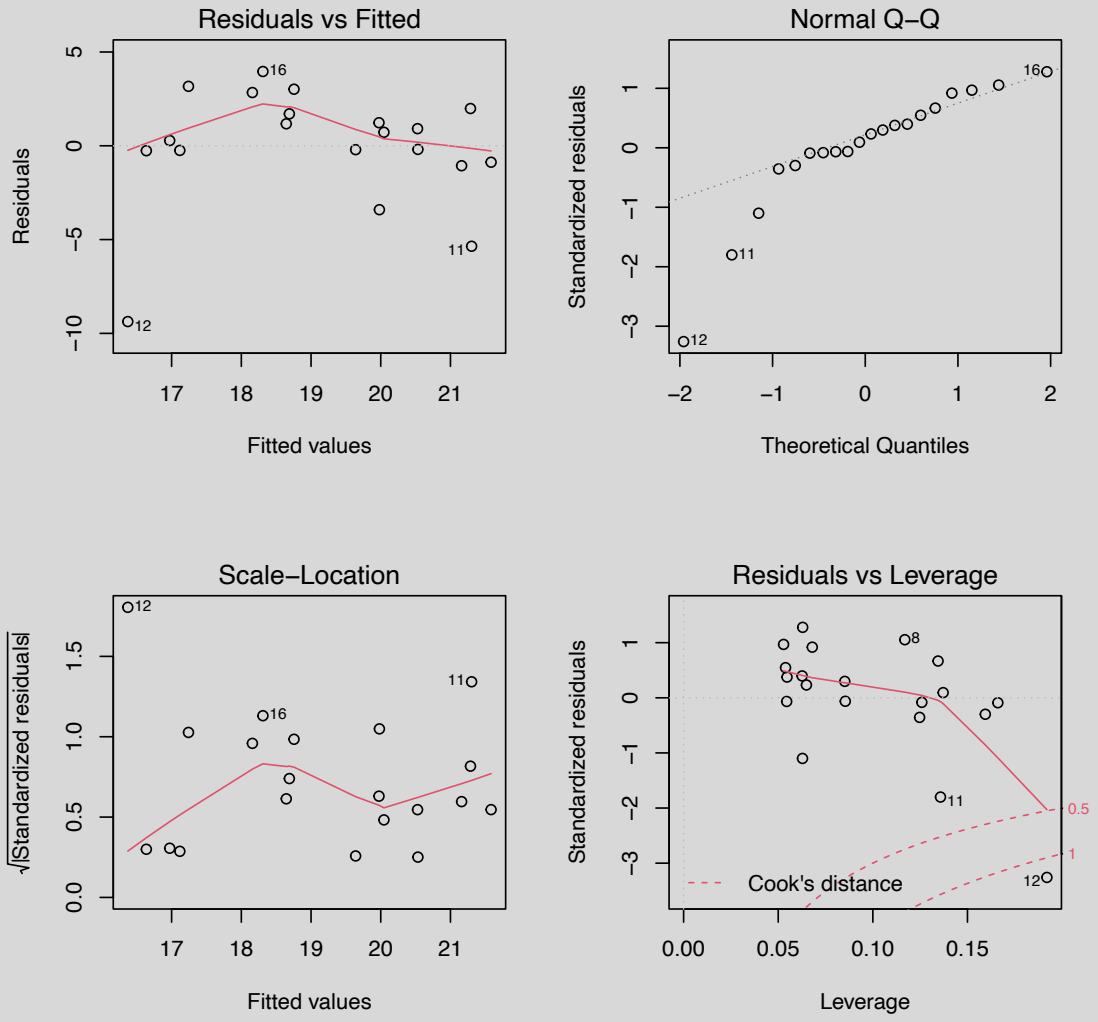
```r
modelldiagnostik <- read.table(file = "data/modelldiagnostik.dat")
model2 <- lm(y ~ x, data = modelldiagnostik)
summary(model2)
```

```
##
## Call:
## lm(formula = y ~ x, data = modelldiagnostik)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3716 -0.4158  0.4993  1.7731  3.9597
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.30866    1.94431  11.988 5.13e-10 ***
## x           -0.18978    0.08244  -2.302   0.0335 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.199 on 18 degrees of freedom
## Multiple R-squared:  0.2274,Adjusted R-squared:  0.1845
## F-statistic: 5.299 on 1 and 18 DF,  p-value: 0.03349
```

```r
plot(y ~ x, data = modelldiagnostik)
abline(model2)
```




```
par(mfrow = c(2,2))
plot(model2)
```




```
par(mfrow = c(1,1))
```

Aufgabe 2

Im vorherigen Beispiel scheint ein Ausreißer großen Einfluss auf die Schätzung des Modells zu haben. Passen Sie das Modell noch einmal ohne diesen Datenpunkt an.

Am einfachsten ist es, wenn Sie in der Funktion `lm` den Parameter `subset=` benutzen: Geben Sie entweder einen Vektor mit den Zeilennummern der Beobachtungen an,

Aufgabe 2 (cont)

die Sie behalten wollen *oder* einen Vektor mit den negativen Zeilennummern der Beobachtungen, die Sie entfernen wollen!

Ist die Steigung immer noch signifikant (bei $\alpha = 0.05$) von 0 verschieden?

Wenn die auffällige Beobachtung für die lineare Regression ausgelassen wird, ist die geschätzte Steigung zwar immer noch negativ, aber auf dem 5%-Signifikanzniveau nicht mehr signifikant.

```
model3 <- lm(y ~ x, data = modelldiagnostik, subset = -12)
summary(model3)
```

```
##
## Call:
## lm(formula = y ~ x, data = modelldiagnostik, subset = -12)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6560 -1.0432  0.2932  1.3056  2.8831
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21.41904    1.33695   16.02 1.08e-11 ***
## x           -0.07717    0.05890   -1.31    0.208
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.108 on 17 degrees of freedom
## Multiple R-squared:  0.09171,Adjusted R-squared:  0.03828
## F-statistic: 1.716 on 1 and 17 DF,  p-value: 0.2076
```

Aufgabe 3

Laden Sie den Datensatz `modelldiagnostik2.dat`. Die Zielgröße ist wieder mit `y`, der Prädiktor mit `x` bezeichnet.

Stellen Sie die Daten in einem Streudiagramm dar. Passen Sie bitte ein lineares Modell an die Daten an und zeichnen Sie die angepasste Gerade hinein.

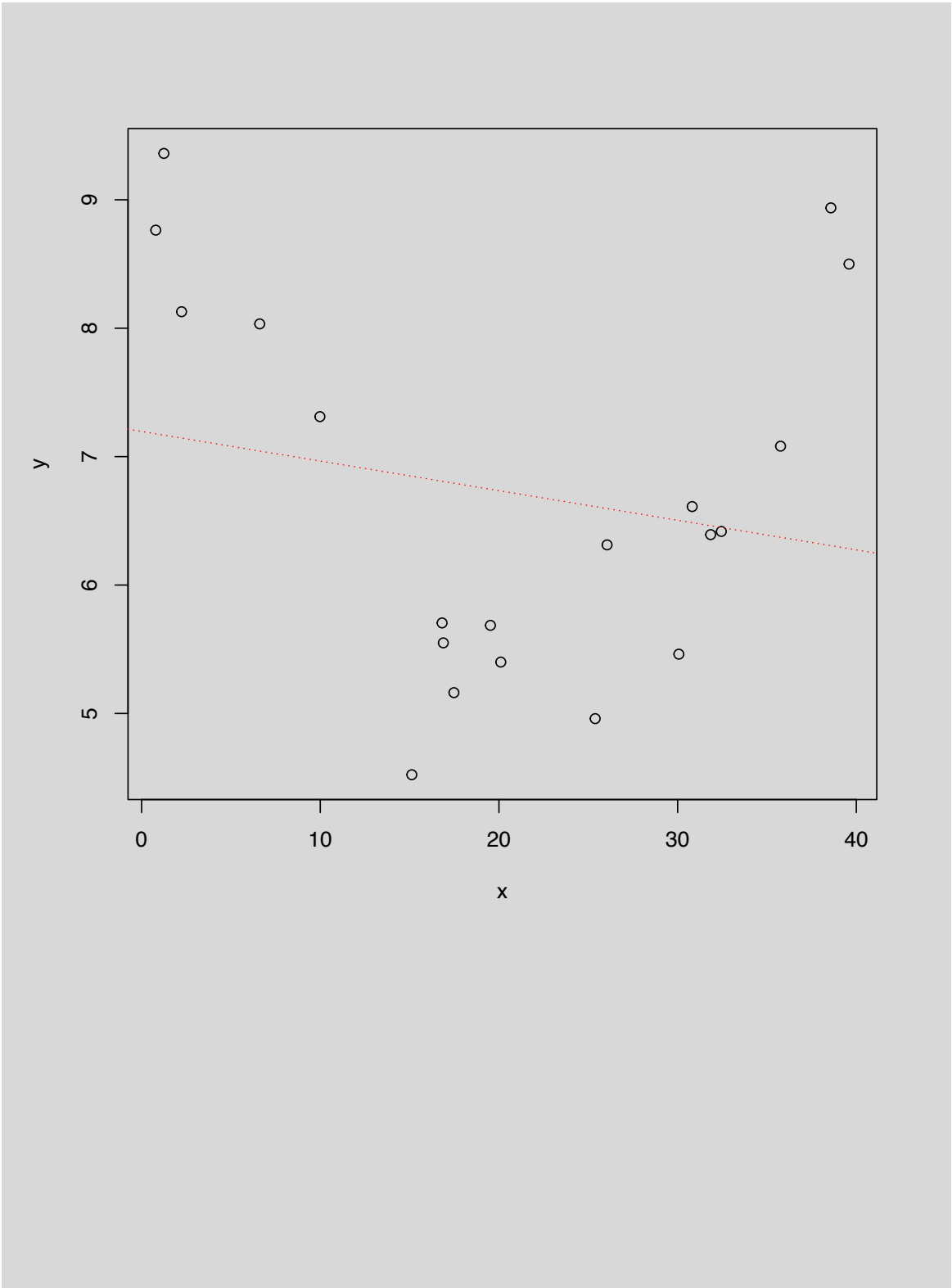
Aufgabe 3 (cont)

Wie beurteilen Sie die diagnostischen Plots?

Die Punkte des Datensatzes liegen entlang einer quadratischen Parabel. Die einfache lineare Regression erkennt diese Systemtik nicht und legt eine schwach abfallende Gerade durch die Punkte. Der R^2-Wert ist entsprechend nahe 0, hier: 0.0377407, was auf einen schlechten Fit hinweist.

Die Diagnostik-Plots zu dem einfachen linearen Modell zeigen den quadratischen Trend in den Residuen, insbesondere im *Residuals vs Fitted*-Plot.

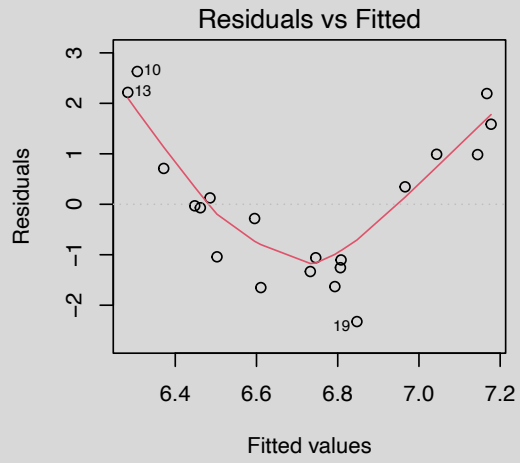
```
modelldiagnostik2 <- read.table(file =  "data/modelldiagnostik2.dat")
plot( y ~ x, data = modelldiagnostik2)

model4 <- lm( y ~ x, data = modelldiagnostik2)
abline(model4, col = "red", lty = "dotted")
```

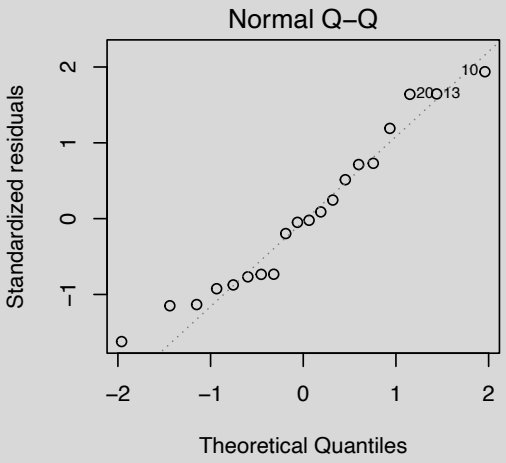

y
x
9
8
7
6
5
0
10
20
30
40

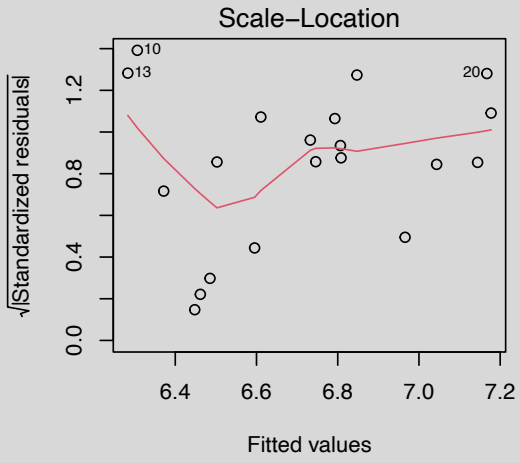
```r
summary(model4)
```

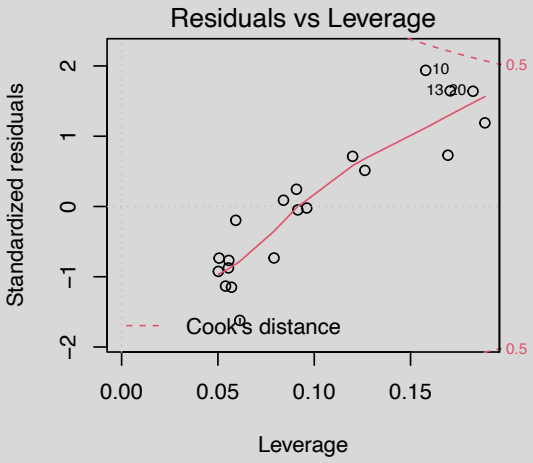
```
##
## Call:
## lm(formula = y ~ x, data = modelldiagnostik2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.32442 -1.14183 -0.04978  0.98554  2.63062
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.19597    0.66130   10.88  2.4e-09 ***
## x           -0.02306    0.02744   -0.84    0.412
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.48 on 18 degrees of freedom
## Multiple R-squared:  0.03774,Adjusted R-squared:  -0.01572
## F-statistic: 0.706 on 1 and 18 DF,  p-value: 0.4118
```

```r
par(mfrow = c(2,2))
plot(model4)
```










```
par(mfrow = c(1,1))
```

In diesem Beispiel ist die Annahme eines linearen Zusammenhangs verletzt, es könnte vielmehr ein quadratischer Zusammenhang vorliegen. Auch einen quadratischen Einfluss kann man mit dem `lm`-Befehl fitten:

```
lm(y ~ x+ I(x^2), data = <dataframe>)
```

Das Modell bleibt trotzdem *linear*, da die Koeffizienten (β_0, β_1, β_2) linear in das Modell eingehen!

Aufgabe 4

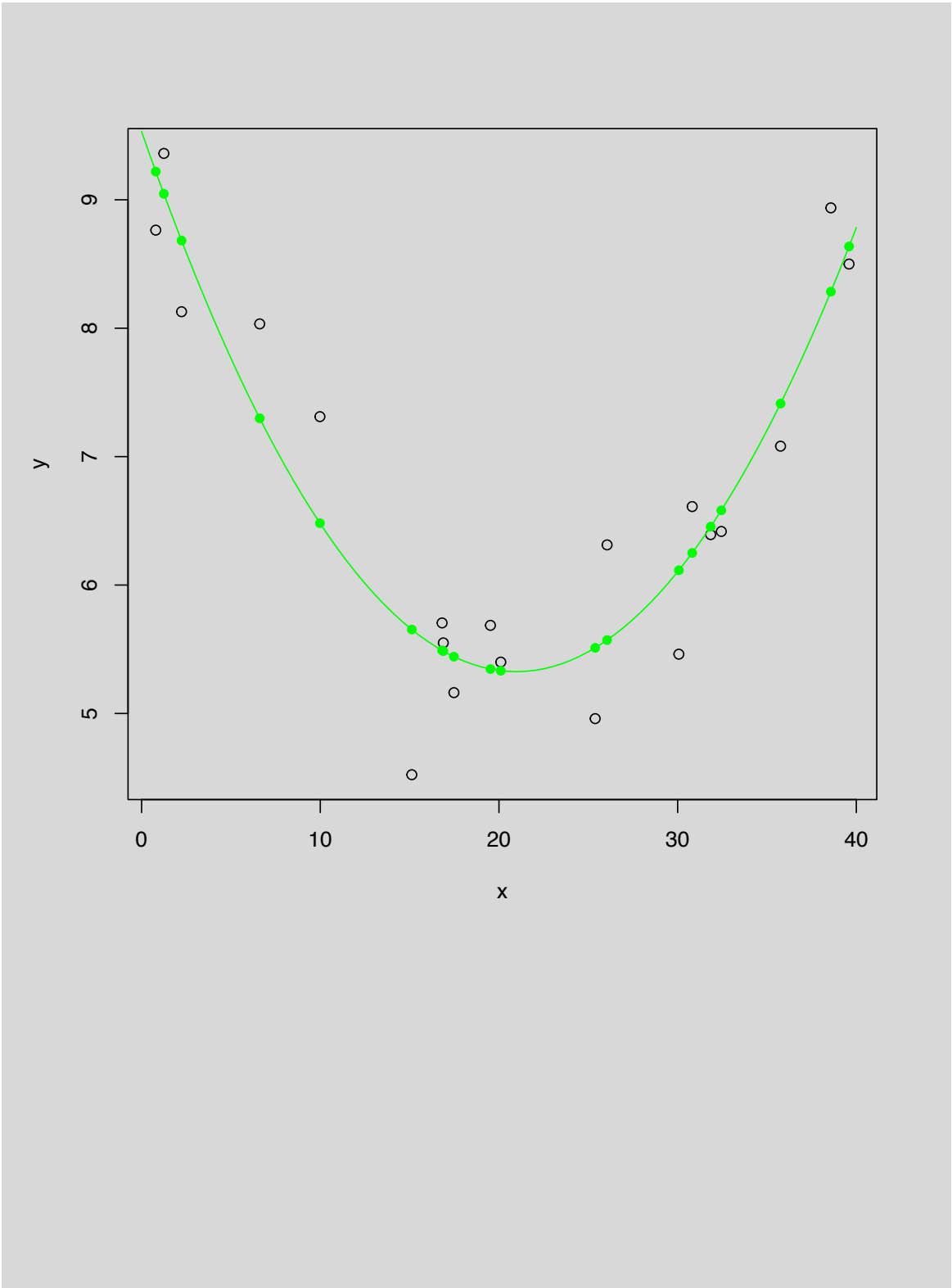
Erweitern Sie das Modell um einen quadratischen Term. Wie sehen die Diagnostikplots nun aus?

Ein multiples lineares Modell können Sie nicht mehr einfach mit `abline` darstellen. Wenn `model5` Ihr erweitertes Modell mit quadratischem Term bezeichnet, können Sie können es folgendermaßen visualisieren:

```
plot(y ~ x, <dataframe>) # Punkte darstellen
points(fitted(model5) ~ x, data = dataframe, col = "green", pch = 16)
```

Durch die Hinzunahme des quadratischen Terms verändert sich das Modell. Der Vorhersage zu den Punkten folgt tatsächlich einer quadratischen Parabel. Der Koeffizient zum quadratischen Term ist statistisch hoch signifikant. Die Residuen-Plots sehen nun in Ordnung aus, d.h. im Residuals vs Fitted Plot ist kein Trend mehr zu erkennen.

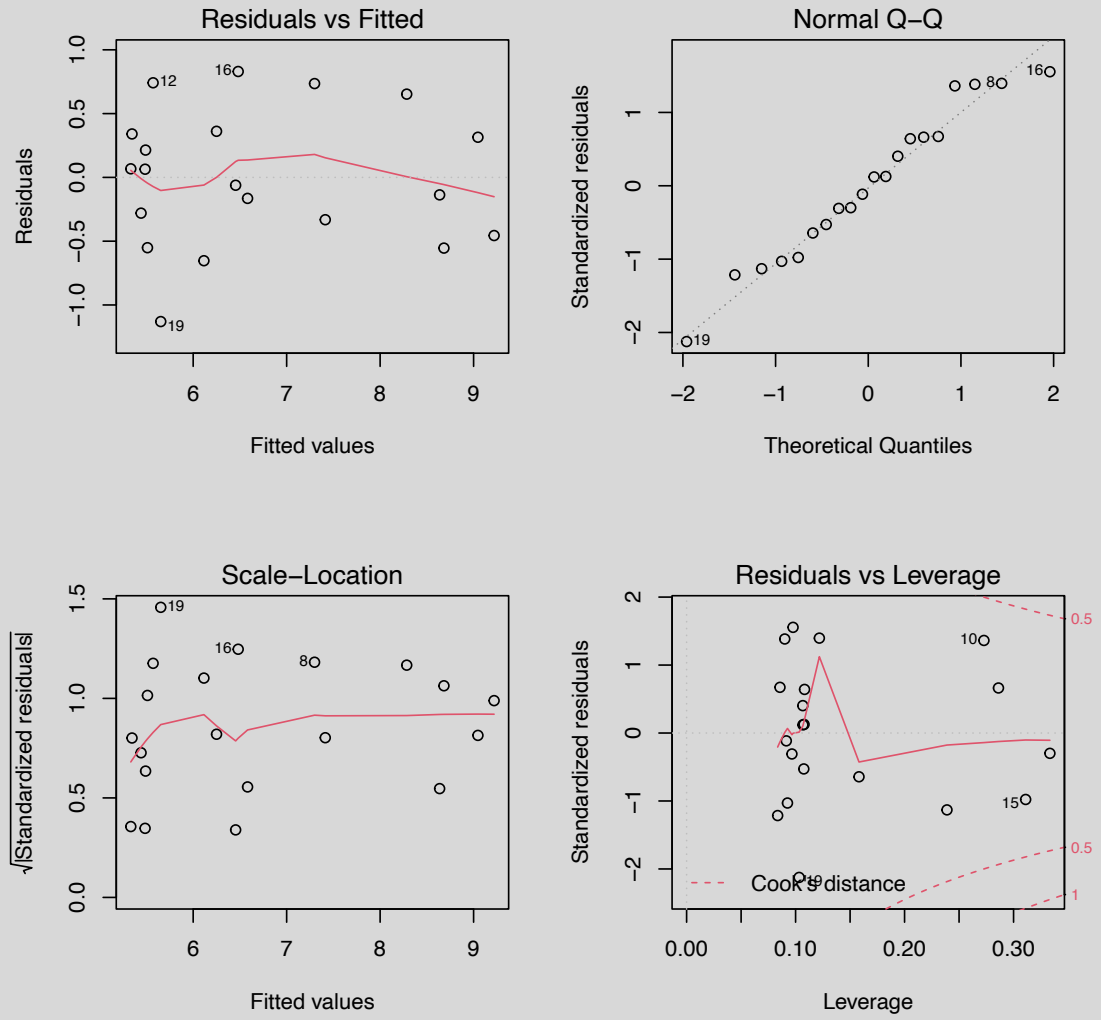
```
model5 <- lm(y ~ x + I(x^2), data = modelldiagnostik2)

plot(y ~ x, data = modelldiagnostik2)
points(fitted(model5) ~ x, data = modelldiagnostik2,
       col = "green", pch = 16)

# komplette Vorhersage
predData <- data.frame(x = seq(0, 40, by = .5))
predData$y_pred <- predict(model5, newdata = predData)
lines(y_pred ~ x, data = predData, col = "green")
```


y
x
5
6
7
8
9
0
10
20
30
40

```r
summary(model5)
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2), data = modelldiagnostik2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13030 -0.36290  0.00118  0.34557  0.82989
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.5318085  0.3371351   28.27 9.82e-16 ***
## x           -0.4010146  0.0378602  -10.59 6.62e-09 ***
## I(x^2)       0.0095576  0.0009204   10.38 8.89e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5618 on 17 degrees of freedom
## Multiple R-squared:  0.869,Adjusted R-squared:  0.8535
## F-statistic: 56.36 on 2 and 17 DF,  p-value: 3.149e-08
```

```r
par(mfrow = c(2,2))
plot(model5)
```



