1 Regressionsmodelle

Lineare Regression

Die lineare Regression wird an einem Beispiel auf Fields Discovering Statistisc using R[1] dargestellt, bei dem ein Plattenverlag anhand des Werbebudgets (in Tausend €) die Verkaufszahlen (in Tausend) vorhersagen möchte. Laden Sie zunächst den Datensatz.

```
> rsales <-
read.delim("http://studysites.uk.sagepub.com/dsur/study/
DSUR%20Data%20Files/Chapter%207/Album%20Sales%202.dat")
```

(Vielleicht müssen Sie "%20" durch Leerzeichen ersetzen, damit der Download gelingt)

Schauen Sie sich die Daten erst einmal an.

Daten anschauen

```
> summary(rsales)
    adverts            sales          airplay
Min.   :   9.104   Min.   : 10.0   Min.   : 0.00
1st Qu.: 215.918   1st Qu.:137.5   1st Qu.:19.75
Median : 531.916   Median :200.0   Median :28.00
Mean   : 614.412   Mean   :193.2   Mean   :27.50
3rd Qu.: 911.226   3rd Qu.:250.0   3rd Qu.:36.00
Max.   :2271.860   Max.   :360.0   Max.   :63.00
    attract
Min.   : 1.00
1st Qu.: 6.00
Median : 7.00
Mean   : 6.77
3rd Qu.: 8.00
Max.   :10.00
```

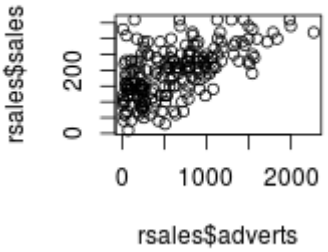

Abbildung 1 : Plattenverkaufszahlen pro Werbebudget (in Tausend), Quelle: Field 2012

Die Verkaufszahlen sind in der Variable *sales* gespeichert und reichen von 10.000 – 360.000 mit einem Mittelwert von 193,200. Die Werbeausgaben sind in *adverts* abgelegt (9.140 – 2.2271.860, Mittelwert 614.412).

Schauen Sie sich auch ein Steudiagramm der beiden Variablen an.

```
> plot(rsales$adverts, rsales$sales)
```

Ein lineares Regressionsmodell wir in R mit der Funktion `lm()` nachdem muster `lm(AbhängigeVariable ~ UnabhängigeVariable(n), data = Daten)` berechnet.

Lineares Modell

`lm()`

```
> rs1 <- lm(sales ~ adverts, data = rsales)
> rs1

Call:
lm(formula = sales ~ adverts, data = rsales)
```

1 Andy P Field, Jeremy Miles, und Zoë Field, *Discovering Statistics Using R* (London; Thousand Oaks, Calif.: Sage, 2012).

```
Coefficients:
(Intercept)      adverts
  134.13994       0.09612
```

summary(lm())

Um mehr als die reinen Koeffizienten angezeigt zu bekommen, nutzen Sie `summary()`.

```
> summary(rs1)

Call:
lm(formula = sales ~ adverts, data = rsales)

Residuals:
     Min       1Q   Median       3Q      Max
-152.949  -43.796   -0.393   37.040  211.866

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.341e+02  7.537e+00  17.799   <2e-16 ***
adverts     9.612e-02  9.632e-03   9.979   <2e-16 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 65.99 on 198 degrees of freedom
Multiple R-squared:  0.3346, Adjusted R-squared:  0.3313
F-statistic: 99.59 on 1 and 198 DF,  p-value: < 2.2e-16
```

summary.aov()

Wenn Sie eine ANOVA-Tabelle angezeigt bekommen möchten verwenden Sie `summary.aov()`.

```
> summary.aov(rs1)
             Df Sum Sq Mean Sq F value Pr(>F)
adverts       1 433688  433688   99.59 <2e-16 ***
Residuals   198 862264    4355
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Elemente eines *lm*-Objektes abrufen

lm()$Bezeichnung

lm()[Index]

BEVOR SIE WEITER MACHEN SCHAUEN SIE sich einmal an, was ein Objekt der Klasse *lm* alles enthält. Die folgende Tabelle zeigt die wichtigsten Inhalte.

```
> str(rs1)
```

Sie können diese Inhalte mit dem Muster `lm()$Bezeichnung` oder mit einer Indexnummer in eckigen Klammern `lm()[Index]` abrufen, z.B. die Koeffizienten.

```
> rs1$coefficients
(Intercept)      adverts
134.13993781   0.09612449
> rs1[1]
$coefficients
(Intercept)      adverts
134.13993781   0.09612449
```

Die folgende Tabelle zeigt die wichtigsten Inhalte eines *lm*-Objektes.

Tabelle 1: Die wichtigsten Inhalte eines lm-Objektes

Bezeichnung	Inhalt
$coefficients	Koeffizienten
$residuals	Residuen
$fitted.values	Geschätzte Werte
$df.residual	Residuale Freiheitsgrade
$call	Modellformel
$model	Die im Modell verwendeten Daten

Konfidenzintervalle mit `confint()`

KONFIDENZINTERVALLE KÖNNEN SIE mit `confint()` abrufen.

