uebung6.r

Janina

Sun Jun 11 21:40:41 2017

```
# Uebungsblatt 6
# Namen: Janina Schoenberger, Benjamin Weigner
# Tutorin: Gergana Stanilova
# Uebung: Mi 12-14 Uhr

library(lasso2)
```

```
## Warning: package 'lasso2' was built under R version 3.3.2
```

```
## R Package to solve regression problems while imposing
##   an L1 constraint on the parameters. Based on S-plus Release 2.1
## Copyright (C) 1998, 1999
## Justin Lokhorst   <jlokhors@stats.adelaide.edu.au>
## Berwin A. Turlach <bturlach@stats.adelaide.edu.au>
## Bill Venables     <wvenable@stats.adelaide.edu.au>
##
## Copyright (C) 2002
## Martin Maechler <maechler@stat.math.ethz.ch>
```

```
data(Prostate)

# Aufgabe 13
# Koeffizientenvektor von verschiedenen Regressionsverfahren

# beta_a = (0,0,-2,3) # Lasso
# lasso selektiert Variablen, dh sie werden aus dem Modell ausgeschlossen
# beta_b = (3,1,-3,4) # NNLS
# mit NNLS negative Werte bei korrelierten Variablen
# beta_c = (2,0.2,-2,2) # Ridge
# Ridge deckelt Variablen (keine Variable ist 0!)
# beta_d= (3,1,0,4) # OLS
# klassische Regression

# Aufgabe 14
# Prostate. Betrachte alle Variablen mit lpsa als Response
# A

# Funktion ridge.regression berechnet Koeffizientenvektor fuer Ridge Regression
ridge.regression <- function(x,y,lambda){ # x=erklaerende Variablen, y=response
  betas <- solve(t(x)%*%x + lambda*diag(8)) %*% t(x)%*%y
  return (betas)
}

# B
# Ridge Regression mit lamda=1 mit Intercept
library('glmnet')
```

```
## Warning: package 'glmnet' was built under R version 3.3.3

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.3.3

## Loaded glmnet 2.0-10
```

```
library('car')
```

```
## Warning: package 'car' was built under R version 3.3.3
```

```
set.seed(3094)
head(Prostate)
```

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
```

```
x <- as.matrix(cbind(Prostate$lcavol,Prostate$lweight,Prostate$age,Prostate$lbph,Prostate$svi,Prostate$
y <-Prostate$lpsa

# normale Kleinste Quadrate Regression
fit <- lm(lpsa ~ lcavol+lweight+age+lbph+svi+lcp+gleason+pgg45, data = Prostate)
coef(fit)
```