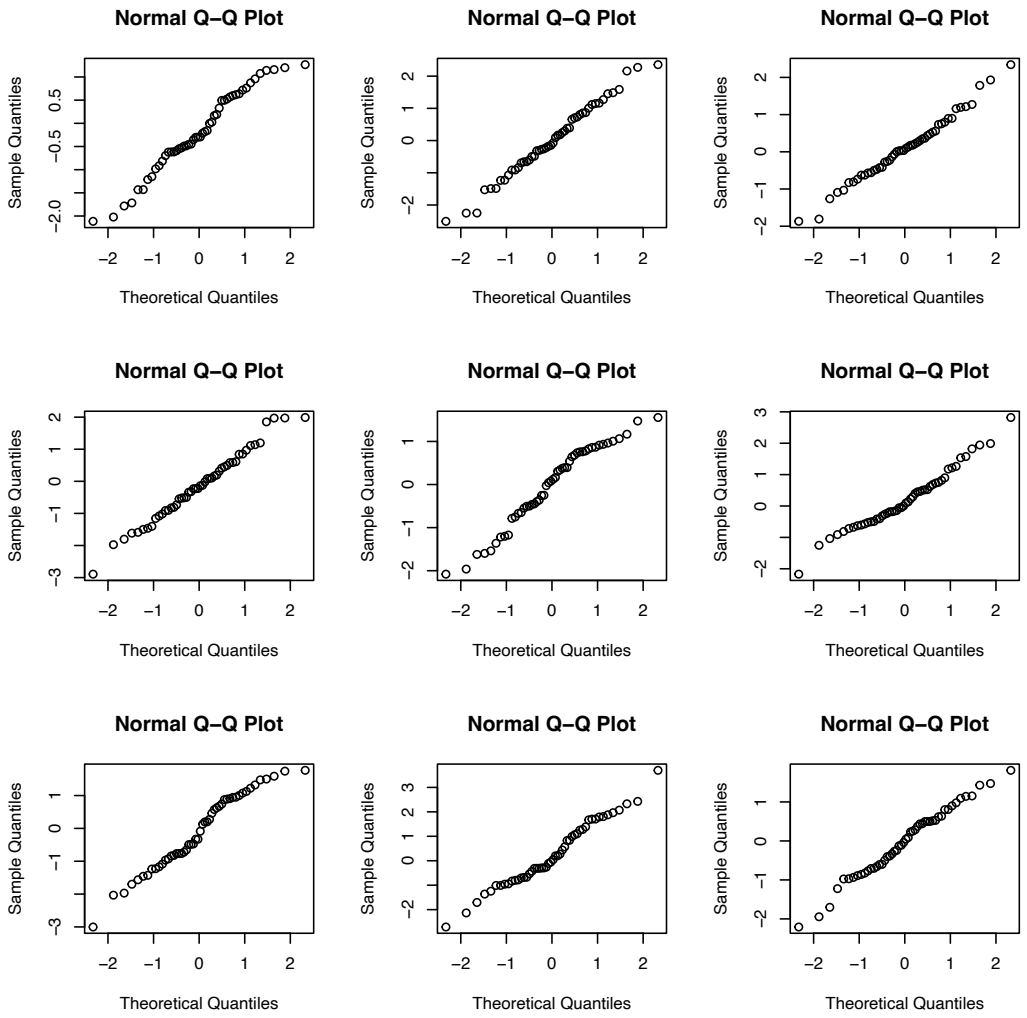

```
par(mfrow = c(1,1))
```

In diesem Beispiel war der nichtlineare Zusammenhang zwischen Y und X relativ klar zu sehen. Bei echten Datensätzen, insbesondere wenn noch mehrere Prädiktoren im Spiel sind, ist ein solcher Zusammenhang oft schwerer und weniger eindeutig zu erkennen!

Aufgabe 5

Um ein Gefühl für das Spektrum des „Normalen“ zu bekommen, erzeugen Sie ein paar Q-Q-Plots für Stichproben aus einer Normalverteilung. So könnten Q-Q-Plots aussehen, wenn die Residuen normalverteilt sind:

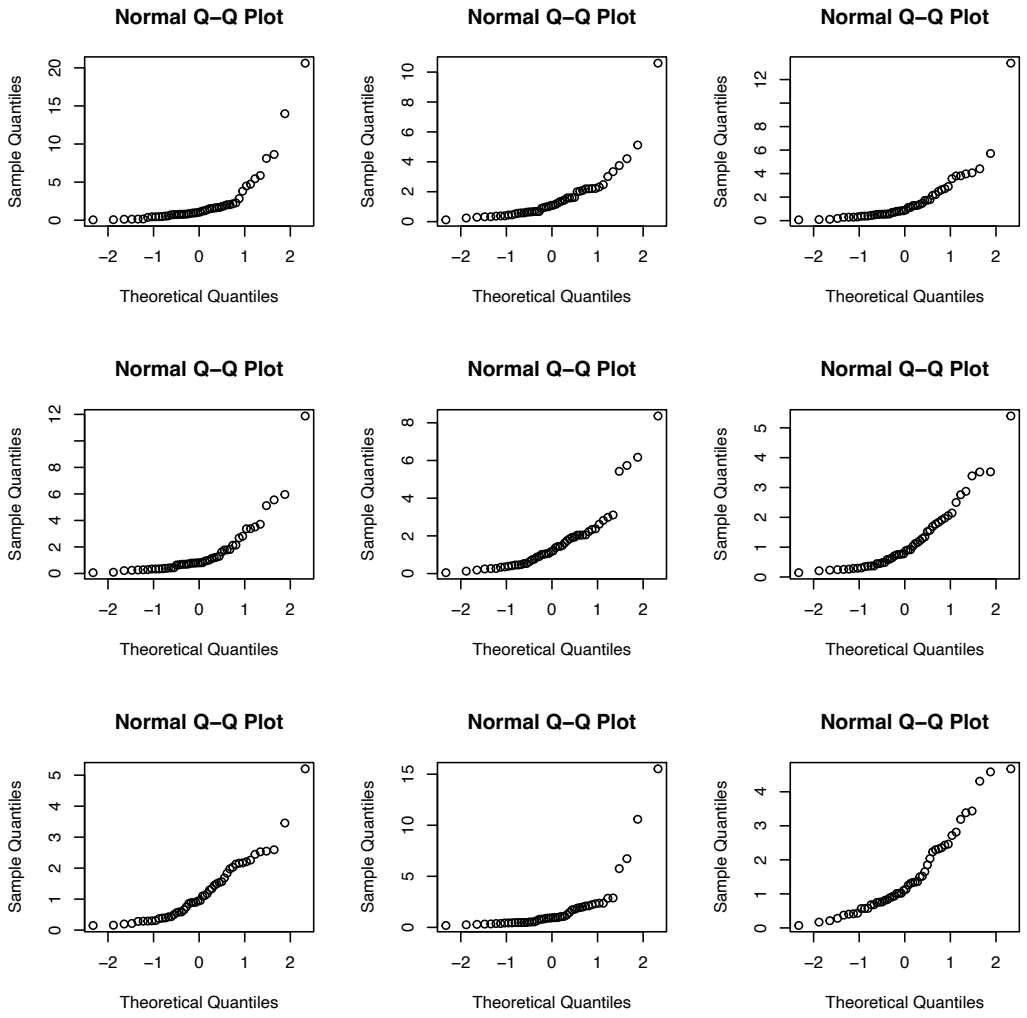
```
par(mfrow = c(3,3))                    # 3x3 Plots
for(i in 1:9) qqnorm(rnorm(50))        # normalverteilte Zufallszahlen
```



Nun drei Beispiele dafür, wie Q-Q Plots aussehen, wenn die standardisierten Residuen *nicht* normalverteilt sind:

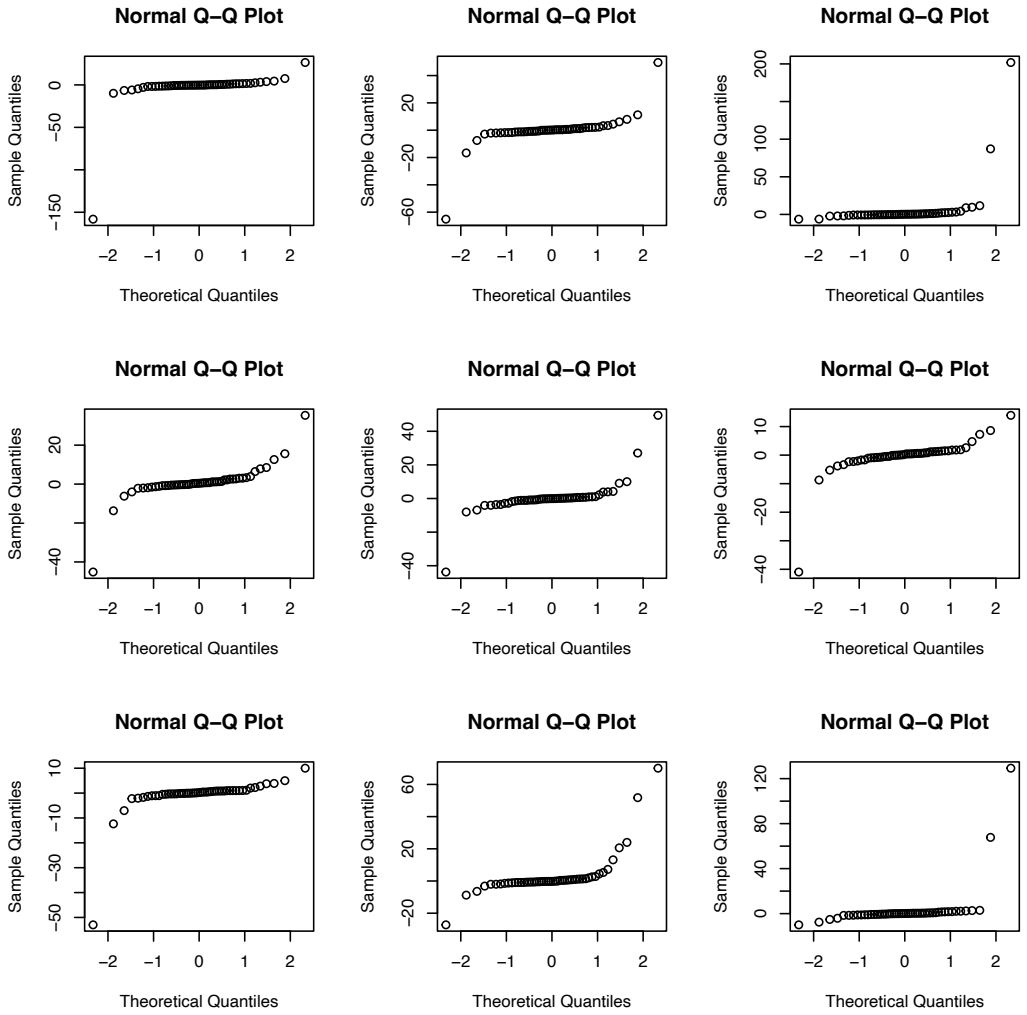
Aufgabe 5 (cont)

```r
par(mfrow = c(3,3))                    # 3x3 Plots
for(i in 1:9) qqnorm(exp(rnorm(50))) # Exponentialverteilung
```



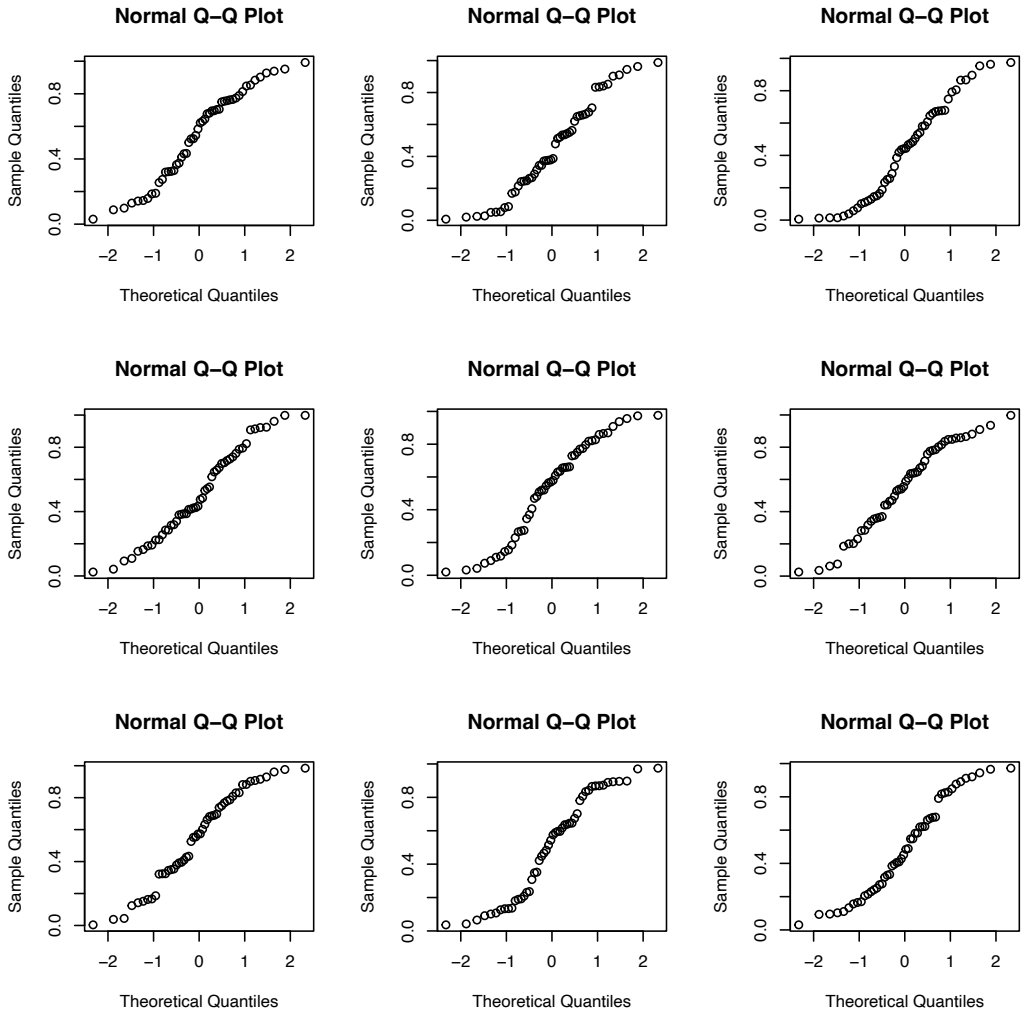

```r
for(i in 1:9) qqnorm(rcauchy(50))     # Cauchyverteilung
```

Aufgabe 5 (cont)




```
for(i in 1:9) qqnorm(runif(50))       # uniforme Verteilung
```

Aufgabe 5 (cont)




```
par(mfrow = c(1,1))                    # zurueck auf 1x1 Plot
```

Ausblick. Mithilfe der Diagnostikplots kann ein angepasstes Modell auf Adäquatheit überprüft werden.

Falls das Modell inadäquat erscheint, gibt es eine Fülle von statistischen Methoden zur Korrektur: Ansatzpunkte sind beispielsweise die Transformation von abhängiger Variable oder von Prädiktoren (Log-Transformation, Wurzeltransformation, Arcus-Sinus-

Transformation). Auch gibt es sogenannte robuste Methoden für die Parameterschätzung als Alternative zur Kleinsten-Quadrate-Schätzung (sie sind nicht so anfällig gegen Ausreißer, und gründen auf weniger Annahmen bezüglich der Fehlerverteilung - Nachteil: weniger flexibel / praktisch (Konfidenzintervalle lassen sich u.U. schwerer ausrechnen, geringere Power, etc.)).

Die Fülle dieser Methoden wird in diesem Kurs nicht abgedeckt - wichtig ist uns, dass Sie lernen, Probleme mit dem Modell durch die Modelldiagnostik zu detektieren um ggf. im konkreten Anwendungsfall dann nicht die falschen Schlüsse zu ziehen, und stattdessen nach einem besser geeigneten Modell Ausschau zu halten.

Fortgeschrittene Methoden der Biometrie

– Lineares Modell –

IMB
TU Dresden

WS 2021/22

Übungsblatt identisch zu unserem!

Inhaltsverzeichnis

4 Lineare Modelle mit mehreren Prädiktoren

Bisher haben Sie lineare Modelle angepasst, bei denen der Einfluss *eines* metrischen Prädiktors auf eine metrische Zielgröße untersucht wurde. Nun werden Sie ein Modell mit mehreren Prädiktoren untersuchen. Handelt es sich bei den Prädiktoren ausschließlich um metrische Größen, spricht man von multipler linearer Regression, bei ausschließlich kategorialen Größen von Varianzanalyse (engl. analysis of variances – ANOVA). Treten beide Typen gleichzeitig auf, so spricht man von Kovarianzanalyse (engl. analysis of covariances - ANCOVA). Die Modellanpassung mit Hilfe der `lm`-Funktion erfolgt analog zur einfachen Regression.

Zusätzlich lernen Sie heute, wie Interaktionen zwischen einzelnen Prädiktoren in ein Modell aufgenommen werden können.

4.1 Exploration und Modellanpassung – Fruitfly-Datensatz

Hintergrund. In den 80er Jahren wurde unter Biologen die These diskutiert, dass sich stark fortpflanzende männliche Fruchtfliegen kürzer leben. Um diese These zu untersuchen, betrachten wir das folgende Experiment: Eine Gruppe von 60 männlichen Fruchtfliegen wurde einzeln unter normalen Laborbedingungen gehalten. Einer zweiten ebenfalls jeweils einzeln gehaltenen Gruppe wurde jeden Tag je sieben weibliche Fruchtfliegen zugeführt. Von allen männlichen Fruchtfliegen wurden die Überlebensdauern erfasst. Ebenfalls erfasst wurden die Thoraxabmessungen aller männlichen Fruchtfliegen, da bekannt ist, dass diese einen wichtigen Einfluss auf die Lebensdauer von Fruchtfliegen hat. Außerdem sollen zwei Subtypen von Fruchtfliegen unterschieden werden.

Aufgabe 1

Laden Sie den Datensatz `fruitfly.dat` als Dataframe unter dem Namen `fruitfly`, und schauen Sie sich die Zusammenfassung an. Die Zielgröße Lebenszeit (in Tagen) ist mit `longevity` bezeichnet. Die Prädiktoren sind `thorax`, `type` und `activity` (dabei steht „yes“ für sexuell aktive und „no“ für enthaltsame Fliegen).

Visualisieren Sie den Zusammenhang zwischen der Zielgröße `longevity` und den Prädiktoren `activity`, `type` und `thorax` je nach Art des Prädiktors mithilfe von Boxplots oder Streudiagrammen. Welche Zusammenhänge können Sie in den Grafiken erkennen?

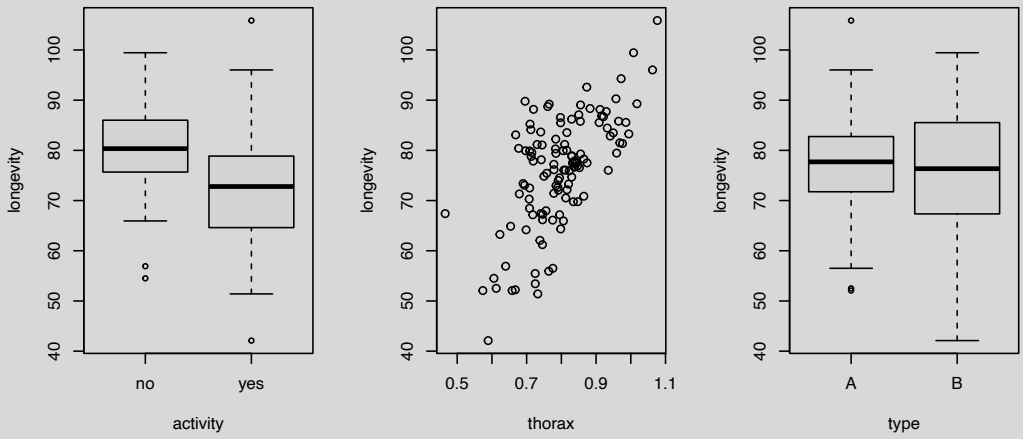
Passen Sie anschließend ein multiples lineares Modell mit allen Einflussgrößen (ohne Interaktionen) für die Zielgröße `longevity` an.

Deskription: Es gibt 120 Beobachtungen. Die Verteilung von `activity` und `type` legt nahe, dass das Experiment als balanciertes Design durchgeführt wurde.

```
fruitfly<- read.delim("data/fruitfly.dat")
summary(fruitfly)

##      thorax         longevity        activity            type
##  Min.   :0.4654   Min.   : 42.10   Length:120         Length:120
##  1st Qu.:0.7248   1st Qu.: 70.16   Class :character   Class :character
##  Median :0.7949   Median : 77.48   Mode  :character   Mode  :character
##  Mean   :0.7982   Mean   : 75.96
##  3rd Qu.:0.8506   3rd Qu.: 83.50
##  Max.   :1.0764   Max.   :105.87

xtabs(~type + activity, data = fruitfly)

##     activity
## type no yes
##    A 30  30
##    B 30  30
```

Deskriptive Plots:

```
par(mfrow=c(1,3))
boxplot(longevity ~ activity, fruitfly)
plot(longevity ~ thorax, fruitfly)
boxplot(longevity ~ type,fruitfly)
```



Die Plots lassen sich folgendermaßen interpretieren:

- Die Überlebensdauer sinkt bei sexueller Aktivität.
- Je größer der Thorax, desto größer die Überlebensdauer.
- Der Subtyp scheint keinen Einfluss auf die Überlebensdauer zu haben.

Multiples (additives) lineares Modell: Die Funktion `lm` passt das lineare Modell an. Es ist ein ANCOVA-Modell, weil es neben Faktoren (kategorialen Einflussgrößen) auch einen metrischen Prädiktor gibt. `summary` gibt folgende Zusammenfassung des Modells:

```
# multiples lineares Modell
mod_fruitfly1 <- lm(longevity ~ activity + thorax + type,fruitfly)
summary(mod_fruitfly1)

##
## Call:
## lm(formula = longevity ~ activity + thorax + type, data = fruitfly)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1110  -5.1165   0.6272   4.5039  16.6682
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  23.3681     5.1018   4.580 1.18e-05 ***
## activityyes  -9.3216     1.3008  -7.166 7.69e-11 ***
## thorax       71.8871     6.2154  11.566  < 2e-16 ***
## typeB        -0.2519     1.3036  -0.193    0.847
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.105 on 116 degrees of freedom
## Multiple R-squared:  0.6019,Adjusted R-squared:  0.5916
## F-statistic: 58.46 on 3 and 116 DF,  p-value: < 2.2e-16
```

Interpretation: Das vorliegende additive lineare Modell beschreibt den Zusammenhang zwischen `thorax` und `longevity` bei männlichen Fliegen durch eine Gerade mit Anstieg 71.89. Wo die Gerade verankert ist, hängt gemäß Modell allerdings von `activity` und `type` ab. Diese beiden Faktoren sorgen jeweils für eine Parallelverschiebung der Geraden, so dass das Modell insgesamt 2x2 = 4 Geraden beschreibt.