```
> confint(rs1)
                  2.5 %      97.5 %
(Intercept) 119.27768082 149.0021948
adverts       0.07712929   0.1151197
```

Standardisierte Betas mit `scale()`

UM STANDARDISIERTE BETAS ZU BERECHNEN, können Sie die Ausgangsvariablen mit `scale()` standardisieren.

```
> lm(scale(sales) ~ scale(adverts), data = rsales)

Call:
lm(formula = scale(sales) ~ scale(adverts), data =
rsales)

Coefficients:
   (Intercept)  scale(adverts)
    -2.141e-17       5.785e-01
```

Unabhängige Variablen mit + hinzufügen.

UM EIN MULTIVARIATES MODELL ZU BERECHNEN, fügen sie die Variablen `airplay` für die Anzahl der im Radio gespielten Stücke der Platte und `attract` für die Attraktivität der Band jeweils mit + hinzu.

```
> rs2 <- lm(sales ~ adverts + airplay + attract,
+     data = rsales)
> summary(rs2)

Call:
lm(formula = sales ~ adverts + airplay + attract, data =
rsales)

Residuals:
     Min       1Q   Median       3Q      Max
-121.324  -28.336   -0.451   28.967  144.132
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -26.612958  17.350001  -1.534    0.127
adverts       0.084885   0.006923  12.261  < 2e-16
airplay       3.367425   0.277771  12.123  < 2e-16
attract      11.086335   2.437849   4.548 9.49e-06

(Intercept)
adverts     ***
airplay     ***
attract     ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 47.09 on 196 degrees of freedom
Multiple R-squared:  0.6647, Adjusted R-squared:  0.6595
F-statistic: 129.5 on 3 and 196 DF,  p-value: < 2.2e-16
```

`update()`

Ein bequemerer Weg, Modelle zu modifizieren ist die Funktion `update()`. Dabei geben Sie als erstes das Ausgangsmodell an und dann mit + Variablen, die sie hinzufügen möchten oder mit − Variablen, die Sie entfernen möchten.

```
> rs2Upd <- update(rs1, . ~ . + airplay + attract)
```

Beachten Sie die Punkte vor und nach dem Tilde. Sie sorgen dafür, dass die im Ausgangsmodell enthaltenen Variablen (*sales* und *adverts*) im neuen Model erhalten bleiben. Im Ergebnis sollte das Modell identisch mit rs2 sein. Überprüfen Sie das.

Interaktion mit `*` und `:`

INTERAKTIONEN DEFINIEREN SIE mit `*`. Dabei wird automatisch der Haupteffekt und die Interaktion berechnen. Wenn Sie einmal nur die Interaktion ohne den Haupteffekt bestimmen wollen, nutzen Sie `:`.

```
> rs3 <- lm(sales ~ adverts * airplay * attract,
+     data = rsales)
> summary(rs3)

Call:
lm(formula = sales ~ adverts * airplay * attract, data =
rsales)

Residuals:
     Min       1Q   Median       3Q      Max
-143.035  -29.574   -0.119   28.310  134.402

Coefficients:
                         Estimate Std. Error t value
(Intercept)            59.6901823 70.2726834   0.849
adverts                -0.0294684  0.1448689  -0.203
airplay                -0.2386319  2.4454731  -0.098
attract                -2.5724900  9.9980800  -0.257
adverts:airplay         0.0045967  0.0045913   1.001
adverts:attract         0.0190848  0.0206018   0.926
airplay:attract         0.5568994  0.3419253   1.629
```

```
adverts:airplay:attract -0.0007493  0.0006550  -1.144
                        Pr(>|t|)
(Intercept)                0.397
adverts                    0.839
airplay                    0.922
attract                    0.797
adverts:airplay            0.318
adverts:attract            0.355
airplay:attract            0.105
adverts:airplay:attract    0.254

Residual standard error: 47.1 on 192 degrees of freedom
Multiple R-squared:  0.6713, Adjusted R-squared:  0.6593
F-statistic: 56.02 on 7 and 192 DF,  p-value: < 2.2e-16
```

Entfernen Sie jetzt z.B. die nicht-signifikante Dreifachinteraktion:

```
> rs4 <- update(rs3, ~. - adverts:airplay:attract)
> summary(rs4)

Call:
lm(formula = sales ~ adverts + airplay + attract +
adverts:airplay +
    adverts:attract + airplay:attract, data = rsales)

Residuals:
     Min       1Q   Median       3Q      Max
-136.659  -28.760    0.203   28.373  136.396

Coefficients:
                  Estimate Std. Error t value
(Intercept)     -3.0687476 43.9502551  -0.070
adverts          0.1288758  0.0427867   3.012
airplay          1.8643055  1.6140641   1.155
attract          6.3428001  6.2680601   1.012
adverts:airplay -0.0005944  0.0006993  -0.850
adverts:attract -0.0037088  0.0052412  -0.708
airplay:attract  0.2583977  0.2211413   1.168
                Pr(>|t|)
(Intercept)      0.94441
adverts          0.00294 **
airplay          0.24950
attract          0.31284
adverts:airplay  0.39637
adverts:attract  0.48004
airplay:attract  0.24406
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 47.14 on 193 degrees of freedom
Multiple R-squared:  0.6691, Adjusted R-squared:  0.6588
F-statistic: 65.04 on 6 and 193 DF,  p-value: < 2.2e-16
```

Modelle vergleichen mit `anova()`

ZWEI ODER MEHRERE MODELLE KÖNNEN SIE mit der Funktion `anova()` vergleichen.