```
##  (Intercept)       lcavol      lweight          age         lbph
##  0.669399027  0.587022881  0.454460641 -0.019637208  0.107054351
##          svi          lcp      gleason        pgg45
##  0.766155885 -0.105473570  0.045135964  0.004525324
```

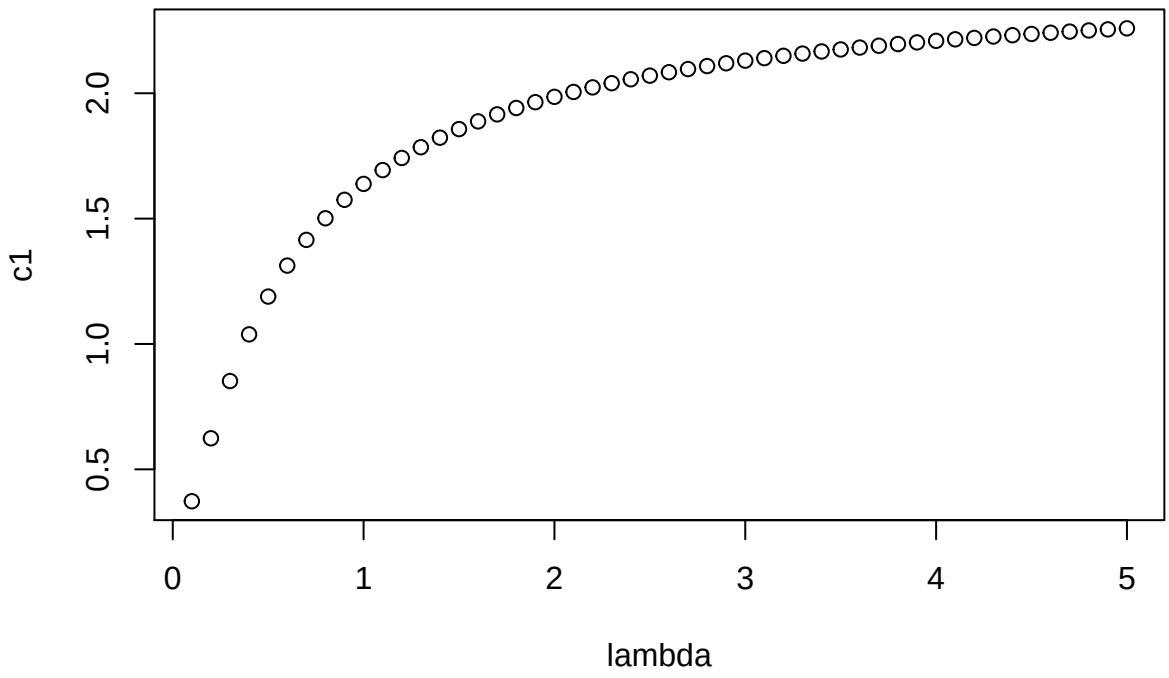
```
# Ridge Regression mit Formel
fit_ridge <- glmnet(x,y,alpha=0,lambda=1)
coef(fit_ridge)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  0.372751241
## V1           0.265541585
## V2           0.304906825
## V3          -0.001522060
## V4           0.052750345
## V5           0.450367595
## V6           0.078077224
## V7           0.085434261
## V8           0.002663404
```

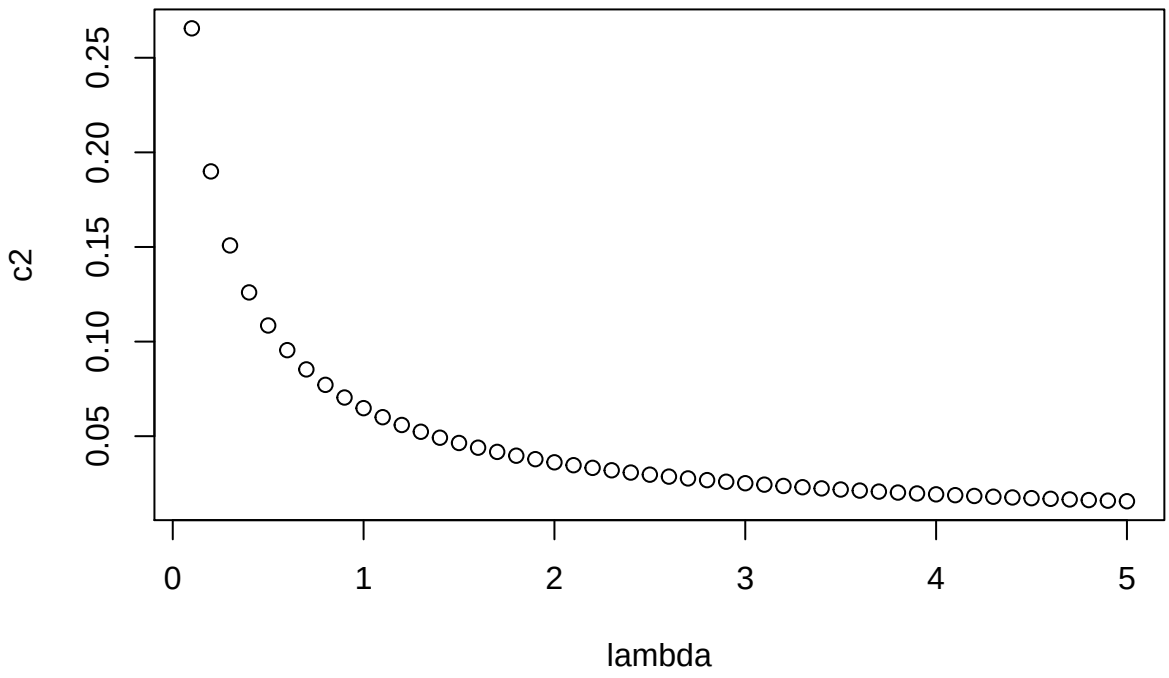
```r
# Ridge Regression mit eigener Formel aus A
ridge.regression(x,y,1)
```