Für die Bestimmung der Verankerung der Geraden ist wichtig, welche Gruppe die Referenzgruppe ist. Bei R ist standardmäßig die in alphabetischer Reihenfolge erste Kategorie die Referenenzgruppe, in unserem Beispiel sind das für `activity` die sexuell nichtaktiven Fliegen („no“ vor „yes“) und bei `type` der Typ A. Der geschätzte Intercept 23.37 entspricht dem Wert der Zielgröße `longevity` bei einer Thoraxgröße von 0 in der Referenzgruppe der enthaltsamen Fliegen vom Referenztyp A. Der Koeffizient für `activity` beschreibt die parallele Verschiebung der Geraden, wenn man auf die aktiven Fliegen eines Typs im Vergleich zu den inaktiven Fliegen gleichen Typs schaut. Konkret schätzt das Modell für sexuell aktive Fliegen eines fixen Typs mit einer bestimmten, fixen Thoraxgröße ein um 9.32 kürzeres Leben als enthaltsame Fliegen gleichen Typs und mit gleicher Thoraxgröße. Entsprechend erwartet das Modell, dass Fliegen vom Typ B im Mittel 0.25 kürzer leben als vergleichbare Fliegen (d.h. gleiche Aktivität und gleiche Thoraxgröße) vom Typ A.

Modelldiagnostik. Analog zur einfachen Regression kann mithilfe des `plot`-Befehls anhand der vier Diagnostikplots überprüft werden, ob die Voraussetzungen zur Anwendung des Modells erfüllt sind.

Zusätzlich zu diesen Diagrammen gibt es für multiple Regressionsmodelle weitere nützliche Diagramme, zum Beispiel die sogenannten **Partial Residual Plots**: Damit lässt sich für jeden einzelnen metrischen Prädiktor die Annahme eines linearen Einflusses überprüfen.

Im jeweiligen Partial Residual Plot für einen der Prädiktoren wird der geschätzte Einfluss aller anderen Prädiktoren herausgerechnet. Damit lässt sich erkennen, ob der übrigbleibende Zusammenhang zwischen dem zu untersuchenden Prädiktor und der abhängigen Variable mit der angenommenen Linearität vereinbar ist. Ein linearer Zusammenhang drückt sich durch eine gleichmäßige Streuung der Datenpunkte über und unter der Geraden über den gesamten Wertebereich des Prädiktors aus.

Die Funktion `prplot` aus dem Paket `faraway` fertigt solche Diagramme an.[1] Um das Paket zu aktivieren, klicken Sie es in RStudio entweder im Reiter „Packages“ an oder führen Sie direkt in R den Befehl `library(faraway)` im Eingabefenster aus. Nun steht Ihnen die Funktion `prplot` zur Verfügung.

Aufgabe 2

Führen Sie eine grafische Modelldiagnostik des angepassten Modells durch. Überprüfen Sie zunächst mithilfe der vier Standardplots, ob es Einwände gegen die Anwendung des Modells gibt.

Schauen Sie sich dann die Partial Residual Plots für jeden der Einflussfaktoren an, für den ersten Prädiktor beispielsweise durch:

```
prplot(<model>,1)
```

Spricht etwas gegen die Linearitätsannahme?

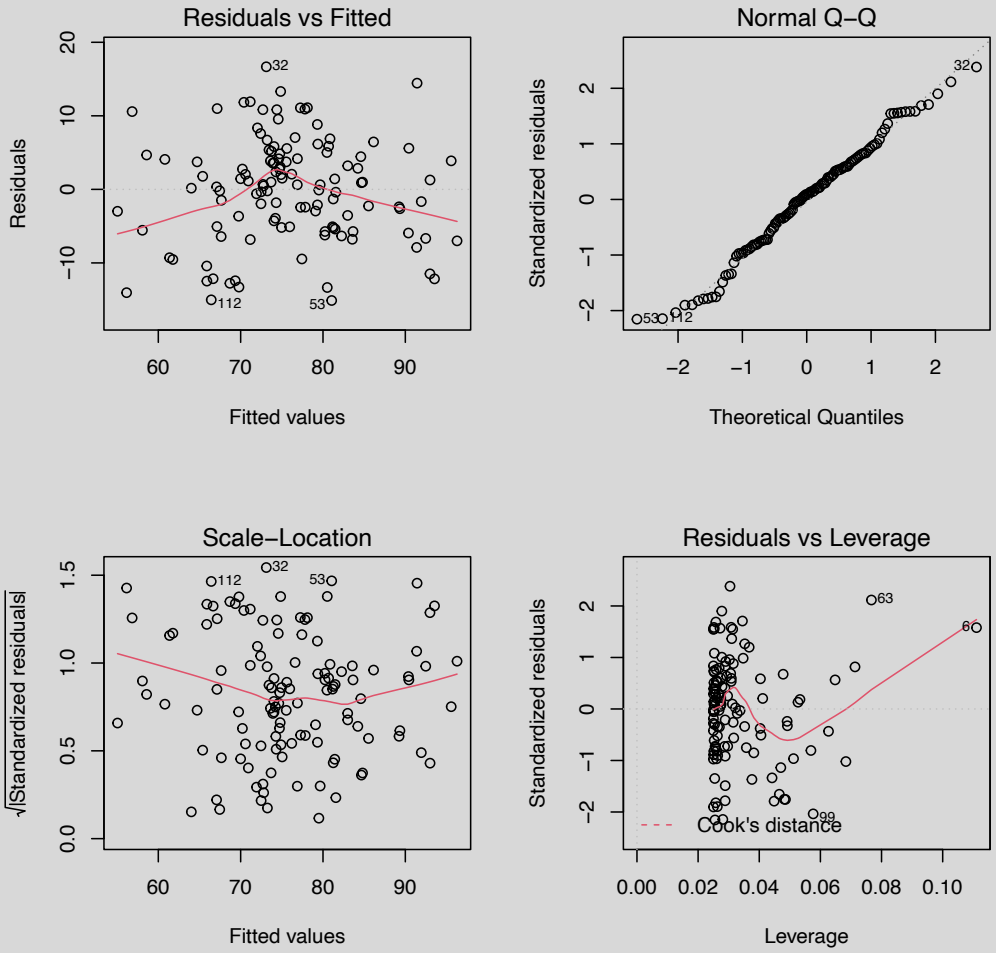
Es zeigen sich keine groben Verletzungen der Modellannahmen:

Diagnostik-Plots zeigen keine gravierende Verletzung der Modellannahmen. Die Residuen vs Fitted-Plots sind in Ordnung, obwohl an den Rändern (d.h. kurze und lange vorhergesagte `longevity`) das Modell dazu neigt, die beobachtete `longevity` zu überschätzen (negative Residuen). Im Scale-Location Plot drückt sich dies auch durch eine erhöhte Varianz aus. Die Verteilung der Residuen widerspricht der

[1] Die Installation des Pakets mittels des `install.packages`-Befehls ist nur vor der erstmaligen Nutzung notwendig.

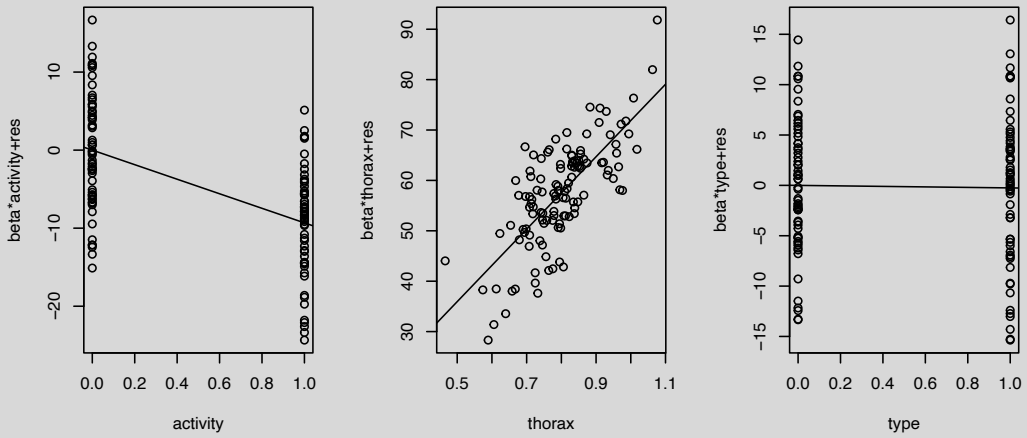
Normalverteilungs-Annahme nicht. Die Fliege mit dem kleinsten Thorax (Beobachtung Nr. 6) hat einen vergleichsweise starken Einfluss auf den Modell-Fit.

```
par(mfrow=c(2,2)) # Graphikfenster mit 2x2 Diagrammen
plot(mod_fruitfly1) # Ausgabe der vier Diagnostikplots
```



Partial Residual Plots: Die Residuen verteilen sich jeweils gleichmäßig auf beiden Seiten der Geraden. Bei nur zwei Ausprägungen eines Prädiktors ist der Begriff Linearität natürlich nicht wirklich sinnvoll. Allerdings kann man beurteilen, ob die Streuung der Beobachtungen zu den verschiedenen Ausprägungen gleich erscheint.

```
library(faraway) # Laden des Pakets
par(mfrow=c(1,3))
for(i in 1:3) prplot(mod_fruitfly1,i)
```



4.2 Modellauswahl

Oft hat man eine ganze Reihe möglicher Prädiktoren, deren Einfluss auf eine Zielgröße untersucht werden soll. Prädiktoren können die Zielgröße dabei nicht nur additiv beeinflussen, sondern untereinander interagieren oder auch nichtlinear, beispielsweise quadratisch, in ein Modell eingehen. Um aus dieser Vielzahl möglicher Prädiktoren die relevanten Einflüsse herauszufiltern, ist eine Beschränkung des Modells notwendig.

Es gibt dabei allerdings kein allgemeingültiges Verfahren, um ein Modell auszuwählen. Modelle können beispielsweise durch schrittweise erfolgende Hinzunahme von Prädiktoren (*forward selection*) oder aus dem vollen Model durch Entfernen nicht signifikanter Prädiktoren (*backward selection*) aufgebaut werden. Statt dieser p-Wert-basierten Verfahren gibt es auch andere Kriterien, beispielsweise *Akaikes Informationskriterium*, welches in R mithilfe der Funktion `step()` automatisiert angewendet werden kann. Auch komplexere Strategien (*purposeful selection*) zum Modellaufbau sind möglich, bei denen zunächst Prädiktoren additiv ins Modell aufgenommen werden und später mögliche Interaktionseffekte hinzugenommen werden.

Je nach verwendetem Verfahren können am Ende unterschiedliche Resultate stehen. Auch Vorwissen darf bei der Modellauswahl berücksichtigt werden. Der Weg zur Modellauswahl sollte daher immer offen und nachvollziehbar dargestellt werden.

Wir wollen im Folgenden den Backward-Selection-Algorithmus auf dem obigen Modell anwenden:

1. Starte mit allen Prädiktoren im Modell (Maximales Modell)
2. Entferne den Prädiktor mit dem größten P-Wert größer als α_{crit}
3. Passe das kleinere Modell an und gehe zu Schritt 2
4. Stoppe, wenn alle p-Werte kleiner als α_{crit} sind

Dabei wird oft ein α_{crit} von 0.05 gewählt.

Den jeweiligen Prädiktor mit dem höchsten P-Wert können Sie mithilfe des Befehls `drop1` ermitteln:

```
drop1(<model>,test="F")
# für <model> den Namen des gespeicherten Modells einsetzen
```

Um eine Variable aus einem Modell zu entfernen, können Sie den `update`-Befehl verwenden. Damit lässt sich das Modell ohne viel Tipparbeit anpassen:

```
model2 <- update(<model>, . ~ . - <zu entfernende Variable>)
```

In der `update`-Formel steht ein Punkt . für den ursprünglichen Ausdruck an dieser Stelle in der Formel. Mit `-` können Sie Ausdrücke aus der Formel entfernen.

Aufgabe 3

Vereinfachen Sie das volle Modell mittels des *Backward-Selection*-Algorithmus. Welche Effekte bleiben übrig?

Mithilfe der `drop1`-Funktion wird die Variable `type` als nicht signifikant erkannt. Mit `update` wird diese Variable aus dem Modell entfernt. Die beiden anderen Variablen `activity` und `thorax` bleiben im finalen Modell signifikant.

```r
drop1(mod_fruitfly1, test="F")
```

```
## Single term deletions
##
## Model:
## longevity ~ activity + thorax + type
##          Df Sum of Sq     RSS    AIC  F value    Pr(>F)
## <none>                 5855.1 474.51
## activity  1    2591.8  8446.9 516.49  51.3484 7.689e-11 ***
## thorax    1    6752.2 12607.2 564.54 133.7735 < 2.2e-16 ***
## type      1       1.9  5856.9 472.55   0.0373    0.8471
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod_fruitfly2 <- update(mod_fruitfly1, . ~ . -type)
# verbleibende Praediktoren erneut beurteilen!
drop1(mod_fruitfly2, test="F")
```

```
## Single term deletions
##
## Model:
## longevity ~ activity + thorax
##          Df Sum of Sq     RSS    AIC F value    Pr(>F)
## <none>                 5856.9 472.55
## activity  1    2593.0  8449.9 514.53  51.798 6.359e-11 ***
## thorax    1    6842.4 12699.3 563.42 136.685 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.3 Interaktionsmodell

Bisher wurde angenommen, dass sich die Effekte der einzelnen Prädiktoren auf die Zielgröße einfach addieren. Es kann allerdings auch sein, dass sich die Prädiktoren gegenseitig in ihrem Effekt beeinflussen. Man spricht dann von *Interaktionseffekten*.

Die Anpassung eines linearen Modells mit Interaktionen in R ist sehr einfach. Entweder Sie fügen die Interaktion mit dem Doppelpunkt-Operator separat hinzu, oder Sie verwenden statt des Pluszeichens ein Malzeichen. Also, gleichbedeutend sind folgende

Aufrufe:

```
mod_fruitfly3 <- lm(longevity ~ thorax + activity + thorax:activity,
                    data = fruitfly)
## oder einfacher:
mod_fruitfly3 <- lm(longevity ~ thorax * activity, data = fruitfly)
```

Aufgabe 4

Interpretieren Sie die einzelnen Koeffizienten des Interaktionsmodells. Ist der Interaktionseffekt statistisch signifikant?

Führen Sie nun erneut eine grafische Modelldiagnostik für das Interaktionsmodell durch. Spricht etwas gegen das angepasste Modell?

Wir schauen uns das Interaktions-Modell genauer an:

```
##
## Call:
## lm(formula = longevity ~ thorax * activity, data = fruitfly)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.0874  -4.4162   0.5808   4.0780  14.0723
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          43.841      6.339   6.916 2.70e-10 ***
## thorax               45.821      7.951   5.763 6.93e-08 ***
## activityyes         -51.855      9.132  -5.679 1.02e-07 ***
## thorax:activityyes   53.277     11.341   4.698 7.28e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.513 on 116 degrees of freedom
## Multiple R-squared:  0.6654,Adjusted R-squared:  0.6568
## F-statistic:  76.9 on 3 and 116 DF,  p-value: < 2.2e-16
```

Interpretation Koeffizienten Eine Interaktion zwischen den Faktoren `thorax` und `activity` bedeutet, dass die Geraden `longevity` vs `thorax` für die beiden `activity` Gruppen *nicht* parallel verlaufen. D.h. der Zuwachs der Lebenszeit bei zunehmender Größe des Thorax erfolgt dann in beiden Gruppen (aktiv oder nicht aktiv) mit je unterschiedlichem Anstieg. Dieser *Unterschied im Anstieg* wird durch den Interaktionsterm geschätzt.

Interpretation der Koeffizienten: Der „Intercept“, also der Schnittpunkt mit der y-Achse, beschreibt die mittlere Lebenszeit in der Referenzgruppe bei einer theoretischen Thoraxgröße von 0. Der Koeffizient `activityyes` beschreibt den Unterschied zwischen aktiven und nichtaktiven Fliegen bei dieser Thoraxgröße 0. Der Anstieg der Geraden für die Referenzgruppe („nicht aktiv“) wird auf 45.82 (Koeffizient `thorax`) geschätzt. Konkret, ist der Thorax um 0.1 größer, dann nimmt die Lebenszeit im Mittel in der nicht-aktiven Gruppe um 4.58 zu. Der Interaktionsterm (Koeffizient `thorax:activityyes`), also der Unterschied zwischen den Anstiegen in beiden Gruppen, wird auf 53.28 geschätzt. Der Anstieg der Geraden für die *aktiven* Fliegen ergibt daher 99.1 (Koeffizient `thorax` + `thorax:activityyes`). Sie profitieren also stärker von ihrer Thoraxgröße als nicht-aktive Fliegen.

Kleine aktive Fliegen sterben nach dem Modell im Mittel früher als kleine nichtaktive Fliegen. Der Unterschied zwischen beiden Gruppen verringert sich aber bei zunehmender Thoraxgröße, bis bei großem Thorax schließlich aktive Fliegen laut Modell im Mittel länger leben.

Diese Zusammenhänge werden auch in der graphischen Darstellung in der letzten Aufgabe deutlich.

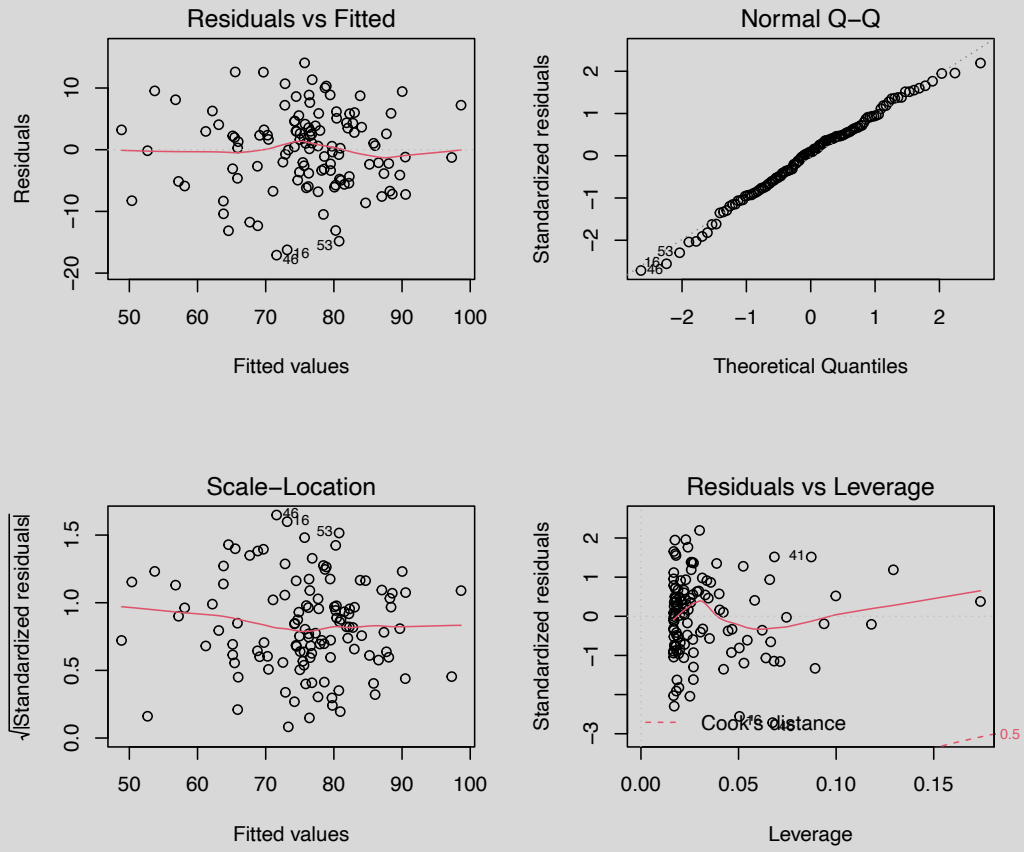
Signifikanz des Interaktionseffekts: Der Interaktionsterm ist statistisch signifikant (auf dem 5%-Niveau). Das bedeutet, dass wir davon ausgehen können, dass sich der Zusammenhang zwischen Thoraxgröße und Lebenszeit zwischen den Gruppen (aktiv vs. nicht aktiv) unterscheidet. Beide Tests, der t-Test im `summary` und der F-Test aus `drop1` stimmen hier exakt überein.

```r
drop1(mod_fruitfly3, test="F")
```

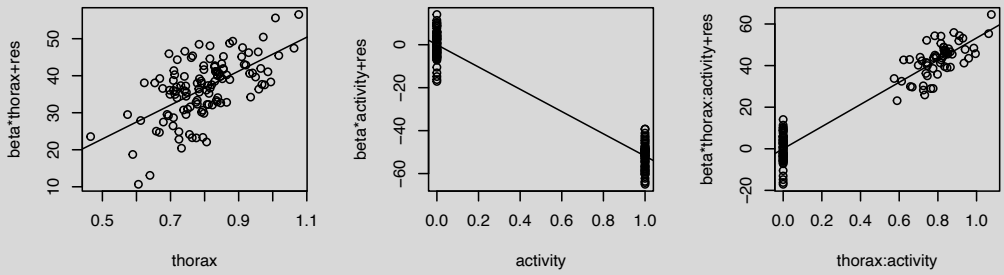
```
## Single term deletions
##
## Model:
## longevity ~ thorax * activity
##                 Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>                       4920.8 453.65
## thorax:activity  1    936.18 5856.9 472.55  22.069 7.283e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Modell-Diagnostik:

```r
par(mfrow = c(2,2))
plot(mod_fruitfly3)
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q–Q
Standardized residuals
Theoretical Quantiles
Scale–Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Cook's distance
Leverage

```
par(mfrow = c(1,3))
for(i in 1:3) prplot(mod_fruitfly3,i)
```



Auf Grundlage der Diagnostikplots spricht nichts gegen das angepasste Modell.

Anteil der vom Model erklärten Varianz. Das **Bestimmtheitsmaß** R^2 ist ein statistisches Maß für den durch das Modell erklärten Anteil an der Varianz der Zielgröße. Es wird folgendermaßen berechnet:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}},$$

wobei RSS der Summe der quadrierten Residuen (engl. *residual sum of squares*) und TSS der Summe der quadrierten Abstände vom Mittelwert (d.h. der Variabilität, engl. *total sum of squares*) der Zielgröße entspricht. R^2 liegt zwischen 0 und 1. Ist $R^2 = 1$ (d.h. $\text{RSS} = 0$), dann kann die Zielgröße vollständig durch die Prädiktoren erklärt werden. Ist R^2 nahe 0 (d.h. RSS in der Größenordnung von TSS), dann kann das gewählte Modell kaum zur Erklärung der Variabilität der Zielgröße beitragen.

Das Bestimmtheitsmaß R^2 steigt, wenn zusätzliche Prädiktoren in das Modell aufgenommen werden. Diese Abhängigkeit von der Anzahl der Prädiktoren kann durch die Verwendung des **adjustierten Bestimmtheitsmaßes** umgangen werden. Beide Maße sind in der Zusammenfassung eines Modells mithilfe des `summary`-Befehls für `lm`-Objekte angegeben.

Aufgabe 5

Wie hoch ist der Anteil der erklärten Varianz im angepassten Interaktionsmodell?

Der Anteil der durch das Modell erklärten Varianz beträgt 0.665, also rund 67%. Dies kann bequem der Modell-Zusammenfassung entnommen werden.