```
> anova(rs4, rs3)
Analysis of Variance Table
```

```
Model 1: sales ~ adverts + airplay + attract +
adverts:airplay + adverts:attract +
    airplay:attract
Model 2: sales ~ adverts * airplay * attract
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    193 428843
2    192 425939  1    2903.2 1.3087 0.2541
```

Wenn Sie statt des F-Tests liebe einen Likelihood-Ratio-Test berechnen möchten ergänzen Sie das Argument test = "LRT".

```
> anova(rs4, rs3, test = "LRT")
Analysis of Variance Table

Model 1: sales ~ adverts + airplay + attract +
adverts:airplay + adverts:attract +
    airplay:attract
Model 2: sales ~ adverts * airplay * attract
  Res.Df    RSS Df Sum of Sq Pr(>Chi)
1    193 428843
2    192 425939  1    2903.2   0.2526
```

`AIC()`

Das Akaike Information Criterion für ein oder mehrere Modelle berechnen Sie mit `AIC()`.

```
> AIC(rs4, rs3)
    df      AIC
rs4  8 2117.681
rs3  9 2118.322
```

`drop1()`

Eine wunderbare Bequemlichkeitsfunktion ist drop1(), bei der die oben Genannten Statistiken jeweils für alle Modelle berechnet wird, bei denen einer der Ausgangsparameter entfernt wird.

```
> drop1(rs4, test = "Chisq")
Single term deletions

Model:
sales ~ adverts + airplay + attract + adverts:airplay +
adverts:attract +
    airplay:attract
                Df Sum of Sq    RSS    AIC Pr(>Chi)
<none>                      428843 1548.1
adverts:airplay  1    1605.5 430448 1546.8   0.3873
adverts:attract  1    1112.6 429955 1546.6   0.4716
airplay:attract  1    3033.7 431876 1547.5   0.2351
```

(Der Chi²-Test ist hier identisch mit dem Likelihood-Ratio-Test.)

Modelldiagnostik

Grafische Prüfung mit `plot()`

Die einfachste Art, sich einen Überblick über Modellannahmen sowie Ausreißer und Einflussreiche Werte zu verschaffen ist `plot()`.

```
> plot(rs2)
```

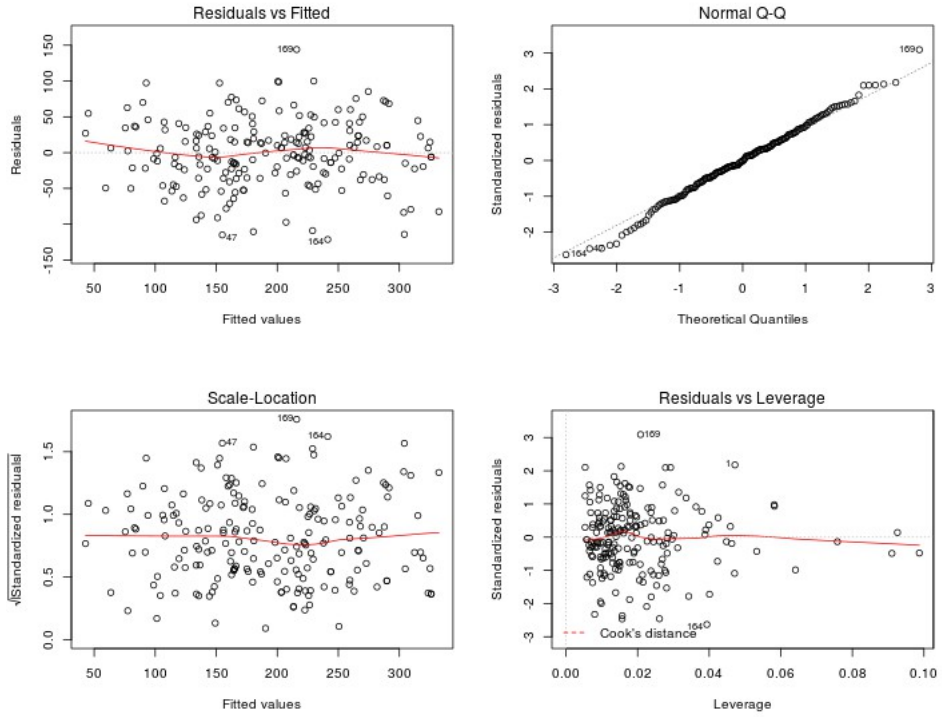

Abbildung 2 : Grafische Prüfung der Modellannahmen mit plot(lm())

Der erste Plot zeigt dir Residuen im Verhältnis zu den geschätzten Werten. Eine gleichmäßig verteilte Punktwolke zeigt Varianzhomogenität und Linearität des Zusammenhangs an. Die in diesem Fall fast gerade rot eingezeichnete Loess-Linie unterstreicht die Linearität.

Dem folgt ein QQ-Plot, mit dem Sie die Normalverteilung der Residuen prüfen können.

Als drittes wird die Wurzel der absoluten standardisierten Residuen gegenüber den geschätzten Werden angezeigt. Mit diesem Plot sollen bestimmte Formen der Varianzinhomogenität besser entdeckt werden können.

Schließlich werden die standardisierten Residuen gegenüber den Hebelwerten dargestellt, um Ausreißer und einflussreiche Werte zu identifizieren. Zusätzlich werden die Grenzbereiche von Cook's Distance = 0,5 und 1 eingezeichnet. (In diesem Fall liegen sie aber außerhalb des angezeigten Bereiches.)

Sie können sich die Residuen auch noch einmal mit `residuals()` abrufen. Standardisierte Residuen erhalten Sie mit `rstandard()` und studentisierte Residuen mit `rstudent()`.

Die Einflusswerte Leverage (hat), DFBeta (dfb. …), DFFit (dffit),

`residuals()`

`rstandard()`

`rstudent()`

`influence.measures()`

Cook's Distance (cook.d) und Kovarianzratio (cov.r) können Sie sich mit `influence.measures()` ausgeben lassen.

Linearität prüfen

`scatter.smooth()`

DIE LINEARITÄT DES ZUSAMMENHANGS KÖNNEN SIE für den bivariaten Fall mit `scatter.smooth()` prüfen.