```
##              [,1]
## [1,]  0.574305826
## [2,]  0.482906681
## [3,] -0.016413660
## [4,]  0.094677837
## [5,]  0.696494891
## [6,] -0.089504475
## [7,]  0.107515164
## [8,]  0.003388734
```

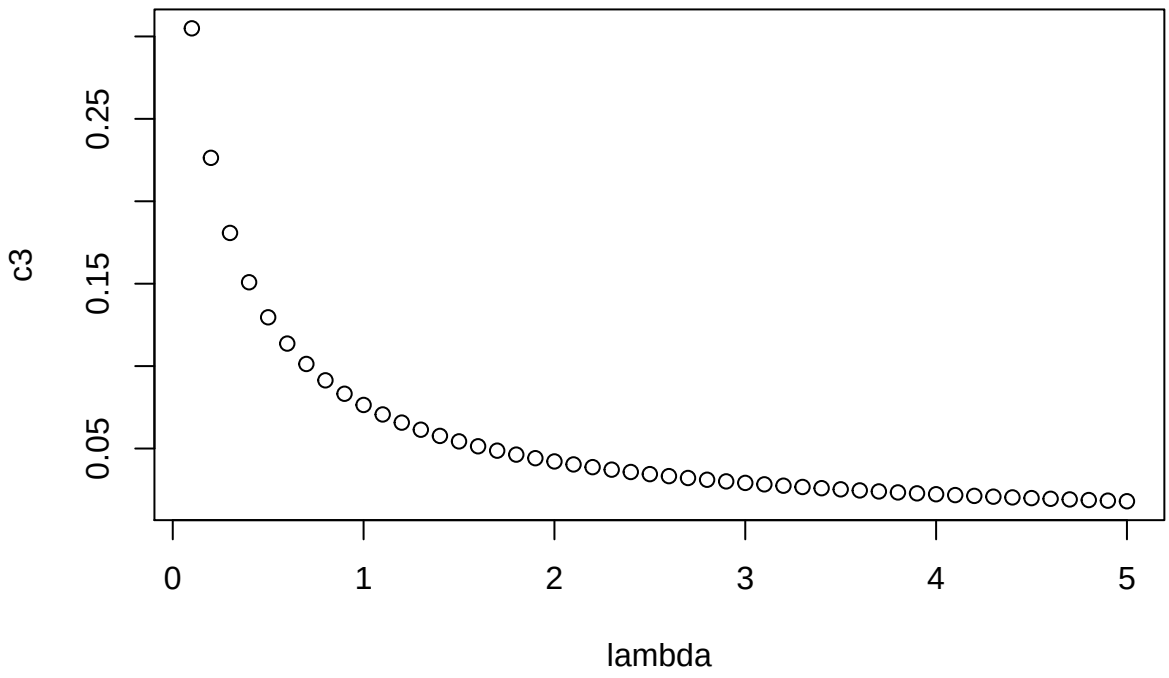
```r
# Interpretation
# Die betas sind bei Ridge mit Formel und OLS kaum unterschiedlich
# Ein Multiplizieren mit der Einheitsmatrix (lambda=1 -> 1*E=E) veraendert
# die Koeffizienten nicht besonders (Kein Bias)