```
summary(mod_fruitfly3)
```

```
##
## Call:
## lm(formula = longevity ~ thorax * activity, data = fruitfly)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.0874  -4.4162   0.5808   4.0780  14.0723
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)           43.841      6.339   6.916 2.70e-10 ***
## thorax                45.821      7.951   5.763 6.93e-08 ***
## activityyes          -51.855      9.132  -5.679 1.02e-07 ***
## thorax:activityyes    53.277     11.341   4.698 7.28e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.513 on 116 degrees of freedom
## Multiple R-squared:  0.6654,Adjusted R-squared:  0.6568
## F-statistic:  76.9 on 3 and 116 DF,  p-value: < 2.2e-16
```

4.4 Visualisierung des angepassten Modells.

Wir wollen nun die Vorhersagen des angepassten Interaktionsmodells grafisch veranschaulichen. Dafür soll in einem Streudiagramm der metrische Prädiktor `thorax` gegen die Zielgröße `longevity` auftragen werden. Der kategoriale Prädiktor `activity` soll farblich dargestellt werden.

Im `plot`-Befehl kann mit dem Parameter `col=` die Farbe und mit `pch=` das Zeichen (*plot character*) pro Beobachtung festgelegt werden. Wir benutzten `ifelse`, um für jeden Punkt festzulegen, ob wir rot (=aktiv) oder blau (=inaktiv) benutzen und ob Dreieck (=aktiv) oder Quadrat (=inaktiv).[2]

```
plot(longevity ~ thorax, data = fruitfly, main = "Fruitflies",
     col = ifelse(fruitfly$activity == 'yes', 'red', 'blue'),
     pch = ifelse(fruitfly$activity == 'yes', 17, 15)
     )
```

[2] Für die Kodierung der Zeichen siehe auch die Hilfe zum R-Befehl `points`.

Aufgabe 6

Zeichnen Sie die geschätzten Geraden für beide Gruppen in das Diagramm ein. Verwenden Sie dafür die Funktion `abline()`. Die Parameter Ihres Modells können Sie mithilfe der `coef()`-Funktion extrahieren. Wie ergeben sich daraus die Anstiege und Schnittpunkte mit der y-Achse für die beiden gesuchten Geraden?

Mithilfe der `coef()`-Funktion werden die geschätzten Koeffizienten (Anstiege und Schnittpunkte mit der y-Achse) aus dem Modell extrahiert und unter dem Namen `betas` gespeichert:

```
betas <- coef(mod_fruitfly3)
betas
```

```
##        (Intercept)             thorax       activityyes thorax:activityyes
##           43.84103           45.82093          -51.85536           53.27723
```

Abhängig davon, in welcher Reihenfolge die Prädiktoren in das Modell aufgenommen wurden, wählen wir nun die entsprechenden Koeffizienten für die beiden Geraden und zeichnen diese mithilfe der Funktion `abline()`. Für die nichtaktiven Fliegen ergibt sich:

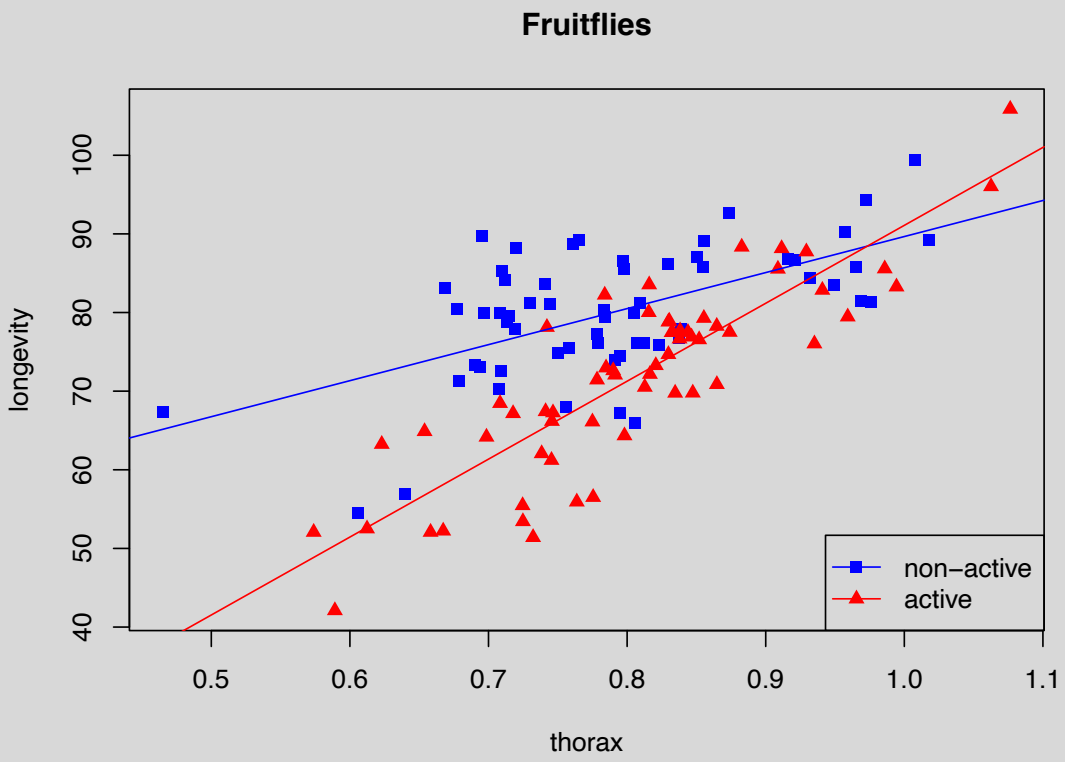
```
abline(a = betas[1] , b = betas[2] , col = 'blue' )
```

Für die aktiven Fliegen ergeben sich Anstieg und Schnittpunkt mit der y-Achse aus der Summe der jeweiligen Koeffizienten:

```
abline(a = betas[1] + betas[3] , b = betas[2] + betas[4] ,col='red')
```

Zur Verdeutlichung fügen wir noch eine Legende hinzu:

```
legend("bottomright", legend = c("non-active","active"),
  col=c('blue','red'), pch = c(15, 17),
  lwd=1)
```



Je größer der Thorax der männlichen Fruchtfliegen ist, desto länger leben diese im Mittel (positiver Koeffizient für `thorax` und auch positiver Interaktions-Koeffizient). Dieser Zusammenhang ist ausgeprägter bei sexuell aktiven Fliegen (positiver Interaktions-Koeffizient). Sexuell aktive Männchen mit kleinem Thorax (≤ 1) haben im Mittel eine geringere Lebenzeit im Vergleich mit enthaltsamen Männchen (negativer Koeffizient für `activity`).

Fortgeschrittene Methoden der Biometrie

– Logistische Regression –

IMB
TU Dresden

WS 2021/22

Übungsblatt identisch zu unserem!

1 Modellanpassung bei logistischer Regression

1.1 Datenexploration – die Burn-Studie

Im Datensatz `burn.dat` finden Sie Informationen zu 1000 Patienten, die im Zeitraum zwischen 2000 und 2007 an Brandverletzungen in verschiedenen Brandzentren behandelt wurden. Ziel der Studie ist es, Risikofaktoren für das Versterben an der Brandverletzung zu identifizieren und ein Vorhersagemodell zu entwickeln.

Variable	**Beschreibung**
ID	Patienten-ID
Facility	Brandzentrums-ID
Death	Tod/Überleben
Age	Alter bei Aufnahme
Gender	Geschlecht
TBSA	Verbrannte Hautfläche (in %)
Inhalation	Inhalationstrauma?
Flame	Direkte Verbrennung?

Aufgabe 1

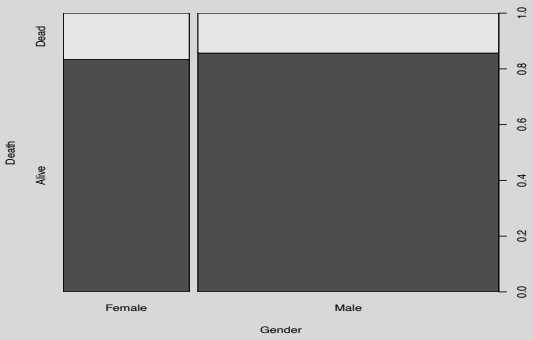
Verschaffen Sie sich zunächst einen ersten Überblick über den Datensatz. Benutzen Sie dazu auch graphische Methoden. Tipp: `summary`, `plot` und `cdplot`.

`Summary` liefert einen ersten Überblick über die Daten. Mit `cdplot` lässt sich die Abhängigkeit der Zielgröße `death` von metrischen Prädiktoren, mit `plot` der Zusammenhang mit kategorialen Größen als Mosaikplot darstellen.

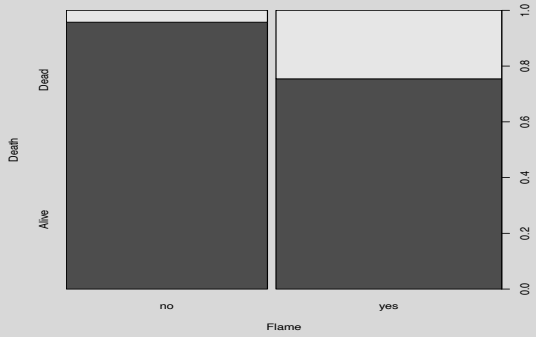
```r
# Exploration
summary(burn)
```

```
##        ID            Facility       Death          Age            Gender
##  Min.   :   1.0   Min.   : 1.00   Alive:850   Min.   : 0.10   Female:295
##  1st Qu.: 250.8   1st Qu.: 2.00   Dead :150   1st Qu.:10.85   Male  :705
##  Median : 500.5   Median : 8.00               Median :31.95
##  Mean   : 500.5   Mean   :11.56               Mean   :33.29
##  3rd Qu.: 750.2   3rd Qu.:18.25               3rd Qu.:51.23
##  Max.   :1000.0   Max.   :40.00               Max.   :89.70
##       TBSA       Inhalation Flame            Age_Cat         TBSA_Cat
##  Min.   : 0.10   no :878    no :471   [0.1,10.9] :250   [0.1,2.5]:262
##  1st Qu.: 2.50   yes:122    yes:529   (10.9,31.9]:250   (2.5,6]  :259
##  Median : 6.00                        (31.9,51.2]:250   (6,16]   :239
##  Mean   :13.54                        (51.2,89.7]:250   (16,98]  :240
##  3rd Qu.:16.00
##  Max.   :98.00
##      TBSA2        Child
##  Min.   :0.3162   no :757
##  1st Qu.:1.5811   yes:243
##  Median :2.4495
##  Mean   :3.0611
##  3rd Qu.:4.0000
##  Max.   :9.8995
```

```r
plot(Death~Gender, data = burn)
```

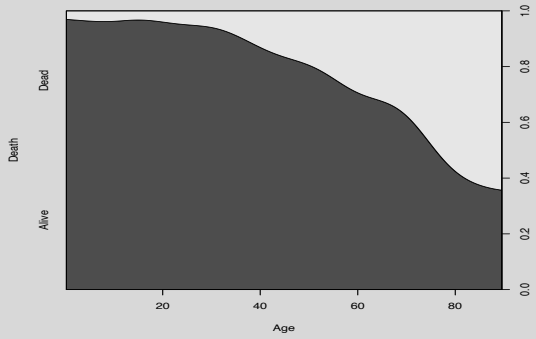


```
plot(Death~Flame, data = burn)
```

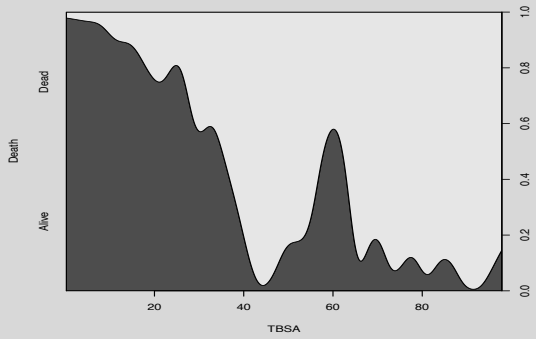



```
plot(Death~Inhalation, data = burn)
```

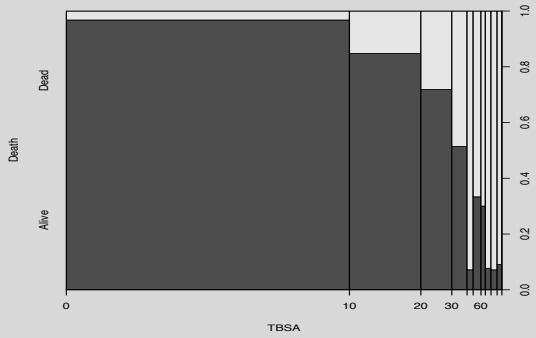

```
cdplot(Death~Age, data = burn)
```



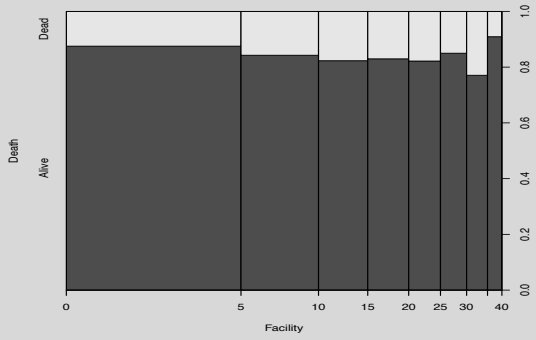

```
cdplot(Death~TBSA, data = burn)
```



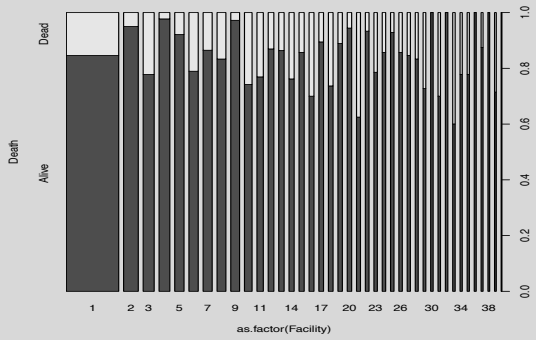

```
# Um sichtbar zu machen,
# wieviele Patienten in welchen Bereichen der Groesse TBSA liegen:
plot(Death~TBSA, data = burn)
```




```
# Facility (also das Krankenhaus) ist als metrische Groesse
# kodiert und wird bei Ausruf des plot-Befehls auch so verwendet:
plot(Death~Facility, data = burn)
```



```r
# mit as.factor() kann sie als
# kategoriale Groesse verwendet werden:
plot(Death~as.factor(Facility), data = burn)
```



1.2 einfache logistische Regression mit kategorialem Prädiktor

Zunächst möchten wir unabhängig von den anderen Größen nur den Einfluss des Risikofaktors "Flame" (direkte oder indirekte Verbrennung) auf die Versterbenswahrscheinlichkeit betrachten.

Die logistische Regression gehört zur Familie der generalisierten linearen Modelle (engl. *generalized linear models*). Für diese steht in `R` die Funktion `glm()` zur Verfügung, welche ganz ähnlich wie die Ihnen bekannte `lm()`-Funktion verwendet wird. Zur Auswahl der logistischen Regression wird lediglich zusätzlich die Option `family = binomial` angegeben.

Aufgabe 2

Passen Sie mithilfe der `glm()`-Funktion eine logistische Regression an.

Verschaffen Sie sich mit `summary()` einen Überblick über die geschätzten Koeffizienten. Wie sind diese zu interpretieren?

Wie erhöht sich das Odds (d .h. die 'Chance') zu versterben, wenn eine direkte Verbrennung vorliegt?

```r
logmodel1 <- glm(Death~Flame, data = burn, family = binomial)
summary(logmodel1)
```

```
##
## Call:
## glm(formula = Death ~ Flame, family = binomial, data = burn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7510  -0.7510  -0.2946  -0.2946   2.5136
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.1157     0.2285 -13.637  < 2e-16 ***
## Flameyes      1.9943     0.2498   7.984 1.42e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 845.42  on 999  degrees of freedom
## Residual deviance: 755.46  on 998  degrees of freedom
## AIC: 759.46
##
## Number of Fisher Scoring iterations: 5
```

Es zeigt sich ein signifikanter Zusammenhang zwischen der direkten Verbrennung und dem Risiko zu versterben.

Mithile der `coef()`-Funktion können die Koeffizienten (auf Logit-Ebene) separat ausgegeben werden:

```r
coef(logmodel1)
```

```
## (Intercept)    Flameyes
##   -3.115735    1.994308
```

Der erste Koeffizient (entspricht β_0) beschreibt das geschätzte Logit (logarithmiertes Odds) in der Referenzkategorie (Flame='no'), der zweite Koeffizient (β_1) den **Unterschied** zwischen beiden Gruppen (Flame='no' gegen Flame='yes') auf der Ebene der Logits. Standardmäßig wird in R immer diejenige Kategorie zur Referenz, die alphabetisch die erste ist.

Um Odds und Odds Ratios zu erhalten, müssen diese Koeffizenten exponentiert werden (e^{β_0}, e^{β_1}):

```
exp(coef(logmodel1))
```

```
## (Intercept)    Flameyes
##   0.0443459   7.3471173
```

Der erste Koeffizient entspricht nun dem Odds (d. h. der 'Chance') zu versterben für Personen in der Referenzkategorie. Der zweite Koeffizient beschreibt das Odds Ratio (d. h. Verhältnis der Chancen) zwischen beiden Gruppen: Um welchen Faktor ist das Odds der zweiten Gruppe (Flame='yes') höher als das Odds der Referenzkategorie (Flame='no'). Das Odds der zweiten Gruppe ergibt sich deshalb aus dem Produkt beider Koeffizienten.

1.3 einfache logistische Regression mit metrischem Prädiktor

Nun möchten wir den Einfluss des metrischen Risikofaktors "TBSA" (Anteil der verbrannten Hautfläche) auf die Versterbenswahrscheinlichkeit betrachten. Alle anderen möglichen Einflussgrößen lassen wir zunächst außen vor.

Aufgabe 3

Passen Sie mithilfe der `glm()`-Funktion eine logistische Regression mit `TBSA` als Prädiktor an.

Ist der Zusammenhang zwischen TBSA und der Versterbenswahrscheinlichkeit statistisch signifikant?

Berechnen Sie das geschätzte Odds Ratio. Wie ist dieses zu interpretieren?

Wie hoch ist das Odds Ratio bei der Erhöhung des Anteils der verbrannten Hautfläche um 10 Prozentpunkte?

```r
logmodel2 <- glm(Death~TBSA, data = burn, family = binomial)
summary(logmodel2)
```

```
##
## Call:
## glm(formula = Death ~ TBSA, family = binomial, data = burn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1709  -0.3976  -0.3112  -0.2745   2.5680
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.345107   0.175648  -19.04   <2e-16 ***
## TBSA         0.085367   0.006956   12.27   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 845.42  on 999  degrees of freedom
## Residual deviance: 538.65  on 998  degrees of freedom
## AIC: 542.65
##
## Number of Fisher Scoring iterations: 5
```

Es zeigt sich ein signifikanter Zusammenhang zwischen dem Anteil der verbrannten Hautfläche und dem Risiko zu versterben.

```r
coef(logmodel2)
```

```
## (Intercept)        TBSA
## -3.34510747  0.08536677
```

Der erste Koeffizient (entspricht β_0) beschreibt das geschätzte Logit (logarithmiertes Odds) bei einem TBSA-Wert von 0 (d. h. den Schnittpunkt mit der y-Achse), der zweite Koeffizient (β_1) den Anstieg der Geraden auf der Ebene der Logits. Erhöht sich der Anteil der verbrannten Haut um eine Einheit (d. h. einen Prozentpunkt), dann steigt das Logit um β_1.

Um Odds und Odds Ratios zu erhalten, müssen diese Koeffizienten exponentiert werden (e^{β_0}, e^{β_1}):

```
exp(coef(logmodel2))

## (Intercept)        TBSA
##  0.03525643  1.08911645
```

Der erste Koeffizient entspricht nun dem geschätzten Odds (d. h. der 'Chance') zu versterben für Personen mit einem TBSA-Wert von 0. Der zweite Koeffizient beschreibt das **Odds Ratio** (d. h. Verhältnis der Chancen) zwischen zwei Patienten, die sich im TBSA-Wert um eine Einheit unterscheiden. Die Annahme, die hierbei in der logistischen Regression steckt, ist, dass das Odds Ratio konstant ist: Es ist egal, ob es sich beispielsweise um einen Unterschied zwischen TBSA-Werten von 10 und 11 oder von Werten von 85 und 86 handelt. Ob diese Linearitätsannahme gerechtfertigt ist, müsste in einer Modelldiagnostik überprüft werden.

Um das Odds Ratio zwischen zwei Patienten, die sich um 10 TBSA-Einheiten unterscheiden, zu ermitteln, wird der zweite Koeffizient zehnmal angewendet, d. h. $(e^{\beta_1})^{10}$ berechnet:

```
exp(coef(logmodel2)[2])^10

##     TBSA
## 2.348244
```

Die gleiche Zahl ergibt sich, wenn der Koeffiezent zunächst mit 10 multipliziert und dann exponenziert wird ($e^{10 \cdot \beta_1}$):

```
exp(10*coef(logmodel2)[2])

##     TBSA
## 2.348244
```

Wahrscheinlichkeiten. Um die Logits, die mit der logistischen Regression geschätzt werden, in Wahrscheinlichkeiten umzurechnen, verwendet man die logistische Funktion:

$$x \mapsto \frac{\exp(\beta_0 + \beta_1 \cdot x)}{1 + \exp(\beta_0 + \beta_1 \cdot x)}$$

Wir möchten nun diese geschätzten Wahrscheinlichkeiten graphisch darstellen.

Die Variable `Death` enthält die beiden Werte `Alive` und `Dead`. Um diese als 0 und 1

darstellen zu können, müssen wir uns zunächst eine neue Variable definieren:

```
burn$Death2 <- burn$Death=="Dead"
```

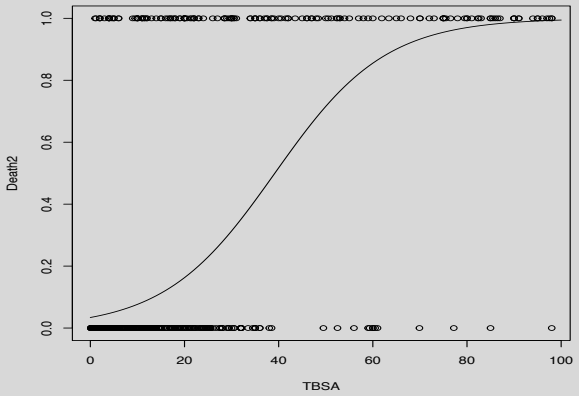
Diese neue Spalte enthält nun die Werte `TRUE` (für verstorbene Patienten) und `FALSE` (für Überlebende). Diese werden in Graphiken automatisch als 1 bzw. 0 dargestellt.

Aufgabe 4

Zeichnen Sie zunächst den Zusammenhang zwischen dem Versterben (`Death2`) und dem Anteil der verbrannten Hautfläche (TBSA) in einem Streudiagramm.

Versuchen Sie nun mithilfe der Funktion `curve()` die oben beschriebene logistische Funktion einzuzeichnen. Setzen Sie hierfür die geschätzten Koeffizienten β_0 und β_1 aus Ihrem angepassten Modell ein. Die Grenzen der Kurve geben Sie mit `from=` und `to=` an. Damit kein neues Diagramm erstellt wird, sondern die Kurve in das obige Diagramm eingezeichnet wird, verwenden Sie die Option `add=T`.