```
scatter.smooth(rsales$sales ~ rsales$adverts)
abline(rs1, col = "red")
```

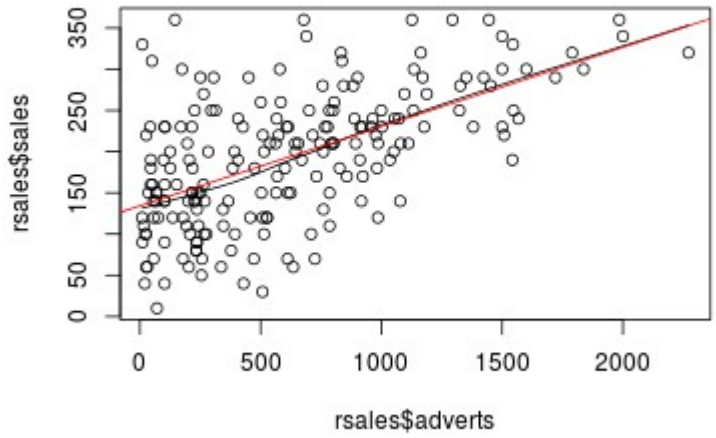

Abbildung 3 : Bivariate Prüfung auf Linearität des Zusammenhangs mit scatter.smooth()

Mit `abline(rs1, col = "red")` wurde hier noch die Regressionsgerade des (bivariaten) Modells in rot ergänzt.

Multivariat können Sie den oben gezeigten Plot der Residuen vs. geschätzter Werte betrachten.

Fractional Polynomials mit dem Paket *mfp*

Die numerische Beurteilung Bewertung der Linearität können Sie am besten mit *Fractional Polynomials* vornehmen. Hierfür können Sie das Paket mfp (für Multivariable Fractional Polynomials) verwenden.

```
> library(mfp)
> mfp(sales ~ fp(adverts, df = 4) + fp(airplay,
+     df = 4, select = 0.05) + fp(attract,
+     df = 4, select = 0.05), data = rsales)
Call:
mfp(formula = sales ~ fp(adverts, df = 4) + fp(airplay,
df = 4,
    select = 0.05) + fp(attract, df = 4, select = 0.05),
data = rsales)


Deviance table:
Resid. Dev
Null model 1295952
Linear model 434574.6
Final model 434574.6

Fractional polynomials:
        df.initial select alpha df.final power1
adverts          4   1.00  0.05        1      1
airplay          4   0.05  0.05        1      1
attract          4   0.05  0.05        1      1
        power2
adverts      .
airplay      .
attract      .


Transformations of covariates:
                      formula
adverts   I((adverts/1000)^1)
airplay I(((airplay+1)/10)^1)
attract     I((attract/10)^1)

Re-Scaling:
Non-positive values in some of the covariates. No re-
scaling was performed.

Coefficients:
Intercept  adverts.1  airplay.1  attract.1
   -29.98      84.88      33.67     110.86

Degrees of Freedom: 199 Total (i.e. Null);  196 Residual
Null Deviance:     1296000
Residual Deviance: 434600 AIC: 2114
```

In diesem Beispiel sind die besten fraktionierten Polynome Transformationen mit dem Exponenten 1 ("^1"), also die nicht-transponierten Daten.

Prüfung auf Normalverteilung der Residuen

DIE NORMALVERTEILUNG DER RESIDUEN wurde grafisch schon mit `plot()` geprüft. Sie können auch einen Shapiro-Wilk-Test auf die Residuen anwenden.