# C
# Ridge Regression mit Intercept. Lambda im Interval [0.1,50] in Schritten von je 0.1
# Plot Koeffizienten in Abhaengigkeit von lambda
# Einfluss der Regularisierung (unterschiedl. Verhalten von sci,lavclo)
lambda <- seq(0.1,5,0.1)
c <- list()
for (i in 1:length(lambda)){
  fit_ridge <- glmnet(x,Prostate$lpsa,alpha=0,lambda=i)
  c[[i]] <- coef(fit_ridge)

}
c1 <- c()
c2 <- c()
c3 <- c()
c4 <- c()
c5 <- c()
c6 <- c()
c7 <- c()
c8 <- c()
c9 <- c()
for (i in 1:length(lambda)){
  c1 <- append(c1,c[[i]][1])
  c2 <- append(c2,c[[i]][2])
  c3 <- append(c3,c[[i]][3])
  c4 <- append(c4,c[[i]][4])
  c5 <- append(c5,c[[i]][5])
  c6 <- append(c6,c[[i]][6])
  c7 <- append(c7,c[[i]][7])
  c8 <- append(c8,c[[i]][8])
  c9 <- append(c9,c[[i]][9])
}
plot(lambda,c1)
```



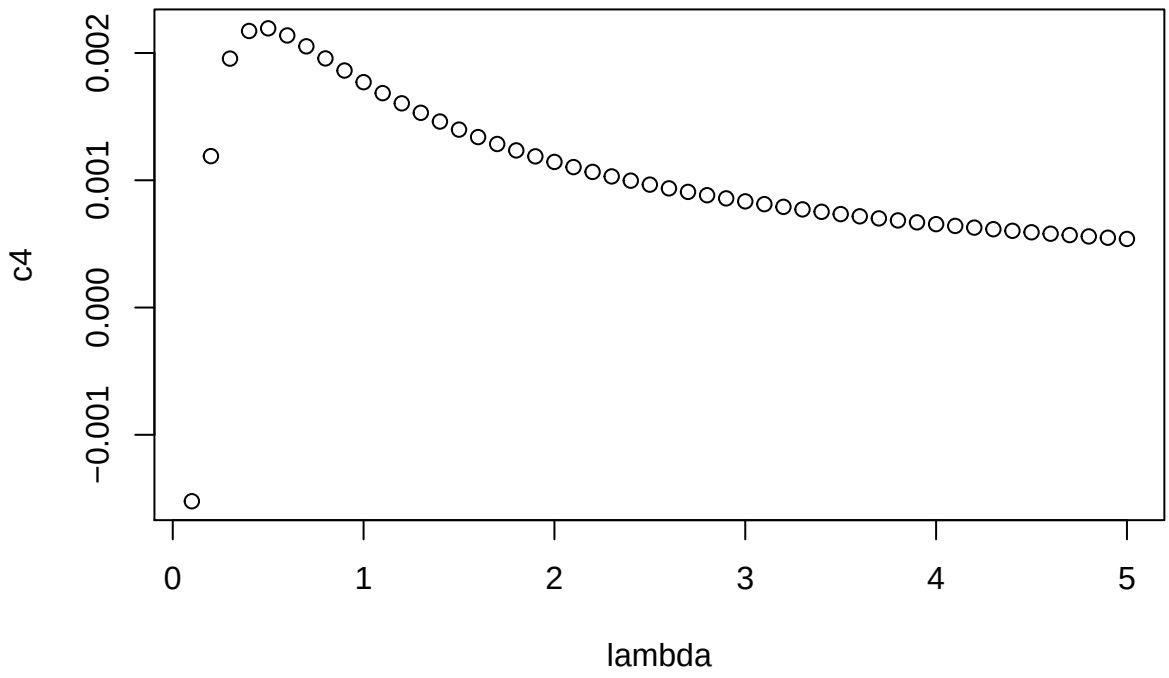