```
# Darstellung als Streudiagramm
plot(Death2 ~ TBSA, data = burn)

# Umrechnen der Logits in Wahrscheinlichkeiten anhand der angegebenen
# Formel. beta0 und beta1 werden mit coef(logmodel2)
# aus dem angepassten Modell ausgegeben.
betas <- coef(logmodel2)

curve(  exp(betas[1] + betas[2]*x ) /
          ( 1 + exp(betas[1] + betas[2]*x )  ),

        from = 0, to = 100, add=TRUE )
```



1.4 Auswahl des additiven Modells

Bisher haben wir nur den Einfluss jeweils eines möglichen Risikofaktors auf die Sterbewahrscheinlichkeit betrachtet. Nun möchten wir ein Modell mit allen Risikofaktoren gemeinsam erstellen. Dieses soll dann mit *Backward Selection* vereinfacht werden, sodass nur noch die Einflussgrößen im Modell bleiben, die einen signifikanten Zusammenhang zeigen.

Aufgabe 5

Passen Sie zunächst ein volles additives Modell mit allen Prädiktoren außer `ID` und `Facility` an.

Vereinfachen Sie dieses Modell schrittweise bei nicht-signifikanten Prädiktoren ($\alpha = 5\%$).

Tipp: Um Signifikanzen zu beurteilen, sollten Sie den Likelihood-Ratio-Test (LRT) mit `drop1(..., test="Chisq")` verwenden und nicht die P-Werte, die im `summary`-Befehl aufgeführt werden.

Anpassung des vollen Modells:

```r
logmodel3 <- glm(Death ~ Age + Gender  + TBSA + Inhalation + Flame,
                 family=binomial, data=burn)
summary(logmodel3)
```

```
##
## Call:
## glm(formula = Death ~ Age + Gender + TBSA + Inhalation + Flame,
##     family = binomial, data = burn)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.04961  -0.25696  -0.09212  -0.03670   2.52857
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -7.719152   0.677906 -11.387  < 2e-16 ***
## Age            0.078780   0.008174   9.638  < 2e-16 ***
## GenderMale    -0.275483   0.300888  -0.916     0.36
## TBSA           0.087889   0.009041   9.722  < 2e-16 ***
## Inhalationyes  1.380090   0.354293   3.895 9.81e-05 ***
## Flameyes       0.435599   0.347289   1.254     0.21
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 845.42  on 999  degrees of freedom
## Residual deviance: 341.66  on 994  degrees of freedom
## AIC: 353.66
##
## Number of Fisher Scoring iterations: 7
```

Zur Beurteilung der Signifikanzen ist der Likelihood-Ratio-Test besser geeignet als der Wald-Test (welcher in der `summary` gezeigt wird). Dieser wird über die Funktion `drop1()` mit der Option `test="Chisq"` berechnet:

```
drop1(logmodel3, test="Chisq")

## Single term deletions
##
## Model:
## Death ~ Age + Gender + TBSA + Inhalation + Flame
##            Df Deviance    AIC     LRT  Pr(>Chi)
## <none>          341.66 353.66
## Age         1   499.19 509.19 157.526 < 2.2e-16 ***
## Gender      1   342.50 352.50   0.833    0.3615
## TBSA        1   521.32 531.32 179.660 < 2.2e-16 ***
## Inhalation  1   356.87 366.87  15.211 9.616e-05 ***
## Flame       1   343.28 353.28   1.614    0.2040
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

`Gender` ist nichtsignifikant und hat den höchsten p-Wert. Mithilfe der `update`-Funktion lässt es sich aus dem Ausgangsmodell entfernen:

```
logmodel4 <- update(fm.burn.1, . ~ . - Gender)
logmodel4

##
## Call:  glm(formula = Death ~ Age + TBSA + Inhalation + Flame, family = binomial,
##     data = burn)
##
## Coefficients:
##   (Intercept)            Age           TBSA  Inhalationyes       Flameyes
##      -7.95897        0.07989        0.08740        1.38280        0.44376
##
## Degrees of Freedom: 999 Total (i.e. Null);  995 Residual
## Null Deviance:     845.4
## Residual Deviance: 342.5  AIC: 352.5
```

Nun wird wieder auf Grundlage der Signifikanzen beurteilt, ob eine weitere Einflussgröße entfernt werden kann:

```
drop1(logmodel4, test="Chisq")

## Single term deletions
##
## Model:
## Death ~ Age + TBSA + Inhalation + Flame
##            Df Deviance    AIC     LRT Pr(>Chi)
## <none>          342.50 352.50
## Age         1   506.01 514.01 163.514  < 2e-16 ***
## TBSA        1   521.33 529.33 178.831  < 2e-16 ***
## Inhalation  1   357.79 365.79  15.294  9.2e-05 ***
## Flame       1   344.18 352.18   1.683   0.1946
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nun hat `Flame` den höchsten nichtsignifikanten p-Wert. Obwohl diese Einflussgröße in der einfachen Regression (s. oben) signifikant war, ist sie es nun im komplexeren Modell nicht mehr. Sie trägt nicht mehr wesentlich zur Erklärung der Zielgröße bei, wenn die anderen Einflussgrößen gemeinsam betrachtet werden, und kann analog zu oben aus dem Modell entfernt werden:

```
logmodel5 <- update(logmodel4, . ~ . - Flame)
logmodel5

##
## Call:  glm(formula = Death ~ Age + TBSA + Inhalation, family = binomial,
##     data = burn)
##
## Coefficients:
##   (Intercept)            Age           TBSA  Inhalationyes
##      -7.74279        0.08089        0.08885        1.49971
##
## Degrees of Freedom: 999 Total (i.e. Null);  996 Residual
## Null Deviance:     845.4
## Residual Deviance: 344.2  AIC: 352.2
```

Ein erneuter Aufruf der `drop1`-Funktion ergibt keine weiteren nichtsignifikanten Koeffizienten:

```
drop1(logmodel5, test="Chisq")

## Single term deletions
##
## Model:
## Death ~ Age + TBSA + Inhalation
##            Df Deviance    AIC     LRT  Pr(>Chi)
## <none>          344.18 352.18
## Age         1   520.73 526.73 176.556 < 2.2e-16 ***
## TBSA        1   533.54 539.54 189.361 < 2.2e-16 ***
## Inhalation  1   362.98 368.98  18.798 1.453e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Das nun erhaltene Modell könnte nun noch um mögliche Interaktionen zwischen den Einflussgrößen ergänzt werden. Zudem sollte die Vorhersagekraft und Adäquatheit des Modells untersucht werden. Insbesondere der unterstellte lineare Zusammenhang zwischen den metrischen Einflussgrößen und den Logits sollte überprüft werden. Leider ist dies nicht so einfach über Diagnostikplots möglich wie bei der linearen Regression. Welche Möglichkeiten es hierfür gibt, werden Sie in der nächsten Veranstaltung erfahren.

Fortgeschrittene Methoden der Biometrie

– Logistische Regression –

IMB
TU Dresden

WS 2021/22

Übungsblatt identisch zu unserem!

In der letzten Übung haben Sie ein logistisches Modell für den `burn`-Datensatz angepasst. Zielgröße war das Überleben der Patienten bei Verbrennungen. Nach einer Modellauswahl zeigten drei Einflussgrößen einen signifikanten Effekt: die metrischen Größen Alter und Anteil der verbrannten Hautfläche (TBSA) sowie die kategoriale Größe Inhalationstrauma.

Aufgabe 1: Laden der Daten und Modellanpassung

Laden Sie den Datensatz `burn.dat` erneut und passen Sie das finale logistische Modell der letzten Übung erneut an.

2 Modell-Diagnostik bei der logistischen Regression

2.1 Linearitätsannahme bei metrischen Prädiktoren

Die logistische Regression trifft die Annahme, dass sich die Logits mit einem metrischen Prädiktor linear verändern. Im Rahmen der Modell-Diagnostik sollte diese Annahme überprüft werden. Dafür wird der Prädiktor in Kategorien eingeteilt (z. B. in vier gleich große Gruppen anhand der Quartile). Dann wird das logistische Modell erneut angepasst, wobei der ursprünglich metrische Prädiktor durch seine kategorisierte Variante ersetzt wird. Die dabei erhaltenen Logits (d. h. die Koeffizienten des logistischen Modells) können nun graphisch dargestellt werden. Liegen sie näherungsweise auf einer Geraden, so kann man davon ausgehen, dass die Linearitätsannahme an den metrischen Prädiktor gerechtfertigt war.

Aufgabe 2: Überprüfen der Linearität in den Logits

Im ePortal finden Sie das R-Skript `logistische_Regression_Modelldiagnostik.R`. Dieses Skript enthält die Funktion `testLinearity()`, welche die oben beschriebenen Schritte durchführt. Laden Sie sich das Skript herunter. Um es in RStudio zu importieren, verwenden Sie den folgenden Befehl:

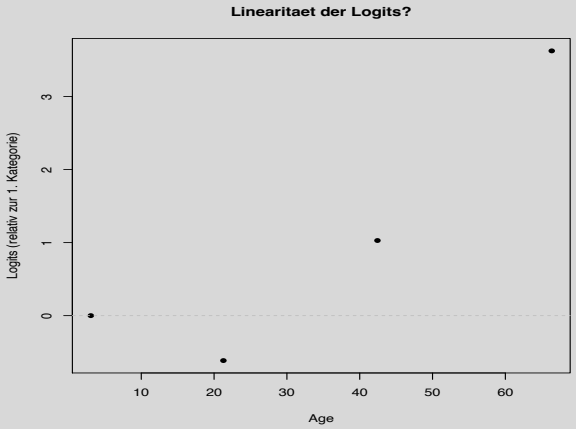
```
source("logistische_Regression_Modelldiagnostik.R")
```

Überprüfen Sie zunächst die Linearitätsannahme für den metrischen Prädiktor `Age` mithilfe des folgenden Aufrufs:

```
testLinearity(data = burn, variable = "Age", model = Modellname )
```

Ist die Linearitätsannahme für das Alter gerechtfertigt? Was könnte unternommen werden, falls dem nicht so ist?

Durch Aufruf des obigen Befehls erhalten Sie folgende Graphik:



Hierbei wird das Alter anhand der Quartile in vier gleich große Gruppen eingeteilt. Die metrische Variable Alter wird dann durch diese kategorisierte Variable ersetzt. Die erste Gruppe mit den jüngsten Patienten ist die Basiskategorie, sie hat daher den Wert (das Logit) 0. Die anderen drei Kategorien werden in Relation dazu berechnet. Auf der x-Achse ist jeweils das mittlere Alter jeder Kategorie dargestellt. Liegen die berechneten Logits ungefähr auf einer Geraden, ist die Linearitätsannahme gerechtfertigt. Dann kann die ursprüngliche metrische Variable verwendet werden. In unserem Beispiel liegen die drei rechten Punkte auf einer Geraden, der erste Datenpunkt (dieser entspricht kleineren Kindern) weicht hingegen von dieser Geraden ab. Es könnte deshalb eine zusätzliche Variable eingeführt werden, welche kleinere Kinder vom Rest der Patienten unterscheidet.

Die Überprüfung der Linearität in den Logits legt nahe, dass der metrische Prädiktor `Age` geändert werden sollte.

Aufgabe 3

Junge Patienten zeigen nicht den typischen Alterseffekt. Führen Sie daher einen zusätzlichen 2-stufigen Faktor `Child` ein, der Patienten $\leq$ 10 Jahre vom Rest unterscheidet. Tipp: Verwenden Sie die R-Funktion `factor`, analog zum Vorgehen auf dem `airquality`-Datensatz in der ersten Übung zur linearen Regression.

Fügen Sie den Faktor `Child` im logistischen Modelll hinzu (z. B. mithilfe der `update()`-Funktion). Überprüfen Sie schließlich erneut mit der `drop1()`-Funktion die Signifikanzen der Prädiktoren ($\alpha = 5\%$) in diesem neuen Modell.

Mithilfe der Funktion `factor` legen Sie eine neue Variable an, die Kinder mit einem Alter von kleiner oder gleich 10 Jahren vom Rest unterscheidet:

```
burn$Child <- factor(burn$Age <= 10, levels=c(FALSE,TRUE),
                     labels=c("no", "yes"))
```

Der Teil `levels=c(FALSE,TRUE)` ist hierbei nicht zwingend nötig, hilft aber bei der Zuordnung der Namen (”no”,”yes”) in der richtigen Reihenfolge.

Nun wird die neue Variable `Child` mit in das Modell aufgenommen:

```
logmodel6 <- update(logmodel5, .~. + Child)
```

Mithilfe von `drop1` wird die Signifikanz überprüft:

```
drop1(logmodel6, test="Chisq")

## Single term deletions
##
## Model:
## Death ~ Age + TBSA + Inhalation + Child
##            Df Deviance    AIC     LRT  Pr(>Chi)
## <none>          330.03 340.03
## Age         1   497.27 505.27 167.240 < 2.2e-16 ***
## TBSA        1   525.28 533.28 195.256 < 2.2e-16 ***
## Inhalation  1   347.09 355.09  17.062 3.618e-05 ***
## Child       1   344.18 352.18  14.152 0.0001686 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alle Variablen sind signifikant. Die entsprechenden Odds Ratios sind:

```
exp(coef(logmodel6))

##   (Intercept)           Age          TBSA Inhalationyes      Childyes
##  1.263135e-04  1.104047e+00  1.096294e+00  4.277487e+00  1.816696e+01
```

Patienten unter 10 Jahren haben laut Modell also ein höheres Risiko zu versterben. Allerdings muss beachtet werden, dass auch das Alter noch als metrische Größe im Modell drinsteckt, wodurch eine sinnvolle Interpretation des Odds Ratios erschwert wird. Überlegenswert wäre, statt mit der metrischen Größe Alter mit den vier zuvor berechneten Alterskategorien zu arbeiten, um die Interpretation zu erleichtern.

Zusatz: Wie Sie diese alternative Einteilung in vier Altersklassen umsetzen könnten, wollen wir Ihnen im Folgenden zeigen.

Zunächst berechnen wir die Quartile:

```
ageQuantiles <- quantile(burn$Age)
ageQuantiles

##     0%    25%    50%    75%   100%
##  0.100 10.850 31.950 51.225 89.700
```

Mithilfe der Funktion `quantile()` können beliebige Quantile berechnet werden. Ohne Angabe des gewünschten Quantils gibt die Funktion automatisch die Quartile aus. Nun können die Daten anhand dieser Quartile in vier Gruppen eingeteilt werden. Dafür verwenden wir die Funktion `cut()`:

```
burn$Age_cat <- factor(
  cut(x=burn$Age, breaks=ageQuantiles,include.lowest=T),
       labels= c(1,2,3,4))
```

Die Function `cut()` ordnet jedem Patienten das jeweilige Altersintervall zu, in welchem er sich befindet. Mit der Option `breaks` geben Sie an, wo die Grenzen der Intervalle liegen, in unserem Fall die Quartile der Altersverteilung. Die Option `include.lowest=T` ist hier nötig, damit der kleinste Wert dem ersten Intervall zugeordnet wird. Er würde ansonsten nicht mit aufgenommen. Im Anschluss werden diese Intervalle in einen Faktor umgewandelt, den vier Intervallen werden hier die Namen 1, 2, 3 und 4 zugeordnet.

Nun können Sie im Modell die metrische Variable `Age` gegen die neue Variable `Age_cat` austauschen, bspw. wieder mithilfe der `update`-Funktion:

```
logModel_Age_cat <- update(logmodel5, .~. - Age + Age_cat)
summary(logModel_Age_cat)

##
## Call:
## glm(formula = Death ~ TBSA + Inhalation + Age_cat, family = binomial,
##     data = burn)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -3.00584  -0.21265  -0.11048  -0.07107   2.62207
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -5.706188   0.588121  -9.702  < 2e-16 ***
## TBSA           0.085384   0.008499  10.046  < 2e-16 ***
## Inhalationyes  1.465006   0.331692   4.417 1.00e-05 ***
## Age_cat2      -0.615312   0.762303  -0.807   0.4196
## Age_cat3       1.028316   0.601499   1.710   0.0873 .
## Age_cat4       3.624702   0.560463   6.467 9.97e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 845.42  on 999  degrees of freedom
## Residual deviance: 369.79  on 994  degrees of freedom
## AIC: 381.79
##
## Number of Fisher Scoring iterations: 7
```

Das Modell ergibt, dass nur die oberste Altersgruppe einen signifikanten Unterschied zur Basiskategorie der Kinder zeigt.

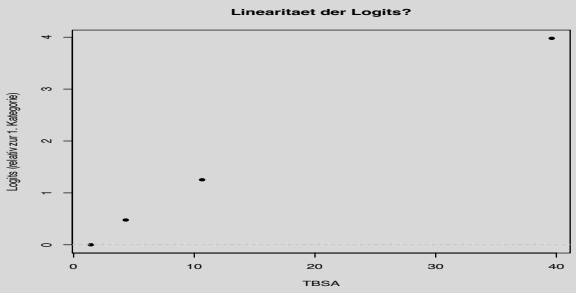
Im Folgenden sollen wir mit dem zuvor berechneten Modell `logmodel6` weiterrechnen.

Aufgabe 4

Überprüfen Sie nun für die zweite metrische Variable `TBSA`, ob die Linearitätsannahme gerechtfertigt ist. Überlegen Sie, was unternommen werden könnte, falls die Annahme nicht erfüllt ist.

Analog zum Vorgehen für das Alter wird die zweite metrische Größe TBSA überprüft:

```
testLinearity(data = burn, variable = "TBSA", model = logmodel6 )
```



Ob die Linearitätsannahme hier verworfen werden sollte, ist eine subjetive Entscheidung. Die Punkte bilden keine optimale Gerade: Bei einem großen Anteil verbrannter Haut ist das geschätzte Logit kleiner als durch die Linearitätsannahme erwartet. Es könnte versucht werden, die Daten zu transformieren, bspw. mit der Wurzel oder dem Logarithmus. Alternativ könnte statt der metrischen Größe auch die kategorisierte Größe verwendet werden. Wir wollen es hier bei der ursprünglichen metrischen Variable belassen.

Zusatz: Es könnte nun versucht werden, zusätzliche Interaktionseffekte mit in das Modell aufzunehmen. Dafür kann in R die Funktion `add1` verwendet werden (analog zu `drop1`). Wir wollen an dieser Stelle darauf verzichten und es beim zuvor aufgestellten Modell belassen.

3 Modellgüte und Klassifizierung

Nun soll es nun darum gehen, wie gut ein ausgewähltes Modell eigentlich die Daten beschreibt. Dies ist die entscheidende Frage, bevor man ein Modell tatsächlich verwendet.

3.1 Anpassungsgüte: der Hosmer-Lemeshow-Test

Die Idee des Hosmer-Lemeshow-Tests ist es, zu schauen, wie gut die vorhergesagten Wahrscheinlichkeiten tatsächlich mit der beobachteten Zielvariable übereinstimmen. Dazu werden die Patienten des Datensatzes nach ihrer *vorhergesagten* Sterbewahrscheinlichkeit in 10 gleich große Gruppen eingeteilt. Innerhalb dieser Gruppen wird geschaut, ob die beobachtete Sterbewahrscheinlichkeit mit der mittleren vorhergesagten Wahrscheinlichkeit zusammen passt.

Aufgabe 5

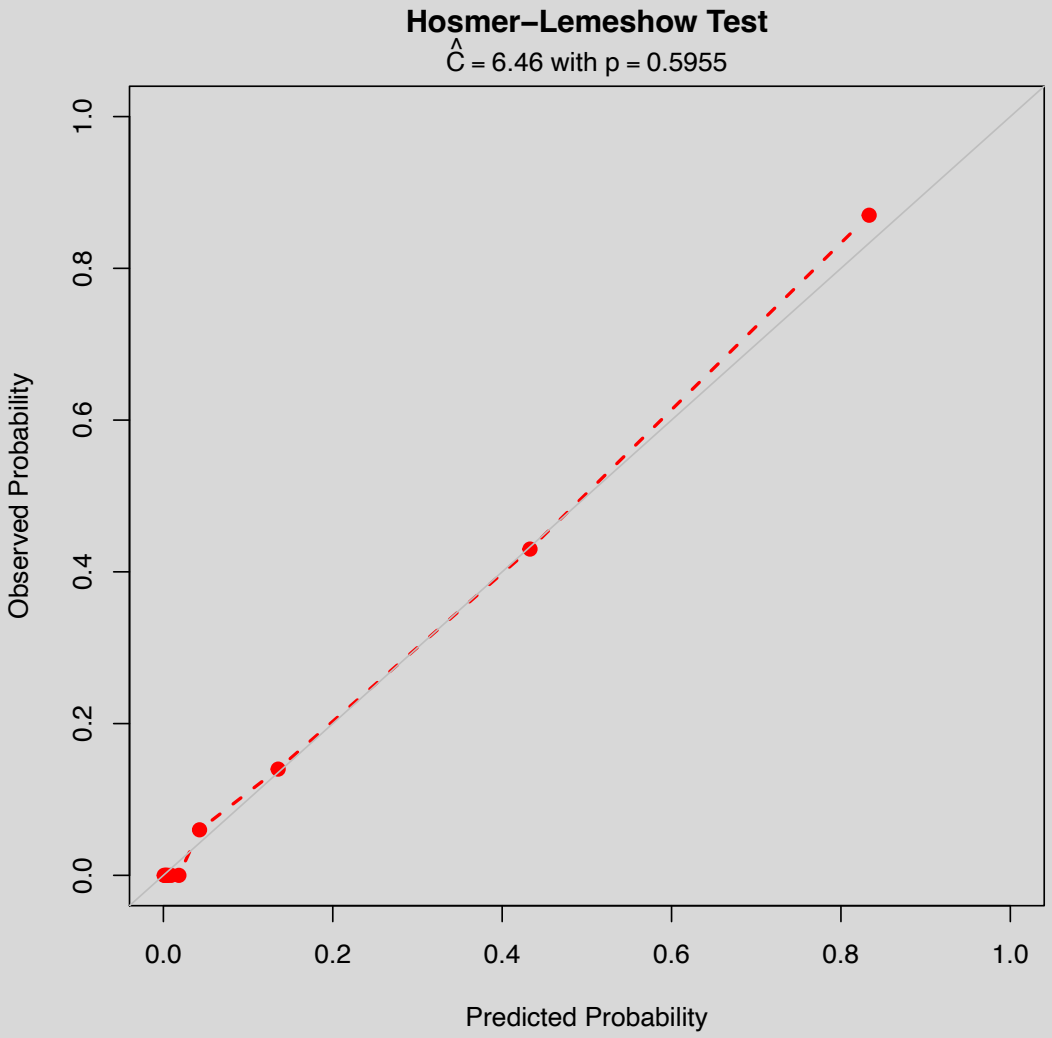
Die Funktion für den Hosmer-Lemeshow-Test ist ebenfalls im R-Skript `logistische_Regression_Modelldiagnostik.R` definiert. Sie rufen ihn folgendermaßen auf:

`hosmerLemeshowTest(`*`Modellname`*`)`

Interpretieren Sie die dargestellte Graphik.

Die Nullhypothese des Hosmer-Lemeshow-Tests lautet: Die vom Modell berechneten Wahrscheinlichkeiten passen zu den beobachteten Wahrscheinlichkeiten in den Daten. Deuten Sie den p-Wert, den Sie über der angegebenen Grafik entnehmen können.

