Weiterbildungs-Lehrgang in Angewandter Statistik ETHZ 2009/2011 28.09.09

Angewandte Regression — Musterlösungen zur Serie 2

1. Erzeugen der Matrizen und Vektoren in R:

```
> A <- rbind(c(3,2,4), c(1,4,6))
> B <- rbind(c(0,2,4), c(-1,-1,0))
> x <- c(1,-2,3)
> y <- c(5,3,-4)
```

a) `> 2*A`

$$2 \cdot \begin{bmatrix} 3 & 2 & 4 \\ 1 & 4 & 6 \end{bmatrix} = \begin{bmatrix} 2\cdot 3 & 2\cdot 2 & 2\cdot 4 \\ 2\cdot 1 & 2\cdot 4 & 2\cdot 6 \end{bmatrix} = \begin{bmatrix} 6 & 4 & 8 \\ 2 & 8 & 12 \end{bmatrix}$$

b) `> A+B`

$$\begin{bmatrix} 3 & 2 & 4 \\ 1 & 4 & 6 \end{bmatrix} + \begin{bmatrix} 0 & 2 & 4 \\ -1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 3+0 & 2+2 & 4+4 \\ 1+(-1) & 4+(-1) & 6+0 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 8 \\ 0 & 3 & 6 \end{bmatrix}$$

c) `> A%*%t(B)`

$$\begin{aligned} \begin{bmatrix} 3 & 2 & 4 \\ 1 & 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 0 & -1 \\ 2 & -1 \\ 4 & 0 \end{bmatrix} &= \begin{bmatrix} 3\cdot 0 + 2\cdot 2 + 4\cdot 4 & 3\cdot(-1) + 2\cdot(-1) + 4\cdot 0 \\ 1\cdot 0 + 4\cdot 2 + 6\cdot 4 & 1\cdot(-1) + 4\cdot(-1) + 6\cdot 0 \end{bmatrix} \\ &= \begin{bmatrix} 20 & -5 \\ 32 & -5 \end{bmatrix} \end{aligned}$$

d) `> A%*%x`

$$\begin{bmatrix} 3 & 2 & 4 \\ 1 & 4 & 6 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix} = \begin{bmatrix} 3\cdot 1 + 2\cdot(-2) + 4\cdot 3 \\ 1\cdot 1 + 4\cdot(-2) + 6\cdot 3 \end{bmatrix} = \begin{bmatrix} 11 \\ 11 \end{bmatrix}$$

e) Nicht definiert, da die Dimensionen nicht passen.

```
> A%*%B
Fehler in A %*% B : nicht passende Argumente
```

f) Nicht definiert, da die Dimensionen nicht passen.

```
> t(B)%*%y
Fehler in t(B) %*% y : nicht passende Argumente
```

g)

```
> A%*%t(A)
     [,1] [,2]
[1,]   29   35
[2,]   35   53
```


h)

```
> t(A)%*%A
     [,1] [,2] [,3]
[1,]   10   10   18
[2,]   10   20   32
[3,]   18   32   52
```

i) `> t(x)%*%x`

$$\begin{bmatrix} 1 & -2 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix} = 1\cdot 1 + (-2)\cdot(-2) + 3\cdot 3 = 14$$

j) `> x%*%t(x)`

$$\begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 1\cdot 1 & 1\cdot -2 & 1\cdot 3 \\ -2\cdot 1 & -2\cdot(-2) & -2\cdot 3 \\ 3\cdot 1 & 3\cdot(-2) & 3\cdot 3 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 3 \\ -2 & 4 & -6 \\ 3 & -6 & 9 \end{bmatrix}$$

2. Matrixschreibweise:

$$\begin{bmatrix} 3 & 1 & 3 \\ 0 & 0 & 4 \\ -4 & 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ -1 \end{bmatrix}$$

Lösung mit R:

```
> t.A <- rbind(c(3,1,3), c(0,0,4), c(-4,0,2))
> t.y <- c(3,2,-1)
> t.beta <- solve(t.A,t.y)
> t.beta
[1] 0.5 0.0 0.5
```

d.h.