```
plot(lambda,c2)
```



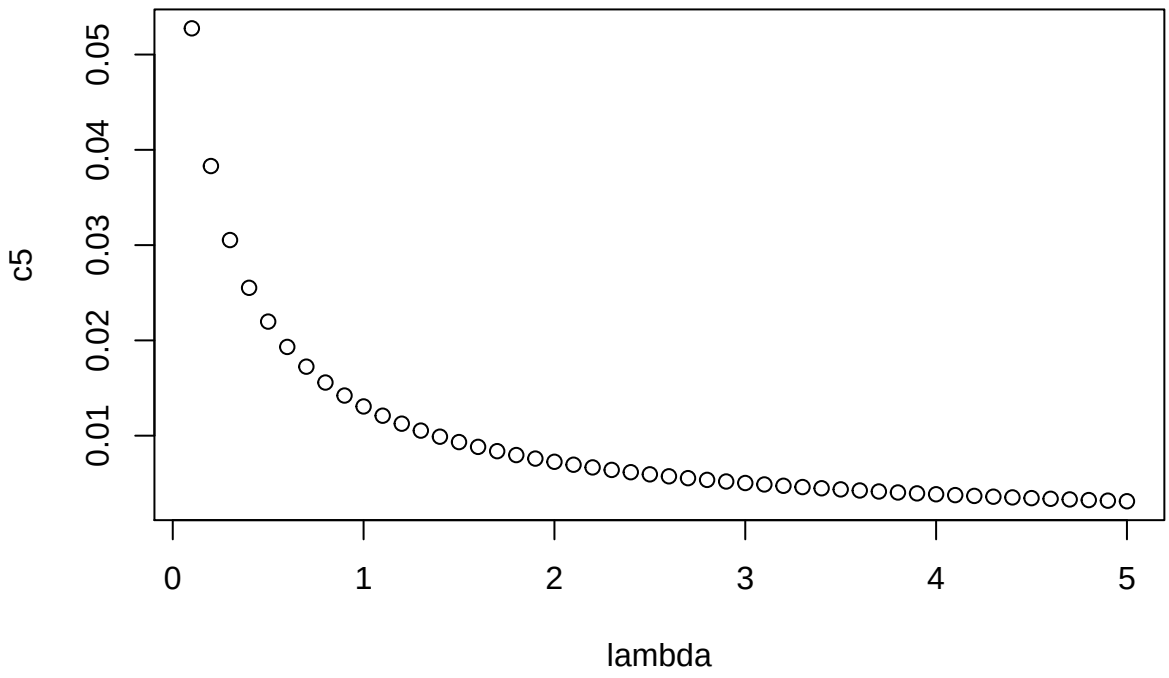

```
plot(lambda,c3)
```



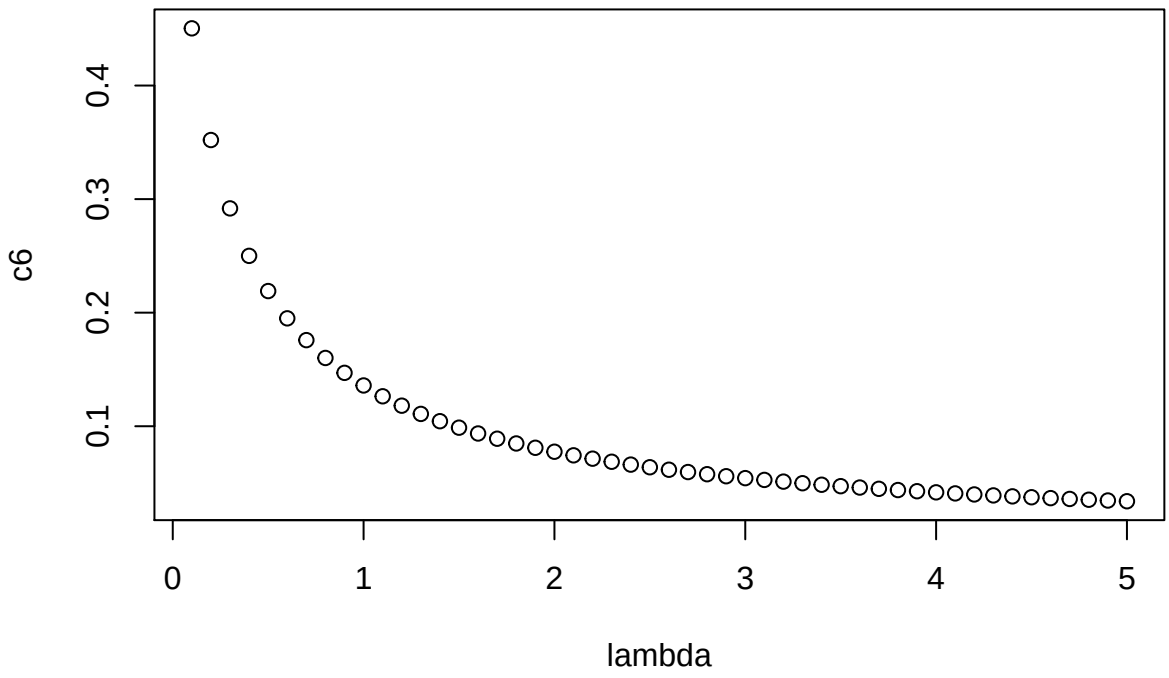

```
plot(lambda,c4)
```



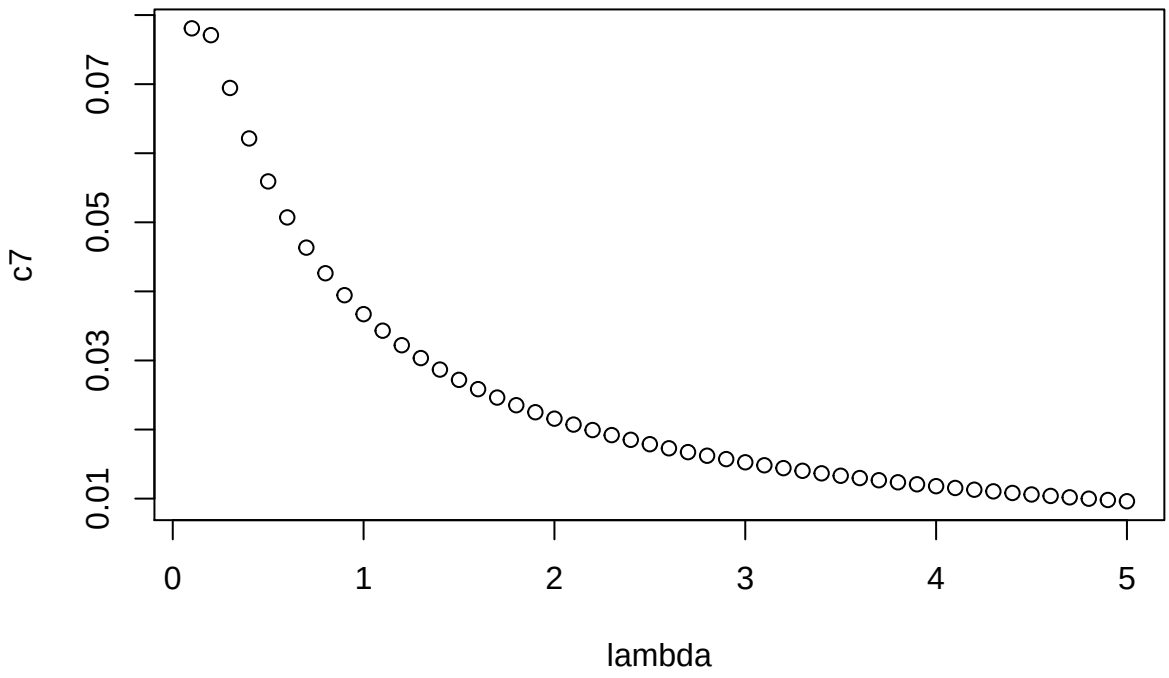

```
plot(lambda,c5)
```



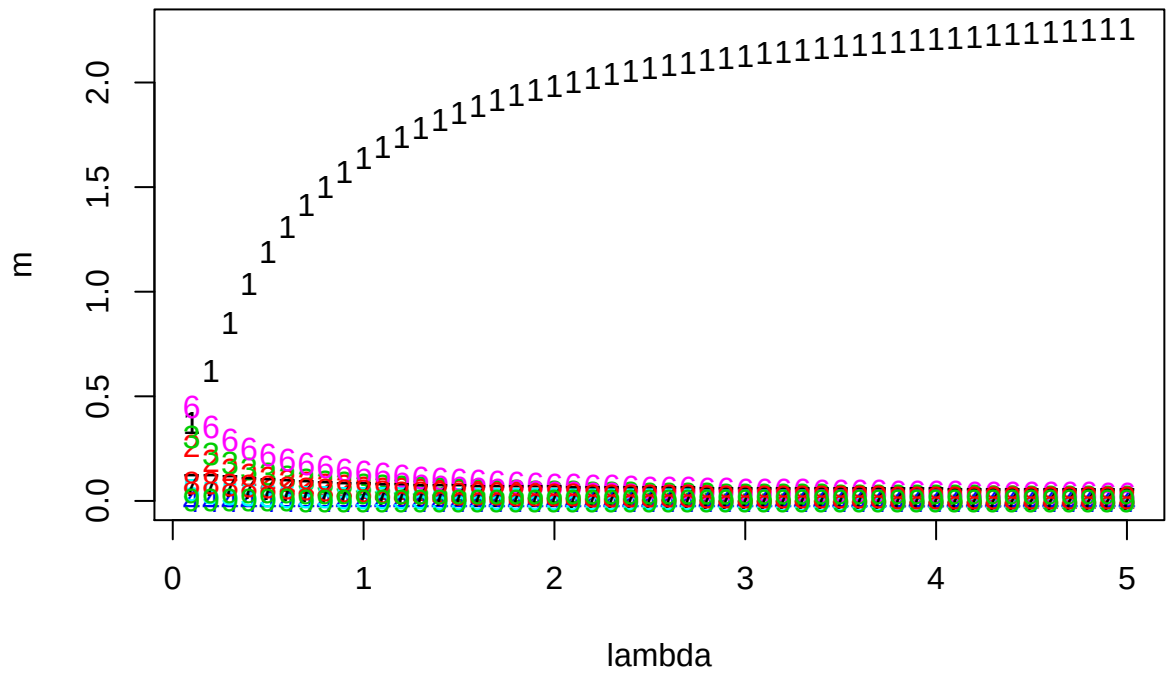

```
plot(lambda,c6)
```




```
plot(lambda,c7)
```




```
m <- cbind(c1,c2,c3,c4,c5,c6,c7,c8,c9)
matplot(lambda,m, xlab='lambda')
```




```