```
hosmerLemeshowTest(logmodel6)
```



Die vorhergesagten und beobachteten Wahrscheinlichkeiten in den 10 Gruppen stimmen bei Betrachtung der Graphik gut überein. Die dargestellten Punkte liegen weitgehend auf der Winkelhalbierenden.

Der p-Wert liegt bei 0.6. Das heißt, dass die Nullhypothese aufgrund der Daten nicht abgelehnt werden muss. Der Hosmer-Lemeshow-Test findet also keine signifikante Abweichung zwischen den vorhergesagten und den beobachteten Sterbe-Wahrscheinlichkeiten, insofern passt das Modell zu den Daten.

3.2 Klassifizierung

Logistische Regressionsmodelle werden oft benutzt, um Wahrscheinlichkeitsvorhersagen zu machen. Wendet man einen Schwellwert (dieser wird im Englischen als "Cutoff" bezeichnet) auf die vorhergesagten Wahrscheinlichkeiten an, dann bekommt man eine Klassifizierung der Patienten.

Aufgabe 6

- Nutzen Sie die `predict`-Funktion mit dem Argument `type="response"` auf dem ausgewählten Modell, um die vorhergesagten Wahrscheinlichkeiten für die Patienten auszurechnen. Speichern Sie diese im Dataframe als neue Spalte.
- Wie werden die Patienten klassifiziert, wenn Sie einen Schwellwert von 50% annehmen?
- Machen Sie eine Gegenüberstellung mit der wirklich beobachteten Zielgröße. Tipp: `xtabs`.
- Wie groß ist Sensitivität und Spezifität auf den Daten, wenn man die obige Grenze anlegt?
- Was würden Sie erwarten, wenn Sie Ihr Modell auf neue Testdaten anwenden? Wird die Güte des Tests die oben berechneten Sensitivitäts- und Spezifitätswerte erreichen?

Mit `predict` lassen sich die für jeden Patienten vorhergesagten Wahrscheinlichkeiten berechnen. Die Angabe `type="response"` sorgt dafür, dass wirklich Wahrscheinlichkeiten ausgegeben werden. Ohne diese Angabe erhielte man die Logits.

```
burn$pred <- predict(logmodel6, type="response")
```

Nun werden die Wahrscheinlichkeiten in zwei Gruppen eingeteilt: Für Wahrscheinlichkeiten über 0.5 wird Tod vorhergesagt, unter 0.5 Überleben:

```
burn$pred_cat <- factor(burn$pred > .5, levels = c(FALSE,TRUE),
                        labels = c("Alive", "Dead"))
```

Die folgende Vierfeldertafel stellt den Zusammenhang zwischen vorhergesagtem Status und Beobachtung dar:

```
vierfeldertafel <- xtabs(~Death + pred_cat, data=burn)
vierfeldertafel
```

```
##        pred_cat
## Death   Alive Dead
##   Alive   823   27
##   Dead     46  104
```

Von den 150 Toten werden durch das Modell 104 richtig klassifiziert. Das entspricht also einer Sensitivität von knapp 70%.

Von den 850 Überlebenden werden 823 richtig klassifiziert. Das entspricht also einer Spezifität von 97%.

Insgesamt wurden 927 der 1000 Patienten richtig klassifiziert.

Auf neuen Daten (=Testdaten) erwartet man etwas schlechtere Sensitivität und Spezifität, weil das Modell speziell auf die vorliegenden Trainingsdaten angepasst wurde. Um ein Vorhersage-Modell realistisch zu bewerten, wäre eine Möglichkeit, dass man dazu neue Testdaten verwendet, die zuvor nicht zur Modellanpassung verwendet wurden.

Übungsblatt identisch zu unserem!

Seminar 1
Einführung in die Statistik von Metaanalysen

Andrea Gottschalk, Ingmar Glauche
IMB, TU Dresden

2. Dezember 2021

Ziel des Seminars. Wir möchten Ihnen im Rahmen dieses Seminars die Fähigkeit vermitteln, eine einfache Metaanalyse mit der Statistiksoftware R durchzuführen. R bietet mit dem `metafor`-Package ein Werkzeug, um solche Auswertungen umzusetzen[1]. Damit ist es auf einfache Art und Weise möglich, die wesentlichen Inhalte der Vorlesung noch einmal am Computer nachzuvollziehen.

Für die exemplarische Durchführung stellen wir Ihnen Beispieldatensätze zur Verfügung, mit welchen Sie Fixed-Effect(FE)- und Random-Effects(RE)-Modelle erstellen und deren Ergebnisse in Forest-Plots darstellen können.

1 Vorbereitung

Wenn das `metafor`-Package bereits auf Ihrem Computer bereits ist, muss es am Anfang Ihres Skriptes neu geladen werden:

```
library(metafor)
```

Sollte das Paket noch nicht installiert sein, müssten Sie es über die Installationsfunktion von RStudio ergänzen. Alternative können Sie direkt in der Konsole die Installation starten:

```
install.packages("metafor")
```

Für weitere Informationen rund um das `metafor`-Projekt kann jederzeit die offizielle Webseite `http://www.metafor-project.org/doku.php` oder die RStudio-interne Hilfeseite genutzt werden:

```
help(metafor)
```

Um mit dem Package arbeiten zu können, müssen die zu untersuchenden Daten eingelesen und als internes Datenobjekt gespeichert werden. Im Praktikumsordner finden Sie zwei Textdateien mit jeweils sechs verschiedenen Studien. Die Datei `SMD_Data.txt` enthält Mittelwerte und Standardabweichungen für jeweils eine Fall- und Kontrollgruppe als primäre Studienresultate. Die Berechnung der

[1] Es gibt weitere derartige Pakete für R, beispielsweise `meta`.

Effektstärke erfolgt hier über die Verwendung standardisierter Differenzen von Mittelwerten. Obwohl die Studiendaten aus didaktischen Gründen „erfunden“ wurden, können Sie sich beispielsweise Konzentrationen zellulärer Bestandteile im Blut als Untersuchungseinheit vorstellen.
In der Datei `B_Data.txt` befinden sich binäre Daten. Hier können Odds Ratios oder Relative Risiken als Effektstärken verwendet werden, die beispielsweise Progression unter Krebstherapie beschreiben.
Setzen Sie als ersten Schritt das aktuelle Working-Directory mit `setwd()` auf den Pfad, in dem sich die beiden Textdateien befinden. Anschließend benutzen Sie die Syntax `read.table()` um die Dateien einzulesen:

```
# Setzen des aktuellen Arbeitsverzeichnisses
setwd("[Pfad_einfügen]")

# read.table = Einlesen von Daten
# header = Tabellenköpfe (nicht) übernehmen
# sep = Angabe des Daten-Seperators, hier = Tabulator

smd_data <- read.table("SMD_Data.txt",header=TRUE, sep="\t")
b_data <- read.table("B_Data.txt",header=TRUE, sep="\t")
```

Alternativ können Sie die Daten auch über den Button *Import Dataset* einlesen. Wenn Sie die Syntax richtig ausgeführt haben, erhalten Sie zwei Datenobjekte `smd_data` und `b_data`, in denen sich jeweils die Datensätze aus den Textdateien befinden. **Schauen Sie sich die Struktur der Textdateien und der R-Objekte an und vergleichen Sie diese.**

2 Durchführung einer Metaanalyse

Im folgenden Kapitel wird das `metafor`-Package vorgestellt. Die einzelnen Funktionen werden zuerst anhand eines FE-Modells genauer erläutert. Da die einzelnen Schritte bei der Berechnung von RE-Modellen grundlegend die gleichen sind, wird in Punkt 2.4 der Ablauf nur grob skizziert, sodass Sie die entsprechenden Berechnungen selbst ausführen können.

2.1 Berechnung von Effektstärke und Varianz

Für die Berechungen der Effektstärken und der zugehörigen Varianzen für die einzelnen Studien stellt das `metafor`-Package die Funktion `escalc()` zur Verfügung. Die Belegung der Argumente dieser Funktion ist dabei abhängig von der Art der Effektstärke, die berechnet werden soll.

`measure`: Angabe einer Zeichenkette, welche die Effektstärke beschreibt. Beispielhaft können hier

- `"SMD"` – für *standardisierte Differenz der Mittelwerte*,
- `"OR"` – für *Log Odds Ratio*, oder
- `"RR"` – für *relatives Risiko*

genannt werden.

`data`: Angabe der Datenmatrix

Neben der Angabe der Typ- und Datenvariablen muss der Funktion noch die Information der Spaltenzuordnung übergeben werden. Für die Effektstärke SMD sind die zu belegenden Argumente exemplarisch aufgeführt:

Typ	**Argument**	**Beschreibung**
SMD	m1i, m2i	Vektor für die Mittelwerte
	sd1i, sd2i	Vektor für die Standardabweichungen
	n1i, n2i	Vektor für Stichprobengrößen
	[1]=Experimentalgruppe, [2]=Kontrollgruppe	

Der R-Code für die SMD-Effektstärke sieht demnach folgendermaßen aus:

```
calc_smd <- escalc(measure="SMD"
                ,m1i=smd_data$Treated_Mean
                ,sd1i=smd_data$Treated_SD
                ,n1i=smd_data$Treated_N
                ,m2i=smd_data$Control_Mean
                ,sd2i=smd_data$Control_SD
                ,n2i=smd_data$Control_N
                ,data=smd_data)
```

Die `escalc()`-Funktion berechnet damit die Effektstärke Y und die Varianz V für jede Studie und fügt die Werte an die ursprüngliche Tabelle an.

Achtung: Zur Berechnung eines *Hedge's g* führt die `escalc()`-Funktion keine Bias-Korrektur für die Varianz durch. Um identische Ergebnisse zur Vorlesung zu erhalten, muss der Korrekturfaktor `J` manuell berechnet und zur Varianzkorrektur herangezogen werden. Dazu sind folgende Schritte erforderlich.

```
# Berechnung des Korrekturfaktors
J <- (1-3/(4*(calc_smd$Treated_N+calc_smd$Control_N-2)-1))
# Überschreiben der alten Varianz-Werte
calc_smd$vi <- J^2*calc_smd$vi
# Anzeigen der benannten Spalten
calc_smd[,c("Study","yi","vi")]

##     Study     yi     vi
## 1 Carroll 0.0945 0.0329
## 2   Grant 0.2774 0.0307
## 3    Peck 0.3665 0.0499
## 4   Donat 0.6644 0.0105
## 5 Stewart 0.4618 0.0426
## 6   Young 0.1852 0.0234
```

2.2 Berechnung des Summary-Effects (FE)

In der Vorlesung haben Sie gelernt, dass eine Metaanalyse mit einem von zwei statistischen Modellen durchgeführt wird. Die Auswahl der Modellklasse ist von Ihren Annahmen bezüglich der Verteilung der wahren Effektstärken abhängig. Beim Fixed-Effect(FE)-Modell wird angenommen, dass allen Studien eine gemeinsame, wahre Effektstärke zu Grunde liegt, während für ein Random-Effects-(RE)-Modell angenommen wird, dass die wahren Effektstärken einer Verteilung entstammen.

Im `metafor`-Package existiert die `rma`-Funktion. Diese Funktion wird benutzt, um FE- und RE-Modelle zu schätzen unter der Annahme, dass die Stichprobenfehler durch eine Normalverteilung approximiert werden können.

Die Belegung des `method`-Arguments ist Grundvoraussetzung für die Anwendung der `rma`-Funktion. Bei einem FE-Modell muss das Argument mit der Zeichenkette `"FE"` belegt werden. Weiterhin ist es notwendig, die bereits berechneten Effektstärken und die zugehörigen Varianzen der einzelnen Studien anzugeben (yi und vi). Optional können natürlich noch weitere Einstellungen vorgenommen werden (siehe Hilfe). Beispielsweise kann die Spalte der Datentabelle, welche die Namen der Einzelstudien enthält, über `slab` angegeben werden.

```
# FE-Modell für kontinuierliche Daten
res_smd.FE <- rma(yi, vi, data=calc_smd, method="FE", slab=Study)

# Anzeige FE-Modell
res_smd.FE

##
## Fixed-Effects Model (k = 6)
##
## I^2 (total heterogeneity / total variability):   58.35%
## H^2 (total variability / sampling variability):  2.40
##
## Test for Heterogeneity:
## Q(df = 5) = 12.0047, p-val = 0.0347
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.4143  0.0640  6.4747  <.0001  0.2889  0.5397  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Im Resultate-Teil werden dabei folgende Abkürzungen verwendet:

- `estimate` = gewichtetes Mittel
- `se` = Standardabweichung der gemittelten Effektstärke
- `zval / pval` = z-Wert / p-Wert

- `ci.lb / ci.ub` = untere und obere Grenze des Konfidenzintervalls

Interpretieren Sie Ihr Ergebnis. Wie müsste die Raute für die gemittelte Effektstärke M aussehen? Ist dieser Effekt als signifikant zu bewerten?

Zur schnellen Übersicht sind die Funktionen `weights()` und `predict()` interessant. Mit ihrer Hilfe können die relativen Gewichte der einzelnen Studien und der Summary-Effect quantitativ dargestellt werden. Als Argument wird hier eine Variable des Typs `rma` akzeptiert, also das Ergebnis der bereits durchgeführten Metaanalyse.

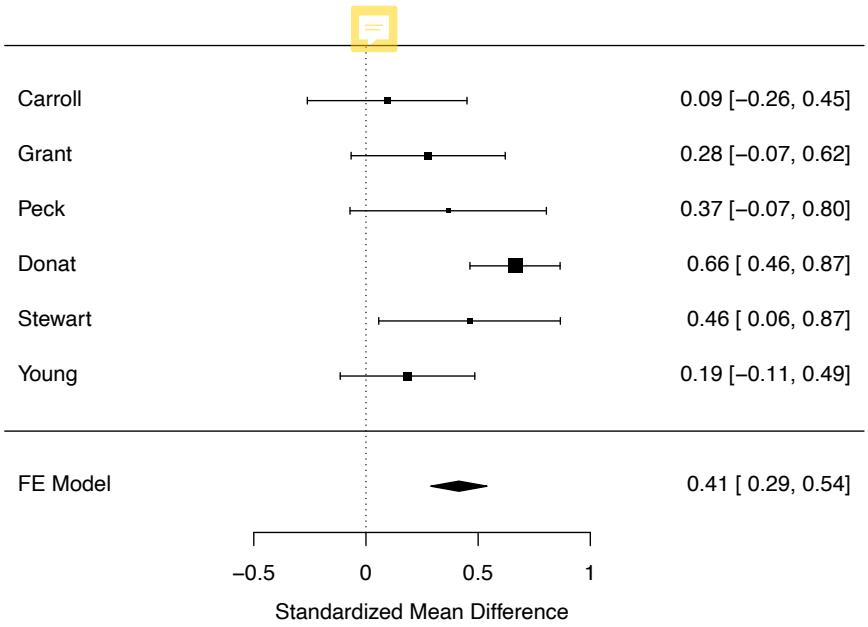
```
weights(res_smd.FE)

##   Carroll     Grant      Peck     Donat   Stewart     Young
## 12.426289 13.335105  8.210527 38.946581  9.600002 17.481496

predict(res_smd.FE)

##
##    pred     se  ci.lb  ci.ub
##  0.4143 0.0640 0.2889 0.5397
```

Eine gute Übersicht über das Funktionsspektrum kann unter dem Link: `https://cran.r-project.org/web/packages/metafor/vignettes/diagram.pdf` abgerufen werden.

2.3 Erstellen von Forest-Plots

`Metafor` bietet mit der Funktion `forest()` ebenfalls die Möglichkeit, die berechneten Gewichte, Effektstärken etc. in einem Forest-Plot zu visualisieren. Dazu wird erneut ein Daten-Objekt des Typs `rma` benötigt, welches die Grundlage des zu erstellenden Diagramms bildet. Die R-Syntax sieht folgendermaßen aus:

```
forest(res_smd.FE)
```



Wie Sie sehen können, wird in der Standardvariante ein übersichtliches Diagramm erstellt. Es werden sowohl die Studiennamen als auch die gewählten Effektstärken inklusive der Konfidenzintervalle dargestellt. Dennoch mag die Darstellung in einigen Bereichen nicht Ihren Erwartungen bezüglich der Vollständigkeit entsprechen. Wir haben weitere Details zur Ergänzung des Forest-Plots im Anhang erklärt und ermutigen Sie, diese ebenfalls auszuprobieren.

2.4 Durchführung einer RE-Metaanalyse

Es ist jetzt Ihre Aufgabe, die gleiche Analyse wie für das FE-Modell noch einmal mit einem RE-Modell durchzuführen. Einziger Unterschied ist die Spezifikation der „Methode“ innerhalb der `rma()`-Funktion. Beschränken Sie sich dabei bitte auf die Auswahl `method="DL"`, die den DerSimonian-Laird-Schätzer spezifiziert und damit der Berechnungsvorschrift entspricht, die Sie in der Vorlesung kennengelernt haben.

```
res_smd.RE <- rma(...)
```

Wie unterschieden sich Ihre Ergebnisse von der FE-Metaanalyse der gleichen Daten?

2.5 Verarbeitung von Binären Daten

Zur schrittweisen Veranschaulichung einer RE-Metaanalyse soll im folgenden noch eine Auswertung für binäre Daten vorgenommen werden. Wie Sie bereits gesehen haben, sind die wesentlichen Abläufe dabei identisch und verlangen nur geringe Anpassungen.

Vollziehen Sie die folgenden Ergebnisse schrittweise nach!

In der Tabelle `b_data` haben Sie bereits die Daten der zweiten Textdatei eingelesen. Um Effektstärke und Varianz zu berechnen, benötigen sie erneut die `escalc()`-Funktion. Am Beispiel des Odds Ratios seien hier die dafür benötigten Argumentbelegungen aufgeführt:

Typ	Argument	Beschreibung
OR	ai	Vektor 4-Feldertafel oben links
	bi	Vektor 4-Feldertafel oben rechts
	ci	Vektor 4-Feldertafel unten links
	di	Vektor 4-Feldertafel unten rechts

Und dazu die zu verwendende R-Syntax für die Berechnung:

```
calc_b_OR <- escalc(measure="OR"
                ,ai=b_data$Treated_Events
                ,bi=b_data$Treated_Non_Events
                ,ci=b_data$Control_Events
                ,di=b_data$Control_Non_Events
                ,data=b_data)
```

Beim RE-Modell existieren eine Reihe von Methoden zur Varianzschätzung. Im Rahmen dieser Übung beschränken Sie sich dabei bitte auf die `method="DL"`, die den DerSimonian-Laird-Schätzer spezifiziert. Nähere Informationen zu anderen Schätzern finden Sie in der Hilfe der `rma()`-Funktion.

Das entsprechende RE-Modell wird dann wie folgt spezifiziert und liefert den nachfolgenden Output:

```
# RE-Modell für binäre Daten mit DerSiomonian-Laird-Schätzer
res_b.RE <- rma(yi, vi, data=calc_b_OR, method="DL", slab=Study)

# Anzeige RE-Modell
res_b.RE

##
## Random-Effects Model (k = 6; tau^2 estimator: DL)
##
## tau^2 (estimated amount of total heterogeneity): 0.1729 (SE = 0.2148)
## tau (square root of estimated tau^2 value):      0.4158
## I^2 (total heterogeneity / total variability):   52.61%
## H^2 (total variability / sampling variability):  2.11
##
## Test for Heterogeneity:
## Q(df = 5) = 10.5512, p-val = 0.0610
##
```

```
## Model Results:
##
## estimate      se     zval    pval    ci.lb    ci.ub
##  -0.5663  0.2388  -2.3711  0.0177  -1.0344  -0.0982  *
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Neben der Ausgabe der Ergebnisse, welche auch bei einem FE-Modell angegeben werden, werden zusätzliche – für die Interpretation eines RE-Modells benötigte– Informationen in der Konsole angezeigt. Zur Wiederholung werden hier noch einmal die einzelnen Definitionen dieser Parameter aufgeführt:

Parameter	**Beschreibung**
Q	Summe der quadratischen Abweichungen der Effektstärke vom Summary-Effect
df	Anzahl der Freiheitsgrade
$Q - df$ (excess variation)	entspricht dem Anteil der Variabilität, der durch die Heterogenität der wahren Effektstärken (nicht der Stichprobenfehler) verursacht wird
p-Wert für (Q, df)	der p-Wert entspricht der Wahrscheinlichkeit, dass die beobachtete Heterogenität unter der Annahme einer gleichen, wahren Effektstärke zufällig zustande gekommen ist.
I^2	relatives Maß für den Anteil der excess variation an der gesamten Dispersion
tau^2	entspricht T^2, Schätzer für die Varianz der wahren Effektstärke
H^2	Verhältnis der beobachteten Heterogenität zur erwarteten Heterogenität unter der Annahme einer gleichen, wahren Effektstärke für alle Studien ($H^2 = Q/df$)

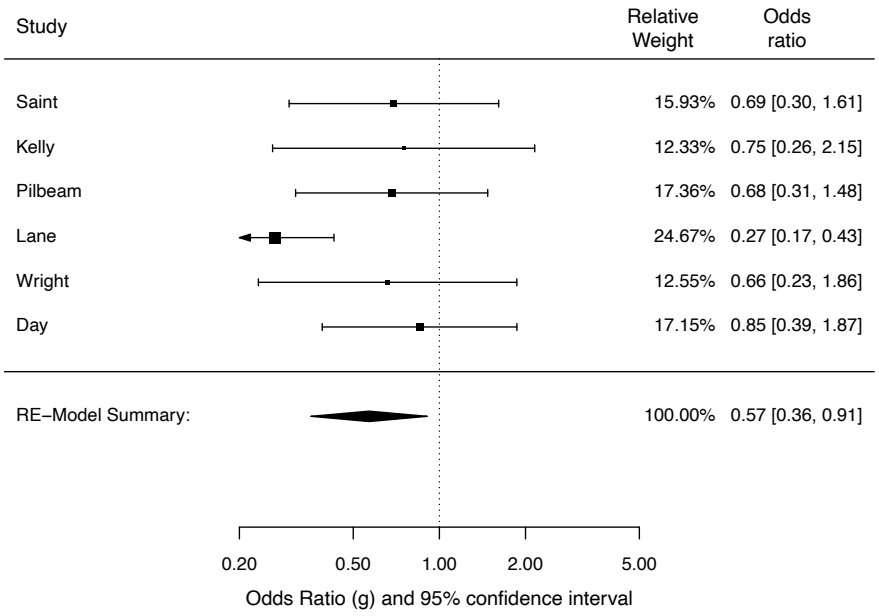
Um mit den eben benannten Ergebnissen arbeiten zu können, müssen diese nicht mühsam von der Konsole abgeschrieben werden, sondern sind innerhalb des `rma`-Objektes verfügbar. Das Objekt stellt eine Datenmatrix dar, auf deren Felder einzeln zugegriffen werden kann. Im Anhang finden Sie ein Beispiel, dass den Zugriff auf die einzelnen Objektfelder zeigt.

Mit Hilfe der Ihnen bekannten Plot-Funktion können die Daten nun visualisiert werden. Da das Odds Ratio logarithmiert vorliegt, muss die x-Achse für eine korrekte Darstellung ebenfalls transformiert werden. Mit `atransf` wird dabei der Wertebereich logarithmiert und mit `at` die x-Achse skaliert. Mit dem Parameter `refline` kann zusätzlich die Position der vertikalen Achse auf der x-Achse

verändert werden.

Weitere Informationen wie T^2, oder I^2 müssen zusätzlich ergänzt werden. Eine kurze Beschreibung haben wir im Anhang angefügt.