$$\beta_1 = \frac{1}{2}, \qquad \beta_2 = 0, \qquad \beta_3 = \frac{1}{2}$$

3. a) Wir wollen hier die "wahren Werte" von $\underline{y}$ aus der Gleichung $\underline{y} = \boldsymbol{x}\underline{\beta}$ berechnen:

```
# Daten Matrix x mit "Einsen-Spalte"
t.x <- matrix(c(rep(1,5),0:4,4,1,0,1,4),ncol=3,byrow=F)
t.beta <- c(10,5,-2)
t.y <- t.x%*%t.beta
```

Das Resultat ist: $\underline{y}^T = (2, 13, 20, 23, 22)$

b) Damit wir die beobachteten Werte Y_i erhalten, müssen wir noch zu den y_i die Fehler E_i addieren.

```
# Erzeugen der Fehler und der beobachteten Werte
t.E <- rnorm(5)
t.Y <- t.y + t.E
# Bestimmung der Koeffizienten t.beta
t.beta.hut <- lm(t.Y ~ t.x[,2] + t.x[,3])$coefficients
```

In unserem Beispiel erhalten wir die folgenden Koeffizienten $\underline{\hat{\beta}}$:

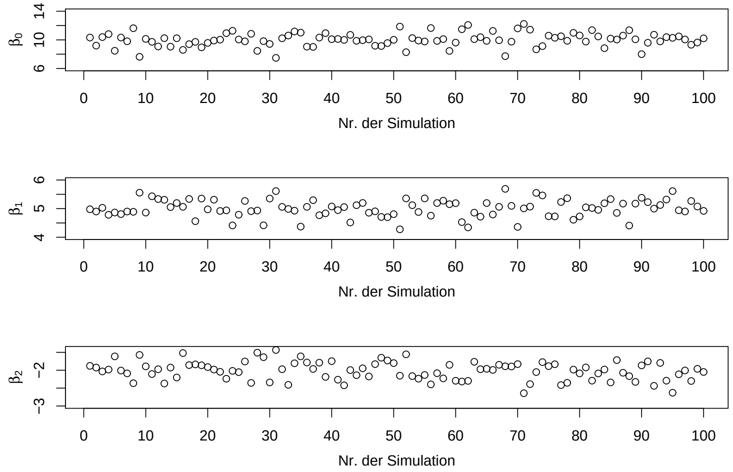
```
> t.beta.hut
(Intercept)    t.x[, 2]    t.x[, 3]
  10.356110    5.164399   -2.004548
```

c) Wir gehen hier wie unter "R-Hinweise" beschrieben vor:

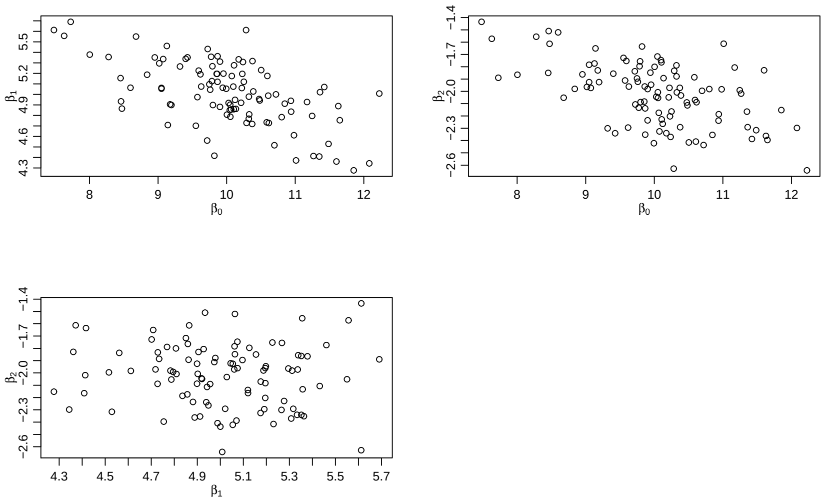
```
# Fehlermatrix für 100 Datensätze erzeugen
t.E <- matrix(rnorm(500),ncol=100)

# Matrix mit 100 identischen Spalten des Inhalts t.y erzeugen
t.y <- matrix(rep(t.y,100),nrow=5,byrow=F)

# Simulierte Y-Werte, pro Spalte ein Datensatz für lineare Regression
t.Y <- t.E + t.y

# For-Schleife zur Berechnung Koeffizienten aus den 100 Simulationen
# Resultatmatrix definieren
r.coef <- matrix(nrow=100,ncol=3)
for (i in 1:100) {
  r.coef[i,] <- lm(t.Y[,i] ~ t.x[,2] + t.x[,3])$coefficients
}

# oder eleganter mit apply
r.coef <- t(apply(t.Y, 2, FUN = function(y)
                  lm(y ~ t.x[,2] + t.x[,3])$coefficients))
```