```
> shapiro.test(rs2$residuals)
```

```
Shapiro-Wilk normality test

data:  rs2$residuals
W = 0.99483, p-value = 0.7253
```

Prüfung auf Varianzhomogenität

AUCH DIE VARIANZHOMOGENITÄT wurde grafisch schon mit `plot()` untersucht. Sie können aber auch einen Levene-Test auf die Gruppierten Residuen anwenden. Hierzu werden zunächst die geschätzten Werte in Quartilen gruppiert, dann das Paket *car* geladen und dann der `leveneTest()` mit den Residuen, und den gruppierten geschätzten Werten durchgeführt.

```
> gruppe <- cut(rs2$fitted.values, breaks =
quantile(rs2$fitted.values))
> library(car)
> leveneTest(rs2$residuals ~ gruppe)
Levene's Test for Homogeneity of Variance (center =
median)
       Df F value Pr(>F)
group   3  0.2177  0.884
      195
```

Multikolinearität mit `cor()`

MULTIKOLINERITÄT KÖNNEN SIE BIVARIAT einschätzen, indem Sie sich eine Korrelationsmatrix anzeigen lassen. Das funktioniert, indem Sie `cor()` auf einen ganzen Datensatz anwenden. Damit nur die Variablen einbezogen werden, die Sie im Modell verwenden, können Sie den Datensatz mit `$model` aus dem *lm*-Objekt extrahieren.

```
> cor(rs2$model)
            sales    adverts   airplay    attract
sales   1.0000000 0.57848774 0.5989188 0.32611105
adverts 0.5784877 1.00000000 0.1018828 0.08075151
airplay 0.5989188 0.10188281 1.0000000 0.18198863
attract 0.3261111 0.08075151 0.1819886 1.00000000
```

Streudiagramm-Matrix mit `plot()`

Analog können sie `plot(rs2$model)` eine Streudiagramm-Matrix erzeugen.

Variance Inflation Factor `car::vif()`

Den *Variance Inflation Factor* können Sie mit `vif()` aus dem *car*-Paket berechnen.

```
> library(car)
> vif(rs2)
adverts  airplay  attract
1.014593 1.042504 1.038455
```

Autokorrelation prüfen mit `durbinWatsonTest()`

AUTOKORRELATION KÖNNEN SIE MIT DEM DURBIN-WATSON-TEST Prüfen. Sie finden ihn auch im *car*-Paket als `durbinWatsonTest()`. (Im Gegensatz zu SPSS wird hier auch ein p-Wert berechnet.)

```
> library(car)
```

```
> durbinWatsonTest(rs2)
lag Autocorrelation D-W Statistic p-value
   1       0.0026951      1.949819   0.752
Alternative hypothesis: rho != 0
```

ANOVA

Eine Varianzanalyse wird in R genau wie die lineare Regression mit `lm()` durchgeführt. Der einzige Unterschied ist, das die unabhängige Variable im Datenformat *factor* vorliegt.

Schauen Sie sich das am Beispiel des Datensatzes lowbwt[2] an.

```
> lowbwt <-
read.table("http://www.umass.edu/statdata/statdata/data/
lowbwt.dat",
+     header = TRUE, skip = 5)
> lowbwt$RACE <- factor(lowbwt$RACE)
> boxplot(lowbwt$BWT ~ lowbwt$RACE)
> tapply(lowbwt$BWT, lowbwt$RACE, mean)
       1        2        3
3103.740 2719.692 2804.015
```

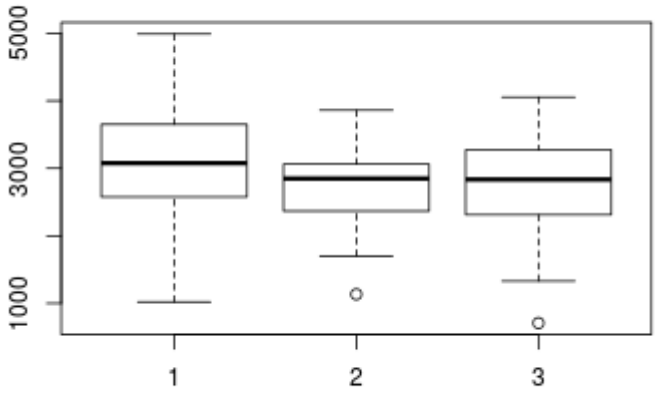

Abbildung 4 : Verteilung des Geburtsgewichtes nach ethnischer Zugehörigkeit, Quelle: Hosmer et al. 2013

```
> lwtAnov1 <- lm(BWT ~ RACE, data = lowbwt)
```

Modellierung mit `lm()`

Genau wie die lineare Regression können Sie sich auch dieses Modell als ANOVA-Tabelle oder als Zusammenfassung im Regressions-Stil anzeigen lassen.

`summary.aov(), summary()`

```
> summary.aov(lwtAnov1)
```

2 David W. Hosmer, Stanley Lemeshow, und Rodney X. Sturdivant, *Applied logistic regression*, 3. Aufl., Wiley series in probability and statistics (Hoboken, NJ: Wiley, 2013).