```
# Grundeinstellung Grafikparameter
par(mar=c(6,4,1,2),cex=.9, font=1)

# Random-Effects-Model mit binären Daten
forest(res_b.RE,showweights = TRUE,digits=2
       ,xlim=c(-3.5,3.5)
       ,ylim=c(-3,9)
       ,refline=log(1)
       ,at=log(c(0.2,0.5,1,2,5)) # Einteilung x-Achse
       ,xlab="Odds Ratio (g) and 95% confidence interval"
       ,mlab="RE-Model Summary:"
       ,atransf=exp) # logarithm. Transformierung x-Achse

# Spaltenbeschriftungen
par(font=1)
text(c(-3.2,1.8,2.8), 7.7, c("Study","Relative\nWeight"
       ,"Odds\nratio"))
```



3 Aufgabe: Selbstständiges erstellen einer weiteren Metaanalyse

In diesem Abschnitt sollen Sie nun das erlangte Wissen nutzen, um eine Metaanalyse selbstständig durchzuführen. Innerhalb des Metafor-Package befindet sich ein Datensatz aus *Colditz et al. (1994)*, welcher die Effektivität von Bacillus-Calmette-Guerin(BCG)-Impfungen zur Prävention von Tuberkulose aus 13 Studien enthält. Laden Sie den Datensatz mit dem Befehl:

```
data(dat.bcg)
```

Für die Analyse ist – neben dem Studiennamen – die Verteilung der einzelnen Spalten in der 4-Felder-Tafel relevant:

	TB positiv	**TB negativ**
Experimentalgruppe	tpos	tneg
Kontrollgruppe	cpos	cneg

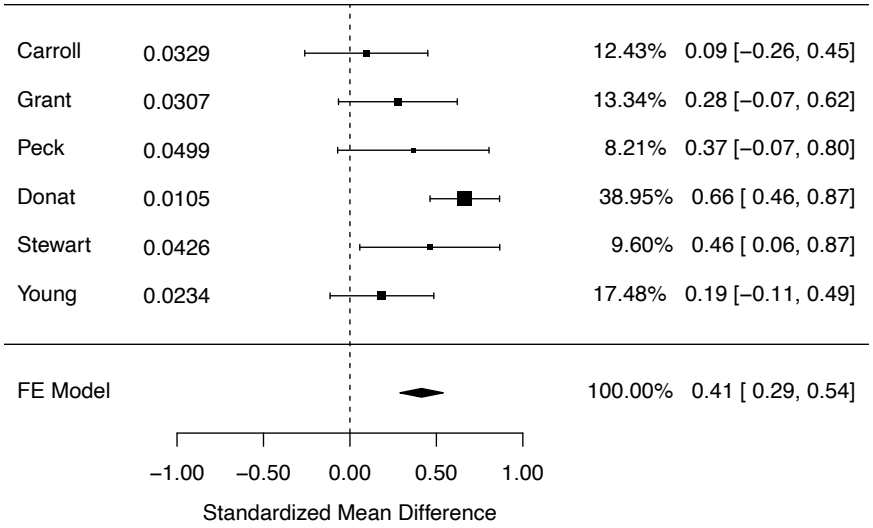
Benutzen Sie nun die Daten (unter Verwendung eines RE-Modells), um einen Forestplot zu erstellen. Achten Sie dabei auf geeignete Beschriftungen und Skalierung. **Verwenden Sie als Effektstärke das Relative Risiko und interpretieren Sie das Gesamtergebnis: Ist es sinnvoll gegen Tuberkulose zu impfen?**

4 Anhang: Optimierung der Forest-Plots

Die Standardversion der Forest-Plots im `Metafor`-Paket ist eher einfach. Manchmal sind jedoch Ergänzungen, wie die Anzeige von zusätzlichen Spalten (z.B. der relativen Gewichte), die Modifikation bzgl. der x-Achse, oder die manuelle Positionierung der grafischen Elemente sinnvoll oder erforderlich. Die Plot-Funktion stellt dabei eine Vielzahl an Parametern bereit, mit der Sie Ihre Grafik verändern können.
Anhand der Beispiele für die FE-Metaanalyse des `SMD_Data.txt`-Datensatzes und die RE-Metaanalyse des `B_Data.txt`-Datensatzes stellen wir Ihnen eine Auswahl an möglichen Veränderungen vor:

FE-Metaanalyse des `SMD_Data.txt`-Datensatzes Zusätzliche Angaben zur Darstellung der Gewichte und zur Begrenzung der Abbildung lassen sich direkt im Befehl `forest()` definieren:

```
# Fixed-Effect-Modell mit SMD
forest(res_smd.FE #rma-Objekt
       ,showweights = TRUE # Anzeige der relativen Gewichte
       ,digits=2 # Anzahl Nachkommastellen
       ,xlim=c(-2,3) # horizontale Begrenzung der Plot-Region
       ,alim=c(-1,1) # Minimum und Maximum der x-Achse
       ,ilab=cbind(round(calc_smd$vi,4)) # zusätzliche Spalte
       ,ilab.xpos=c(-1)) # Position der zusätzlichen Spalte
```

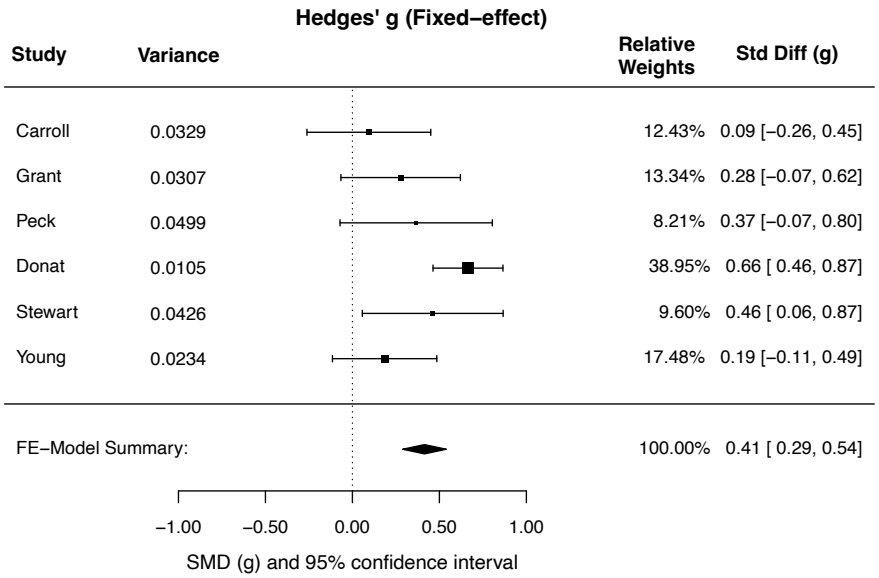


Weitere Ergänzungen sind beispielsweise bei der Beschriftung erforderlich. An dieser Stelle ist das metafor-Paket allerdings eher rudimentär ausgestattet. Während eine x-Achsen- und Summaryzeilen-Beschriftung als Parameter des Befehls `forest()` möglich ist, müssen die restlichen Einstellungen mit Hilfe der R-internen Grafikparameter vorgenommen werden.

Über die Funktion `par()` können Grundeinstellungen gesetzt werden. Dies betrifft z.B. Schrifttyp und -größe oder die Positionierung der Elemente innerhalb des Plots. Wenn die Parameter gesetzt wurden, können mit `text()` unter der Angabe von horizontaler und vertikaler Position zusätzliche Beschriftungen eingefügt werden (**Tipp**: mit `par("usr")[1:4]` erfahren Sie die aktuellen Einstellungen der Plot-Region in der Reihenfolge [xMin, xMax, yMin, yMax]).

```
# par: für Layout-Parameter
# mar: zum Setzen von Rand (margin)-Größen
#      c(unten, links, oben, rechts)
# cex: Schriftgröße
# font: definiert Schriftsatz
par(mar=c(6,4,1,2),cex=.9, font=1)

# Fixed-Effect Modell mit SMD
forest(res_smd.FE #rma-Objekt
       ,showweights = TRUE # Anzeige der Gewichte
       ,digits=2 # Anzahl Nachkommastellen
       ,xlim=c(-2,3) # horizontale Begrenzung der Plot-Region
       ,alim=c(-1,1) # Minimum und Maximum der x-Achse
       ,ilab=cbind(round(calc_smd$vi,4)) # zusätzliche Spalte
       ,ilab.xpos=c(-1) # Position der zusätzlichen Spalte
       ,xlab="SMD (g) and 95% confidence interval" #Plot-Bezeichng.
```

```
       ,mlab="FE-Model Summary:") # Summary-Bezeichnung

# Änderung der Grafikparameter für zusätzliche Bezeichnungen
# Spaltenbeschriftungen
par(font=2)
text(c(-1.80,-1,1.75,2.5), 7.7
       ,c("Study", "Variance", "Relative\nWeights"
       ,"Std Diff (g)"))
# Diagrammüberschrift
par(cex=1, font=2)
text(0.4,8.5,"Hedges' g (Fixed-effect)")
```

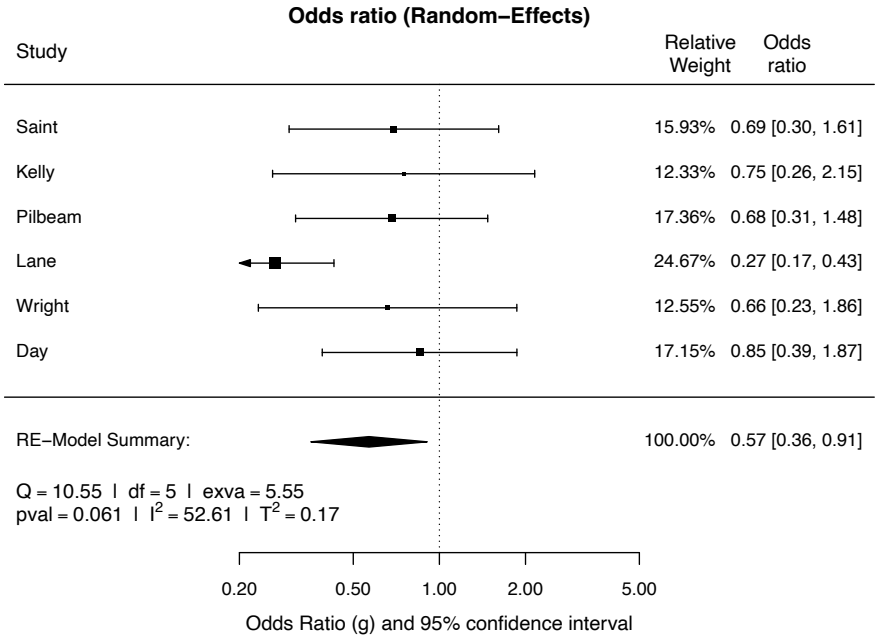


Sie haben nun einen vollständigen Forest-Plot für eine Fixed-Effect-Metaanalyse erstellt. Sie können nach Bedarf weitere Veränderungen an den Parametern vornehmen. Wenn Sie mit den Ergbnissen zufrieden sind, kann der Plot auch separat durch einen Klick auf *Export* im Plot-Reiter exportiert werden.

RE-Metaanalyse des `B_Data.txt`-Datensatzes Zusätzlich zum oben beschriebenen Output können Informationen wie T^2 oder I^2 mit `text()` im Diagramm mit angezeigt werden. Hilfreich dabei sind die Funktionen `bquote()` (Anzeigen der Werte von Variablen im Fließtext) und `round()` (Runden auf gewünschte Nachkommastelle).

```
# Grundeinstellung Grafikparameter
par(mar=c(6,4,1,2),cex=.9, font=1)

# Random-Effects Model mit binären Daten
```

```r
forest(res_b.RE,showweights = TRUE,digits=2
       ,xlim=c(-3.5,3.5)
       ,ylim=c(-3,9)
       ,refline=log(1)
       ,at=log(c(0.2,0.5,1,2,5)) # Einteilung x-Achse
       ,xlab="Odds Ratio (g) and 95% confidence interval"
       ,mlab="RE-Model Summary:"
       ,atransf=exp) # logarithm. Transformierung x-Achse

# Spaltenbeschriftungen
par(font=1)
text(c(-3.2,2.1,2.8), 7.7
       , c("Study","Relative\nWeight","Odds\nratio"))

# Extra-Informationen
text(-3.5,-2.2,bquote(Q == .(round(res_b.RE$QE,2)) ~ " | "
    ~ df == .(res_b.RE$k-1) ~ " | "
    ~ exva == .(round(res_b.RE$QE-(res_b.RE$k-1),2))), pos=4)
text(-3.5,-2.7,bquote(pval == .(round(res_b.RE$QEp,4)) ~ " | "
    ~ I^{2} == .(round(res_b.RE$I2,2)) ~ " | "
    ~ T^{2} == .(round(res_b.RE$tau2,2))), pos=4)

# Diagrammüberschrift
par(cex=1, font=2)
text(0,8.5,"Odds ratio (Random-Effects)")
```



Danksagung

Dieses Dokument wurde erstellt mit der Unterstützung von Markus Fuhrmann.

Übungsblatt identisch zu unserem!

Seminar 2
Einführung in die Statistik von Metaanalysen

Andrea Gottschalk, Ingmar Glauche
IMB, TU Dresden

11. Dezember 2021

Ziel des Seminars. Im ersten Seminar haben Sie die Durchführung einfacher Metaanalysen (Fixed-Effect(FE)- und Random-Effects(RE)-Modelle) mit dem `metafor`-Package unter R kennengelernt. Im heutigen Seminar möchten wir Ihnen die Möglichkeit geben, diese Kenntnisse weiter zu vertiefen. Wir konzentrieren uns dabei auf die Themenbereiche Publikationsbias, Subgruppenanalyse und Metaregression.

Für die exemplarische Durchführung wurden zwei Beispieldatensätze ausgewählt, an denen Sie die genannten Thematiken methodisch durchführen und die Ergebnisse interpretieren lernen.

1 Vorbereitung

Das `metafor`-Package ist zwar auf Ihrem Computer bereits installiert, muss aber mit jedem Neustart von RStudio erneut geladen werden:

```
library(metafor)
```

Für weitere Informationen rund um das `metafor`-Projekt kann jederzeit die Paket-Webseite `http://www.metafor-project.org/doku.php` oder die RStudio-interne Hilfeseite genutzt werden:

```
help(metafor)
```

2 Publikationsbias

Sie haben in der Vorlesung gehört, dass es gerade bei kleineren Studien eine Tendenz gibt, positive Ergebnisse häufiger zu publizieren als negative Ergebnisse. Der daraus resultierende Publikationsbias stellt für Metaanalysen eine wesentliche Fehlerquelle dar und sollte entsprechend adressiert werden.

2.1 Beipspiel eines Publikationsbias

Im Datensatz von *Hackshaw (1998)*[1] wird im Rahmen einer Metaanalyse die Fragestellung untersucht, ob Passivrauchen zu einem erhöhten Lungenkrebsrisiko führt.
Der Datensatz wird unter `dat.hackshaw1998` innerhalb des `metafor`-Pakets zur Verfügung gestellt. Informationen zum Datensatz, beispielsweise zur Art der betrachteten Studien und der berechneten Effektstärken, erhalten Sie mithilfe der `help()`-Funktion.

Berechnen Sie zu diesem Datensatz mit dem Befehl `rma` ein RE-Modell und speichern Sie die Ergebnisse im Objekt `res_hh.RE`.

```
res_hh.RE <- rma(...)
```

```
##
## Random-Effects Model (k = 37; tau^2 estimator: DL)
##
##   logLik  deviance       AIC       BIC      AICc
## -10.1755   43.2572   24.3509   27.5728   24.7039
##
## tau^2 (estimated amount of total heterogeneity): 0.0170 (SE = 0.0171)
## tau (square root of estimated tau^2 value):      0.1305
## I^2 (total heterogeneity / total variability):   24.21%
## H^2 (total variability / sampling variability):  1.32
##
## Test for Heterogeneity:
## Q(df = 36) = 47.4979, p-val = 0.0952
##
## Model Results:
##
## estimate      se    zval    pval   ci.lb   ci.ub
##   0.2139  0.0471  4.5390  <.0001  0.1215  0.3062  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

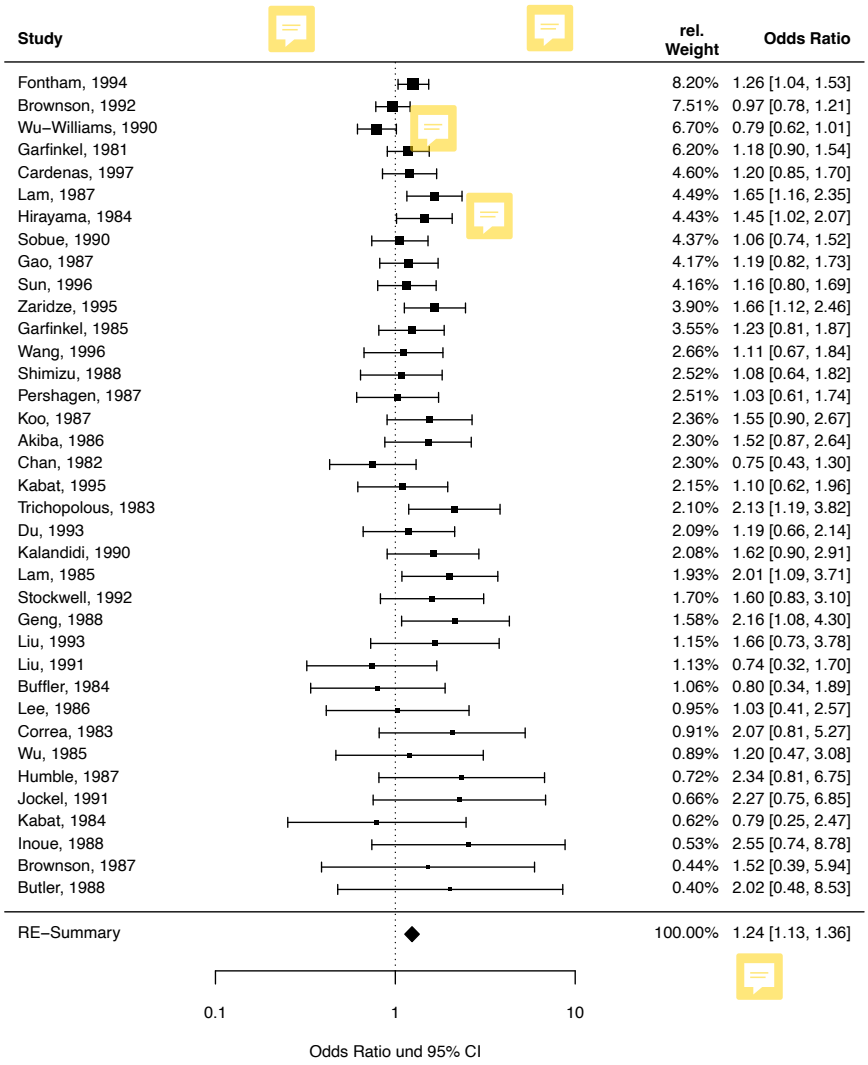
Interpretieren Sie das Ergebnis der Metaanalyse: Wird ein signifikanter Unterschied in der Gefährdung festgestellt? Wie bewerten Sie die Heterogenität der einzelnen Studien?

Wenn Sie den Datensatz korrekt geladen und das RE-Modell korrekt berechnet haben, kann der dazugehörige Forest-Plot wie folgt erstellt werden. Das Argument `order="prec"` hat dabei zur Folge, dass die Studien nach ihrem Gewicht geordnet werden.

```
forest(res_hh.RE,showweights = TRUE, cex=.75
       ,order="prec" #ordnet nach absteigendem rel. Gewicht
       ,xlim=c(-5,6)
```

[1] Hackshaw AK, Law MR, Wald NJ, The accumulated evidence on lung cancer and environmental tobacco smoke, BMJ 315(7114), 1997.

```
       ,ylim=c(-1,40)
       ,refline=log(c(1))
       ,at=log(c(0.1,1,10))
       ,xlab="Odds Ratio und 95% CI"
       ,mlab="RE-Summary"
       ,atransf=exp)

par(cex=.75,font=2)
text(-5,39,"Study",pos=4)
text(6,39,"Odds Ratio",pos=2)
text(3.8,39,"rel.\nWeight")
```



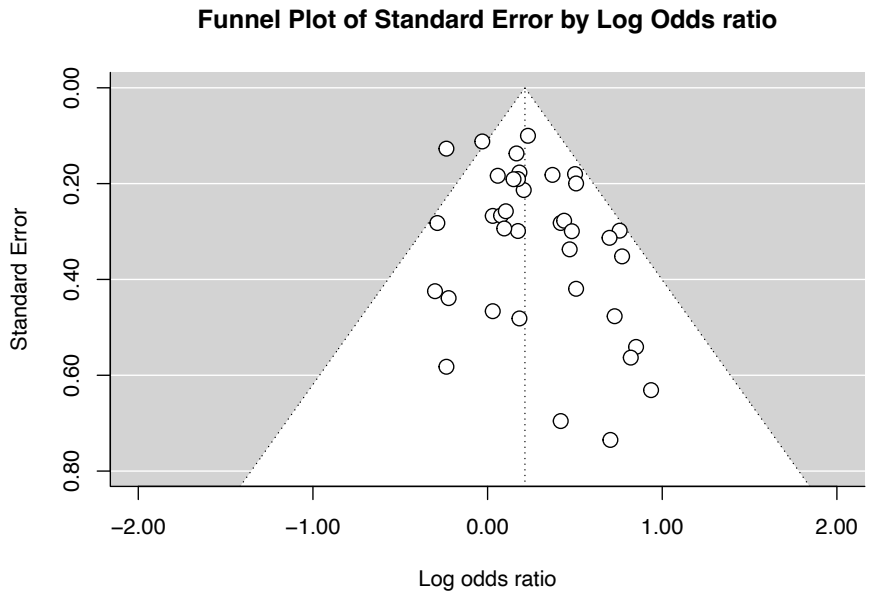
Bei genauerer Betrachtung fällt auf, dass der Großteil der kleineren Studien größere Effektstärken geschätzt hat, auch wenn diese Effekte selbst nicht signifikant sind. Um diese Hinweise auf einen Publikationsbias genauer zu adressieren, werden üblicherweise Funnel-Plots verwendet.

2.2 Erstellen eines Funnel-Plots

In einem Funnel-Plot wird die Effektstärke der einzelnen Studien auf der x-Achse gegen ein Varianzmaß, meist den Standardfehler, auf der y-Achse aufgetragen. Mit wachsender Stichprobengröße nimmt der Standardfehler ab. Das bedeutet, dass die Effektstärken mit abnehmenden Werten auf der y-Achse näher am wahren Wert der zugrunde liegenden Effektstärke liegen sollten und sich damit die bereits vorgestellte Trichterform ergibt. Im „unteren“ Teil des Diagramms werden kleinere Studien dargestellt. Ein mögliches Ungleichgewicht, hervorgerufen durch Veröffentlichungen mit stärkeren Effektstärken, kann so leichter identifiziert werden.

Ein Funnel-Plot wird mit der Funktion `funnel()` aufgerufen und kann genau wie ein Forest-Plot mit verschiedenen Parametern modifiziert werden:

```
funnel(res_hh.RE #rma-Object
       ,main="Funnel Plot of Standard Error by Log Odds ratio"
       ,pch=21 #Punkte-Typ
       ,xlab="Log odds ratio"
       ,xlim=c(-2,2) #Skalierung x-Achse
       ,ylab="Standard Error"
       ,ylim=c(0,0.8) #Skalierung y-Achse
       ,digits=2, cex=1.5)
```



2.3 Quantifizierung eines Publikationsbias

In der Vorlesung wurden Ihnen mit *Rosenthal's Fail-safe N* und *Trim and Fill* zwei Methoden erläutert, um einen Publikationsbias zu quantifizieren.