Erzeugen der Streudiagramme:

```
# Darstellung aller geschätzten Parameter als Funktion des Datensatzes
par(mfrow=c(3,1))
par(mar=c(5,5,0,0)+0.5,cex=1.1, mgp=c(2.4,1,0))
plot(1:100,r.coef[,1],xlab='Nr. der Simulation',ylab=expression(beta[0]))
plot(1:100,r.coef[,2],xlab='Nr. der Simulation',ylab=expression(beta[1]))
plot(1:100,r.coef[,3],xlab='Nr. der Simulation',ylab=expression(beta[2]))

# Streudiagramme der verschiedenen Koeffizienten gegeneinander
par(mfrow=c(2,2))
plot(r.coef[,1],r.coef[,2],xlab=expression(beta[0]),ylab=expression(beta[1]))
plot(r.coef[,1],r.coef[,3],xlab=expression(beta[0]),ylab=expression(beta[2]))
plot(r.coef[,2],r.coef[,3],xlab=expression(beta[1]),ylab=expression(beta[2]))
```

Die Werte der y-Achsenabschnitte schwankten in unserer Simulation zwischen 7 und 12, die Steigung von $x^{(1)}$ zwischen 4 und 6, die Steigung von $x^{(2)}$ zwischen -3 und -1.



Bei diesem Modell sind offensichtlich β_1 und β_2 mit β_0 negativ korreliert. Aufgrund der geschickt gewählten Stützwerte sind β_1 und β_2 voneinander unabhängig: Die Vektoren $(x^{(1)} - \bar{x}^{(1)})$ und $(x^{(2)} - \bar{x}^{(2)})$ sind orthogonal.

4. Daten einlesen und Scatterplot von `y` gegen `raddos` erzeugen:

```
t.url <- "http://stat.ethz.ch/Teaching/Datasets/NDK/antkoerp.dat"
d.antikoerp <- read.table(t.url, header=T)
plot(d.antikoerp$raddos, d.antikoerp$y, xlab="raddos", ylab="y")
```

a) Resultat für das Regressionsmodell $Y_i = \alpha + \beta\ \texttt{raddos}_i + E_i$:

```
r.antik <- lm(y ~ raddos, data = d.antikoerp)
summary(r.antik)

Call:
lm(formula = y ~ raddos, data = d.antikoerp)

Residuals:
    Min      1Q  Median      3Q     Max
-222.12  -80.12  -21.79   83.21  226.55

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 154.1303   113.0942   1.363   0.2061
raddos        1.2466     0.5203   2.396   0.0402 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 146.9 on 9 degrees of freedom
Multiple R-Squared: 0.3895,Adjusted R-squared: 0.3216
F-statistic: 5.741 on 1 and 9 DF,  p-value: 0.04016
```