```
             Df   Sum Sq Mean Sq F value  Pr(>F)
RACE          2  5070608 2535304   4.972 0.00788 **
Residuals   186 94846445  509927
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(lwtAnov1)

Call:
lm(formula = BWT ~ RACE, data = lowbwt)

Residuals:
     Min       1Q   Median       3Q      Max
-2095.01  -503.01   -13.74   526.99  1886.26

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3103.74      72.88  42.586  < 2e-16 ***
RACE2        -384.05     157.87  -2.433  0.01594 *
RACE3        -299.72     113.68  -2.637  0.00908 **
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 714.1 on 186 degrees of freedom
Multiple R-squared:  0.05075, Adjusted R-squared:
0.04054
F-statistic: 4.972 on 2 and 186 DF,  p-value: 0.007879
```

Kontraste

Wenn Sie sich die Tabelle der Koeffizienten ansehen, bemerken Sie, dass R automatisch Dummi-Variablen gebildet hat, wobei das erste Level als Referenzkategorie verwendet wird.

Diese Dummi-Kodierung ist innerhalb jedes *factor*-Objektes abgespeichert. Sie können sie sich mit `contrasts()` anzeigen lassen.

`contrasts()`

```
> contrasts(lowbwt$RACE)
  2 3
1 0 0
2 1 0
3 0 1
```

R hat einige vorgefertigte Kontraste. Hier sind die wichtigsten.

Funktion	Beschreibung
`contr.treatment(n, base = 1, ...)`	Dummi-Kontraste: `n` steht für die Anzahl der Kategorien, mit `base =` wird angegeben, welche Kategorie die Referenzkategorie ist
`contr.helmert(n, …)`	Helmert-Kontraste: Vergleicht die erste Kategorie mit der zweiten, dann die dritte mit dem Mittelwert der beiden ersten u.s.w.
`contr.poly(n, scores = 1:n, ...)`	Orthogonale polynomiale Kontraste: n steht wieder für die Anzahl der Kategorien, scores = bezeichnet die Exponenten (x, x^2, x^3 ...)

Tabelle 2: Wichtige vordefinierte Kontraste

Kontraste ändern

Man könnte z.B. die Dummi-Kontraste so ändern, dass die letzte Kategorie als Referenzkategorie definiert wird.

```
> contr.treatment(3, base = 3)
  1 2
1 1 0
2 0 1
3 0 0
```

Wenn Sie diese geänderten Kontraste dem *factor*-Objekt zuweisen, wird es sich bei der Modellierung anders verhalten.

```
> contrasts(lowbwt$RACE) <- contr.treatment(3,
+     base = 3)
> lwtAnov2 <- lm(BWT ~ RACE, data = lowbwt)
> summary(lwtAnov2)

Call:
lm(formula = BWT ~ RACE, data = lowbwt)

Residuals:
     Min       1Q   Median       3Q      Max
-2095.01  -503.01   -13.74   526.99  1886.26

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2804.01      87.24  32.141  < 2e-16 ***
RACE1         299.72     113.68   2.637  0.00908 **
RACE2         -84.32     165.00  -0.511  0.60991
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 714.1 on 186 degrees of freedom
Multiple R-squared:  0.05075, Adjusted R-squared:
0.04054
F-statistic: 4.972 on 2 and 186 DF,  p-value: 0.007879
```

Für unser Beispiel, bei dem 1 für "white", 2 für "black" und 3 für "other" steht, sind die ursprünglichen Kontraste aber inhaltlich sinnvoller.

Interaktionen

Wenn Sie Interaktionen modellieren, können Sie sie mit interaction.plot() übersichtlich anzeigen lassen und mit tapply() die einzelnen Mittelwerte berechnen

```
> lwtAnov3 <- update(lwtAnov1, . ~ . * HT)
> summary(lwtAnov3)

Call:
lm(formula = BWT ~ RACE + HT + RACE:HT, data = lowbwt)

Residuals:
     Min       1Q   Median       3Q      Max
-2142.08  -474.97    11.92   531.03  1878.03

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2851.08      89.42  31.884   <2e-16 ***
RACE1         260.89     116.33   2.243   0.0261 *
RACE2         -99.25     172.91  -0.574   0.5667
HT           -788.33     365.97  -2.154   0.0325 *
RACE1:HT      630.36     490.12   1.286   0.2000
RACE2:HT      509.84     568.99   0.896   0.3714
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 709.7 on 183 degrees of freedom
Multiple R-squared:  0.07739, Adjusted R-squared:
0.05218
F-statistic:  3.07 on 5 and 183 DF,  p-value: 0.01098

> interaction.plot(lowbwt$RACE, lowbwt$HT,
+     lowbwt$BWT)
> tapply(lowbwt$BWT, list(lowbwt$RACE, lowbwt$HT),
+     mean)
```