Rosenthal's Fail-Safe N: Bei dieser Methode wird davon ausgegangen, dass ein signifikanter p-Wert bezüglich des Summary-Effekts der Metaanalyse detektiert wurde. Rosenthal stellte 1979 die Frage, wieviele Studien man eventuell übersehen haben müsste, um die Signifikanz des gefundenen Ergebnisses in Frage zu stellen. Rosenthals Ansatz versucht nun, zu der bestehenden Metaanalyse solange Studien *ohne nachweisbaren Effekt* ($Y_i = 0$ für Mittelwertdifferenzen bzw. $Y_i = 1$ für OR/RR) hinzuzufügen, bis der Summary-Effekt *nicht mehr* signifikant ist. Die Anzahl der *künstlichen* Studien, die auf diese Art und Weise eingebunden werden kann, ist ein Indikator, inwiefern die Signifikanz des Gesamtergebnisses durch wenige Studien beeinflussbar ist.

Mit der `metafor`-Funktion `fsn()` kann die Methode angewandt werden:

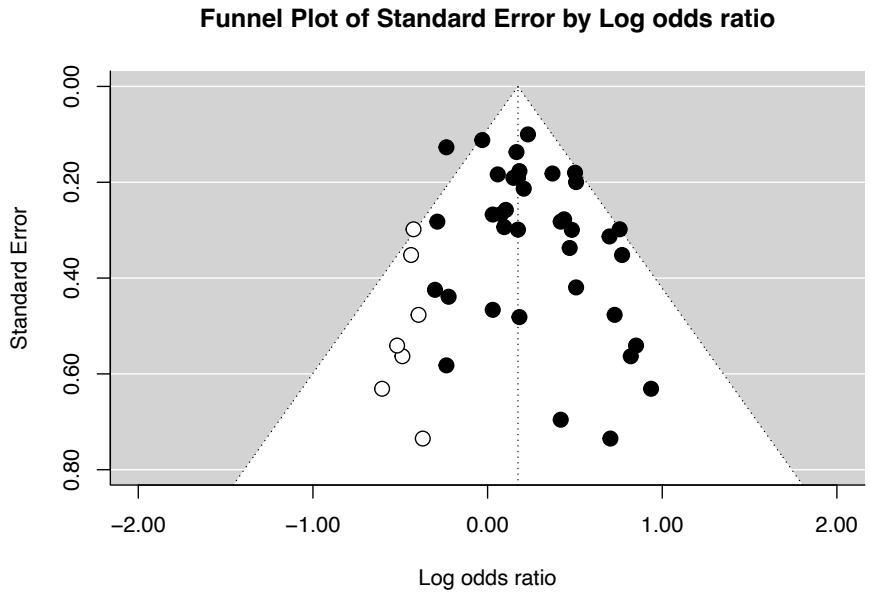
```
fsn(yi,vi, data=dat.hackshaw1998, type="Rosenthal")

##
## Fail-safe N Calculation Using the Rosenthal Approach
##
## Observed Significance Level: <.0001
## Target Significance Level:   0.05
##
## Fail-safe N: 393
```

Interpretieren Sie das Ergebnis der Analyse! Was sagt die Anzahl N aus?

Die Methode von Rosenthal hat wesentlich dazu beigetragen, die Problematik des Publikationsbias genauer zu erforschen. Allerdings ist die Methode nicht unumstritten und wird heutzutage kaum noch angewendet.

Trim and Fill: Bei dieser Methode wird eine symmetrische Verteilung der Effektstärke um einen wahren Wert angenommen. Um die Symmetrie herzustellen, werden zuerst iterativ kleine Studien aus der Metaanalyse entfernt, bis die Studien symmetrisch um die neue, unverzerrte Effektstärke verteilt sind. Dieses *trimming* reduziert allerdings auch die Varianz und resultiert in zu kleinen Konfidenzintervallen. Um diese Varianzschätzung wieder zu korrigieren, werden alle entfernten Studien mit ihren Spiegelbildern wieder in die Metaanalyse aufgenommen. Dadurch wird der Summary-Effekt nicht mehr verändert, aber der Varianzschätzer wieder verbessert.
Mit der Funktion `trimfill()` wird unter Angabe eines `rma`-Objektes die Kompensation des Publikationsbias berechnet. Die schwarzen Punkte stellen hier die schon vorhandenen und die weiß markierten die durch die *Trim and Fill*-Methode hinzugefügten Studien dar.

```r
taf <- trimfill(res_hh.RE)
funnel(taf #rma-Object
       ,main="Funnel Plot of Standard Error by Log odds ratio"
       ,xlab="Log odds ratio"
       ,xlim=c(-2,2) #Skalierung x-Achse
       ,ylab="Standard Error"
       ,ylim=c(0,0.8) #Skalierung y-Achse
       ,digits=2, cex=1.5)
```



Verlgeichen Sie die Variablen `taf` und `res_hh.RR` und bewerten Sie die Unterschiede.

Weitere Informationen zu alternativen Darstellungsformen der Funnel-Plots finden Sie am Ende dieser Seminaranleitung.

3 Subgruppenanalyse

In der Zusatzaufgabe der letzten Übung hatten wir Sie gebeten, den Datensatz von *Colditz et al.(1994)* mithilfe einer Metaanalyse auszuwerten. Dieser Datensatz enthält 13 Studien, in denen die Effektivität von Bacillus-Calmette-Guerin(BCG)-Impfungen zur Prävention von Tuberkulose untersucht wurde.

Die weiteren Auswertungen bauen auf diesem Datensatz auf. Führen Sie deshalb zunächst die Analyse dieser Daten erneut durch: Berechnen Sie mithilfe der `escalc()`-Funktion die jeweiligen Relativen Risiken und Varianzen der Einzelstudien. Führen Sie anschließend mit dem `rma()`-Befehl eine Random-Effects-Metaanalyse durch.

```r
# Lädt den BCG-Datensatz
data(dat.bcg)

# Berechnet yi (Relatives Risiko) und vi des BCG-Datensatzes
bcg_RR <- escalc(...)

# Schätzt ein Random-Effects-Modell
res_bcg.RE <- rma(...)
```

```
##
## Random-Effects Model (k = 13; tau^2 estimator: DL)
##
##   logLik  deviance       AIC       BIC      AICc
## -12.6823   37.1505   29.3647   30.4946   30.5647
##
## tau^2 (estimated amount of total heterogeneity): 0.3088 (SE = 0.2299)
## tau (square root of estimated tau^2 value):      0.5557
## I^2 (total heterogeneity / total variability):   92.12%
## H^2 (total variability / sampling variability):  12.69
##
## Test for Heterogeneity:
## Q(df = 12) = 152.2330, p-val < .0001
##
## Model Results:
##
## estimate      se     zval    pval    ci.lb    ci.ub
##  -0.7141  0.1787  -3.9952  <.0001  -1.0644  -0.3638  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretieren Sie das Ergebnis der Metaanalyse! Beachten Sie besonders die Aussagen zur Heterogenität der Ergebnisse.

Um die Ursachen der Heterogenität näher zu untersuchen, ist weiteres Vorwissen zu den Studien erforderlich. Für die BCG-Untersuchung ist diese Information schon in den zusätzlichen Spalten („alloc“, „ablat“) angegeben, welche Aussagen zum angewendeten Randomisierungsschema und zum Breitengrad der einzelnen Studien enthalten.

In der Vorlesung haben Sie drei Varianten kennengelernt, um Subgruppenanalysen durchzuführen. Neben einem Fixed-Effect-Ansatz existieren Random-Effects-Methoden entweder mit einem individuellem oder einem gemitteltem Schätzer für τ^2. Wir werden im Folgenden den Random-Effects-Ansatz vorstellen, bei dem τ^2 über alle Subgruppen geschätzt wird.

Random-Effects-Modell mit einem gemittelten Schätzer für τ^2

Im BCG-Datensatz befindet sich die Spalte „alloc“. Daraus können sie das Schema entnehmen, mit dem Patienten innerhalb der Studien in die Behandlungsgruppen zugewiesen wurden. Diese Spalte enthält einen *character*-Datentyp, welcher entweder den Wert „randomisiert“, „alternierend“ oder „systematisch“ trägt. Es soll untersucht werden, ob dieses Zuordungsschema einen Teil der beobachteten Heterogenität der Studien erklären kann.

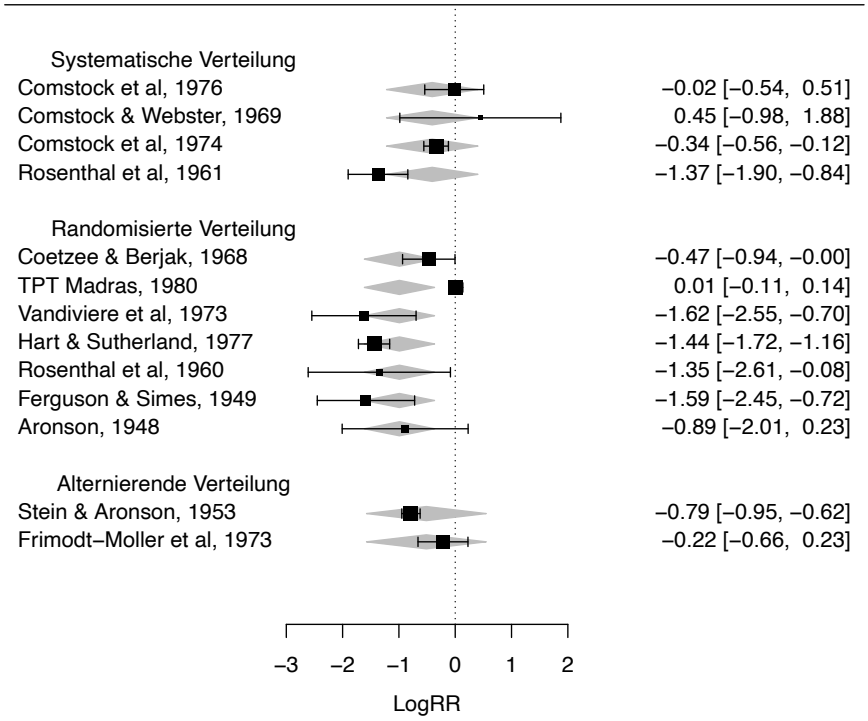
Innerhalb des `metafor`-Paketes wird die Gruppenzugehörigkeit am günstigsten als (kategoriale) Kovariable definiert. Dieser sogenannte Moderator wird innerhalb der `rma()`-Funktion über das Argument `mods` definiert.

```
# Berechnung des rma-Objekts
res_bcg.RE_alloc <- rma(yi, vi
                  ,mods = ~alloc # Angabe des Moderators
                  ,data = bcg_RR # Datenquelle
                  ,method="DL")
# Anzeigen des rma-Objekts
res_bcg.RE_alloc

##
## Mixed-Effects Model (k = 13; tau^2 estimator: DL)
##
## tau^2 (estimated amount of residual heterogeneity):     0.5596 (SE = 0.4045)
## tau (square root of estimated tau^2 value):             0.7480
## I^2 (residual heterogeneity / unaccounted variability): 92.45%
## H^2 (unaccounted variability / sampling variability):   13.24
## R^2 (amount of heterogeneity accounted for):            0.00%
##
## Test for Residual Heterogeneity:
## QE(df = 10) = 132.3676, p-val < .0001
##
## Test of Moderators (coefficients 2:3):
## QM(df = 2) = 1.4349, p-val = 0.4880
##
## Model Results:
##
##                  estimate      se     zval    pval    ci.lb   ci.ub
## intrcpt           -0.5125  0.5421  -0.9454  0.3444  -1.5751  0.5500
## allocrandom       -0.4780  0.6286  -0.7605  0.4470  -1.7099  0.7540
## allocsystematic    0.1042  0.6822   0.1528  0.8786  -1.2329  1.4414
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vergleichen Sie das Ergebnis mit der Metaanalyse des Datensatzes ohne Berücksichtigung der Subgruppen. Ist das Zuordnungsschema eine mögliche Ursache für die beobachtete Heterogenität?

Für die Visualisierung der Ergebnisse bietet sich wiederum ein Forest-Plot an. Allerdings sind einige Zusatzangaben nötig, um die Subgruppen entsprechend hervorzuheben:

```r
forest(res_bcg.RE_alloc
  # Ordnen nach Behandlungsschema
  ,order=order(dat.bcg$alloc)
  # vertikale Begrenzung der Plot Region
  ,ylim=c(1, 24)
  # vertikale Angabe der Position für die Subgruppen
  ,rows=c(3:4,7:13,16:19)
  # x-Achsen Beschriftung
  ,xlab="LogRR")

# Beschriftungen für die Subgruppen
text(-5 # horizontale Positionierung (linker Rand)
     ,c(20,14,5) # vertikale Positionierung
     ,c("Systematische Verteilung",
        "Randomisierte Verteilung",
        "Alternierende Verteilung"))
```



Für weitere Informationen zur Durchführung von Subgruppenanalysen verweisen wir beispielsweise auf die Publikation *Conducting meta-analyses in R with the metafor package* von Wolfgang Viechtbauer (Journal of Statistical Software, 2010, 36(3):1).

4 Metaregression

Die Metaregression stellt eine weitere Möglichkeit dar, den Einfluss von Kovariablen auf die Effektstärken der einzelnen Studien zu untersuchen und damit die beobachtete Heterogenität zu erklären. Im Gegensatz zur Subgruppenanalyse handelt es sich allerdings um metrische Kovariablen.

Im BCG-Datensatz ist in der Spalte „ablat" der geografische Breitengrad angegeben, an welchem die jeweilige Studie durchgeführt wurde. Diese Information steht hier stellvertretend für das Vorhandensein von nichtpathogenen Mykobakterien in der Umwelt, welche zu einer natürlichen Immunisierung gegen Tuberkulose beitragen können.

Im ersten Schritt gilt es zu beurteilen, ob der Breitengrad einen substantiellen Einfluss auf die Heterogenität der Studienergebnisse besitzt. Über das Argument `mods` in der `rma`-Funktion wird nun der Breitengrad als metrische Kovariable in der Metaanalyse berücksichtigt:

```
# Berechnung des rma-Objekts
res_bcg.RE_abl <- rma(yi, vi
                  ,mods = ~ablat # Angabe des Moderators
                  ,data = bcg_RR # Datenquelle
                  ,method="DL")
# Anzeigen des rma-Objekts
res_bcg.RE_abl

##
## Mixed-Effects Model (k = 13; tau^2 estimator: DL)
##
## tau^2 (estimated amount of residual heterogeneity):     0.0633 (SE = 0.0548)
## tau (square root of estimated tau^2 value):             0.2516
## I^2 (residual heterogeneity / unaccounted variability): 64.21%
## H^2 (unaccounted variability / sampling variability):   2.79
## R^2 (amount of heterogeneity accounted for):            79.50%
##
## Test for Residual Heterogeneity:
## QE(df = 11) = 30.7331, p-val = 0.0012
##
## Test of Moderators (coefficient 2):
## QM(df = 1) = 18.8452, p-val < .0001
##
## Model Results:
##
##          estimate      se     zval    pval    ci.lb    ci.ub
## intrcpt    0.2595  0.2323   1.1172  0.2639  -0.1958   0.7149
## ablat     -0.0292  0.0067  -4.3411  <.0001  -0.0424  -0.0160  ***
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

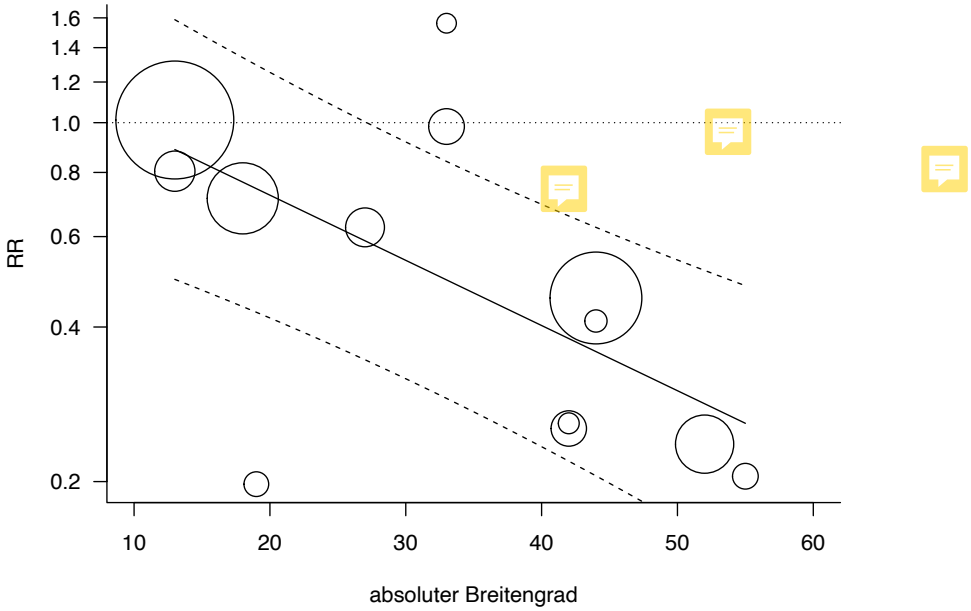
Interpretieren Sie die mithilfe des Modells geschätzten Koeffizienten. Gibt es einen signifikanten Einfluss des Breitengrades auf das Relative Risiko?

Für die grafische Darstellung dieses Zusammenhangs werden noch zwei Tricks verwendet:

- Um die Gewichte der einzelnen Studien besser abzubilden, wird die Größe der Punkte an das Inverse der Varianz gekoppelt.
- Mit der Funktion `predict()` kann für jeden Breitengrad eine Vorhersage zum erwarteten Relativen Risiko gemacht werden. Zusätzlich zu diesem vorhergesagten Mittelwert werden zwei Arten von Intervallen berechnet, die Sie auch schon in den Übungen zur Linearen Regression am Anfang des Semesters kennengelernt haben: (i) `ci.lb` und `ci.ub` geben Ihnen die untere bzw. obere Schranke des *95%-Konfidenzintervalls*, welches Ihnen die Unsicherheit in der Mittelwertschätzung beschreibt. (ii) Durch `cr.lb` und `cr.ub` wird hingegen ein *Vorhersageintervall* definiert, welches Ihnen einen Bereich angibt, in dem Sie die Werte von 95% aller neuen Studien erwarten. Wenn man diese Intervalle für alle Breitengrade verbindet, entsteht ein Konfidenz- bzw. Vorhersageband für den geschätzten linearen Zusammenhang.

Schauen Sie sich den R-Code für die grafische Ausgabe an. Wo werden die entsprechenden Elemente zur Punktgröße und zur Vorhersage definiert? Welches Intervall wird verwendet und warum?

```r
# Einstellung der Grafikoptionen
par(mar=c(5,5,1,2))

# Sch?tzt das Relative Risiko der Breitengrade von 13-55
preds <- predict(res_bcg.RE_abl, newmods=c(13:55), transf=exp)

# Berechnung für Punktgröße
# Relative Risiken = proportional zum inversen Standardfehler
wi <- 1/sqrt(bcg_RR$vi)
size <- 2 + 10 * (wi - min(wi))/(max(wi) - min(wi))

# grafische Darstellung: ln(RR) gegen Breitengrad
plot(bcg_RR$ablat,exp(bcg_RR$yi) # X- und Y-Achse
     ,pch=21 # Punktform
     ,cex=size # Punktgröße
     ,xlab="absoluter Breitengrad", ylab="RR"
     ,las=1 # Positionierung der Achsenbezeichnungen
     ,bty="l" # Diagrammumrandung
     ,xlim=c(10,60) # Skalierung x-Achse
     ,log="y") # Logarithmierung der Y-Achse

# geschätzte Werte + Vorhersageintervalle
# lty="dashed" = gestrichelte Linie
lines(13:55, preds$pred)
lines(13:55, preds$cr.lb, lty="dashed")
lines(13:55, preds$cr.ub, lty="dashed")

# zeichnet gepunktete Linie bei Y=1
abline(h = 1, lty = "dotted")
```



Diese Grafik ist eine übliche Darstellung für Ergebnisse einer Metaregression. Vielleicht haben Sie ja irgendwann ein „glückliches Händchen“ und identifzieren selbst einen solchen unerwarteten Zusammenhang.

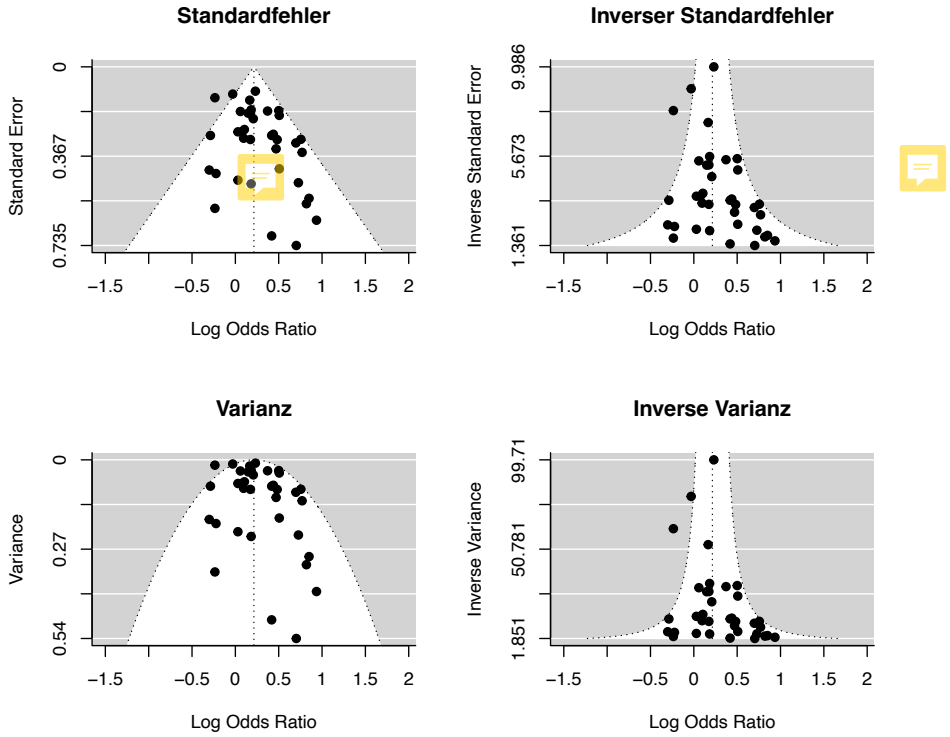
5 Zusatz: Variationen des Funnel-Plots

Bisher wurde in den benutzten Funnel-Plots nur der Standardfehler auf der y-Achse aufgetragen. Für einige Datensätze ist es jedoch sinnvoll, andere Einflussgrößen für eine bessere Verständlichkeit des Sachverhaltes zu verwenden. Mit dem Parameter `yaxis` können verschiedene Optionen wie folgt angegeben werden:

Parameter `yaxis =`	**Beschreibung**
keine Angabe	Standardfehler
`"seinv"`	inverser Standardfehler
`"vi"`	Varianz
`"vinv"`	inverse Varianz
`"ni"`	Stichprobengröße
`"ninv"`	inverse Stichprobengröße
`"sqrtni"`	Quadratwurzel der Stichprobengröße
`"sqrtninv"`	inverse Quadratwurzel der Stichprobengröße
`"lni"`	logarithmierte Stichprobengröße
`"wi"`	Gewichtung

Um ein Gefühl für die verschiedenen Darstellungsoptionen zu bekommen, sind im Anschluss die ersten 4 Funnel-Plots exemplarisch aufgeführt:

```
# Setzen der Grafikparameter auf ein 2x2 Array
par(mfrow=c(2,2))

# Erstellen Funnel-Plots
funnel(res_hh.RE, main="Standardfehler")
funnel(res_hh.RE, yaxis="seinv", main="Inverser Standardfehler")
funnel(res_hh.RE, yaxis="vi", main="Varianz")
funnel(res_hh.RE, yaxis="vinv", main="Inverse Varianz")
```



Danksagung

Dieses Dokument wurde erstellt mit der Unterstützung von Markus Fuhrmann.