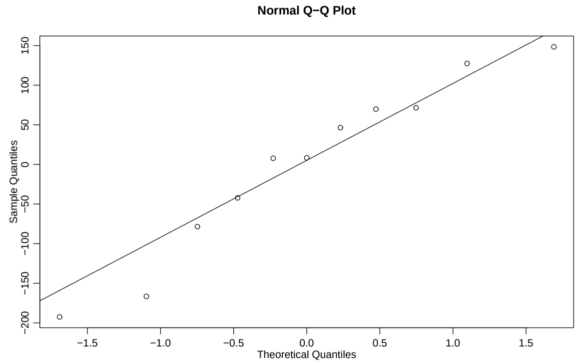
$\widehat{\beta}$ ist auf dem 5%-Niveau signifikant von 0 verschieden. Das Modell ist aber nicht sehr gut: $R^2 = 0.389$ ist klein und $\widehat{\sigma}^2 = 146.9^2 = 21591$ gross.

b) Das polynomiale Modell ist besser, denn $R^2 = 0.613$ ist viel grösser als vorher und $\widehat{\sigma}^2 = 15413$ kleiner.

```
r.antik2 <- lm(y ~ raddos + I(raddos^2), data = d.antikoerp)
summary(r.antik2)

Call:
lm(formula = y ~ raddos + I(raddos^2), data = d.antikoerp)

Residuals:
     Min       1Q   Median       3Q      Max
-192.509  -60.369    8.353   70.648  148.446

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.986e+02  1.901e+02  -1.045   0.3267
raddos       5.555e+00  2.055e+00   2.704   0.0269 *
I(raddos^2) -1.077e-02  5.018e-03  -2.147   0.0641 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 124.1 on 8 degrees of freedom
Multiple R-Squared: 0.6126,Adjusted R-squared: 0.5157
F-statistic: 6.325 on 2 and 8 DF,  p-value: 0.02253
```

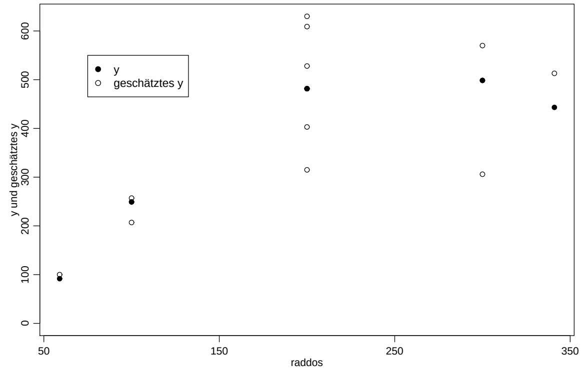
c) Die Fehler können als normalverteilt betrachtet werden.

```
qqnorm(resid(r.antik2))
qqline(resid(r.antik2))
```

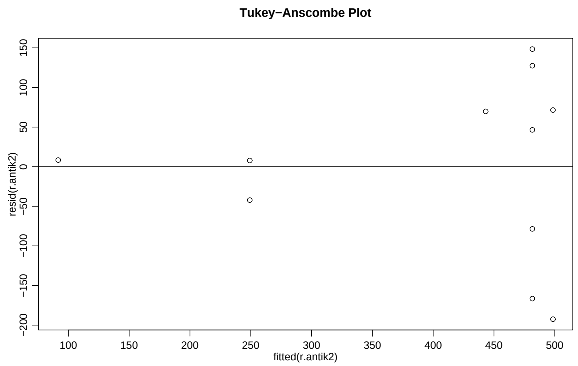
d) Das polynomiale Modell passt recht gut, wenn man aber den Tukey-Anscombe-Plot betrachtet, sieht man, dass die Arbeit noch nicht beendet ist!

```
t.yAchse <- c(0, max(d.antikoerp$y, fitted(r.antik2)))
plot(d.antikoerp$raddos,d.antikoerp$y, ylim=t.yAchse, pch=1,
     ylab="y und geschätztes y", xlab="raddos")
points(d.antikoerp$raddos, fitted(r.antik2), pch=2)
legend(75, 550, c("y", "geschätztes y"), pch=c(1,2), cex=1.1)
```



Tukey-Anscombe Plot:

```
plot(fitted(r.antik2), resid(r.antik2))
abline(h=0)
title("Tukey-Anscombe Plot")
```



Die Streuung der Residuen nimmt nach rechts zu.