```
          0        1
1 3111.967 2954.000
2 2751.826 2473.333
3 2851.079 2062.750
```

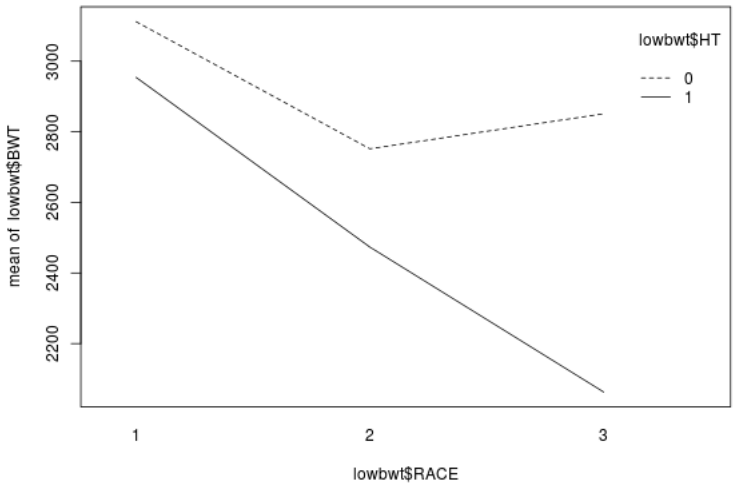

Abbildung 5 : Verteilung des Geburtsgewichtes nach ethnischer Zugehörigkeit, Quelle: Hosmer et al. 2013

Logistische Regression

`glm(..., family = "binomial",...)`

Logistische Regressionen und auch andere generalisierte lineare Modelle können Sie mit der Funktion `glm()` berechnen. Die logistische Variante definieren Sie mit dem Argument `family = "binomial"`.

Berechen Sie z.B. eine logistische Regression mit dem Datensatz *lowbwt*[3], indem Sie die dichotome Variable *LOW* als abhängige Variable einsetzen.

```
> lr01 <- glm(LOW ~ SMOKE + LWT + HT + UI,
+     data = lowbwt, family = "binomial")
> summary(lr01)

Call:
glm(formula = LOW ~ SMOKE + LWT + HT + UI, family =
"binomial",
    data = lowbwt)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6635  -0.8060  -0.6646   1.0806   1.9433

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
```

3 David W. Hosmer, Stanley Lemeshow, und Rodney X. Sturdivant, *Applied logistic regression*, 3. Aufl., Wiley series in probability and statistics (Hoboken, NJ: Wiley, 2013).

```
(Intercept)  0.721855   0.849074   0.850  0.39523
SMOKE        0.652998   0.335676   1.945  0.05174 .
LWT         -0.016306   0.006546  -2.491  0.01274 *
HT           1.922127   0.682665   2.816  0.00487 **
UI           0.896265   0.442936   2.023  0.04303 *
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 212.83  on 184  degrees of freedom
AIC: 222.83

Number of Fisher Scoring iterations: 4
```

In dieser Form werden noch die log-OR angezeigt. Um tatsächlich die Odds Ratios auszugeben, müssen Sie die Koeffizienten (und entsprechen die Konfidenzintervalle) mit `exp()` expotenzieren.

Koeffizienten mit `exp()` expotenzieren, um OR zu berechnen.

```
> exp(coef(lr01))
(Intercept)       SMOKE         LWT          HT
  2.0582484   1.9212914   0.9838264   6.8354818
         UI
  2.4504346
> exp(confint(lr01))
                2.5 %     97.5 %
(Intercept) 0.4112178 11.7059787
SMOKE       0.9953210  3.7267842
LWT         0.9704422  0.9958362
HT          1.8590581 28.5951632
UI          1.0219543  5.8746967
```

Bequemer können Sie das mit `logistic.display()` aus dem Paket *epiDisplay* bewerkstelligen.

`logisticDisplay()`

```
> library(epiDisplay)
> logistic.display(lr01)

Logistic regression predicting LOW

                 crude OR(95%CI)
SMOKE: 1 vs 0    2.02 (1.08,3.78)

LWT (cont. var.) 0.99 (0.97,1)

HT: 1 vs 0       3.37 (1.02,11.09)

UI: 1 vs 0       2.58 (1.14,5.83)

                 adj. OR(95%CI)     P(Wald's test)
SMOKE: 1 vs 0    1.92 (1,3.71)      0.052

LWT (cont. var.) 0.98 (0.97,1)      0.013

HT: 1 vs 0       6.84 (1.79,26.05)  0.005

UI: 1 vs 0       2.45 (1.03,5.84)   0.043
```

```
                  P(LR-test)
SMOKE: 1 vs 0     0.052

LWT (cont. var.)  0.007

HT: 1 vs 0        0.004

UI: 1 vs 0        0.045

Log-likelihood = -106.4129
No. of observations = 189
AIC value = 222.8257
```

Modelldiagnostik

Likelihood-Test im Vergleich zum Null-Model

Einen Likelihood-Ratio-Test (Chi²) im Vergleich mit dem Null-Modell können Sie berechnen, indem sie zunächst das Null-Model (mit `~ 1` als unabhängige Variable) erstellen und dann den Test mit `anova(..., test = "LRT", ...)` duchführen.

```
> lrNull <- glm(LOW ~ 1, data = lowbwt, family =
"binomial")
> anova(lr01, lrNull, test = "LRT")
Analysis of Deviance Table

Model 1: LOW ~ SMOKE + LWT + HT + UI
Model 2: LOW ~ 1
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       184     212.83
2       188     234.67 -4  -21.846 0.000215 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Pseudo-R²

Wenn Sie eine Pseudo-R²-Statistik berechnen wollen, können Sie das mit `LogRegR2()` aus dem Paket descr.

```
> library(descr)
> LogRegR2(lr01)
Chi2                 21.84626
Df                   4
Sig.                 0.0002150483
Cox and Snell Index  0.1091584
Nagelkerke Index     0.1535079
McFadden's R2        0.09309273
```

Hosmer-Lemeshow-Test

Den Hosmer-Lemeshow-Test können sie mit

```
> library(MKmisc)
> HLgof.test(fitted(lr01), lr01$y)
$C

Hosmer-Lemeshow C statistic

data:  fitted(lr01) and lr01$y
X-squared = 13.433, df = 8, p-value = 0.0978
```

```
$H

Hosmer-Lemeshow H statistic

data:  fitted(lr01) and lr01$y
X-squared = 13.069, df = 8, p-value = 0.1095
```

Cessie-van-Houwelingen-Test

Besser als der Hosmer-Lemeshow-Test soll der Cessie-van-Houwelingen-Test sein, weil er nicht auf einer willkürlichen Gruppierung der Daten beruht. Er ist in der Funktion `residuals()` des Paketes *lrm* realisiert. Sie müssen das Modell dazu allerdings erst einmal neu als *lrm*-Objekt anlegen.

```
> library(rms)
> lr02 <- lrm(LOW ~ LWT + RACE + SMOKE + HT +
+     UI, data = lowbwt, y = TRUE, x = TRUE)
> residuals(lr02, "gof")
Sum of squared errors     Expected value|H0
           34.4618771            34.2609952
                   SD                     Z
            0.3299107             0.6088978
                    P
            0.5425922
```

Klassifikationstabellen

KLASSIFIKATIONSTABELLEN KÖNNEN SIE mit der folgenden Funktion erstellen:

```
> klassTab <- function(x, cutoff = 0.5) {
+     prob <- predict(x, type = "response")
+     ta <- table(vorhergesagt = cut(prob,
+         c(0, cutoff, 1)), beobachtet =
x$y)/length(prob)
+     correct <- sum(ta[1, 1], ta[2, 2])
+     list(Klassifikationstabelle = round(ta,
+         3), `Anteil korrekter Schätzungen` =
round(correct,
+         3))
+ }
> klassTab(lr01)
$Klassifikationstabelle
            beobachtet
vorhergesagt     0     1
     (0,0.5] 0.640 0.228
     (0.5,1] 0.048 0.085

$`Anteil korrekter Schätzungen`
[1] 0.725
```

ROC und AUC

Receiver-Operating-Characteristic-Kurven (ROC) und Area under the curve (AUC) können Sie mit `roc()` aus dem Paket *pROC* berechnen.

```
> library(pROC)
```

```
> plot(roc(lr01$y, predict(lr01, type = "response")))

Call:
roc.default(response = lr01$y, predictor = predict(lr01,
type = "response"))

Data: predict(lr01, type = "response") in 130 controls
(lr01$y 0) < 59 cases (lr01$y 1).
Area under the curve: 0.7153

Call:
roc.default(response = lr01$y, predictor = predict(lr01,
type = "response"))

Data: predict(lr01, type = "response") in 130 controls
(lr01$y 0) < 59 cases (lr01$y 1).
Area under the curve: 0.7153
```

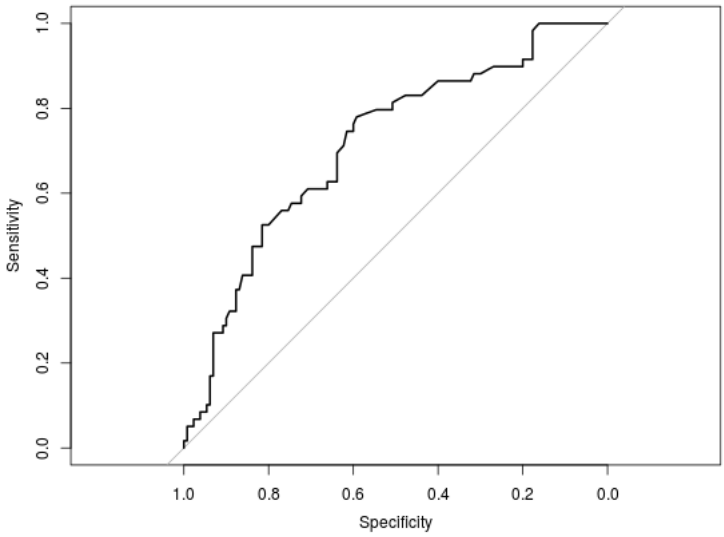

Abbildung 6 : ROC-Kuve mit pROC::roc()

2 Literatur

Field, Andy P, Jeremy Miles, und Zoë Field. *Discovering Statistics Using R*. London; Thousand Oaks, Calif.: Sage, 2012.

Hosmer, David W., Stanley Lemeshow, und Rodney X. Sturdivant. *Applied logistic regression*. 3. Aufl. Wiley series in probability and statistics. Hoboken, NJ: Wiley, 2013.