# 1 im matplot, bzw c1 ist der Intercept. Mit wachsendem lamba nimmt dieser zu
# Der Einfluss der anderen Variablen nimmt ab. Dh die klassische Regression ist nah
# an der Mittelwertsgerade

# D
# RSSQ = sum((Y-Yquer)^2)
# (Y-Yquer) sind die Residuen
# Yquer = X*Beta
# (wobei X die Designmatrix und Beta der Vektor mit den Beta-Koeffizienten (aus 14A) ist)
ridge.rssq <- function(x,y,lambda){
  beta <- ridge.regression(x,y,lambda)
  yquer <- x%*%beta
  residuen <- y-yquer
  rssq <- sum((residuen)^2)
  return (rssq)
}

rssq_fr1 <- ridge.rssq(x,y,0.00000008)
rssq_fr2 <- ridge.rssq(x,y,0.1)
rssq_fr3 <- ridge.rssq(x,y,10)
rssq_fr4 <- ridge.rssq(x,y,100)
fit_ols <- lm(lpsa ~ lcavol+lweight+age+lbph+svi+lcp+gleason+pgg45, data = Prostate)
rssq_ols <- sum(resid(fit_ols)^2)

rssq_fr1
```

```
## [1] 44.29694
```

```
rssq_fr2
```

```
## [1] 44.2977
```

```
rssq_fr3
```

```
## [1] 46.48599
```

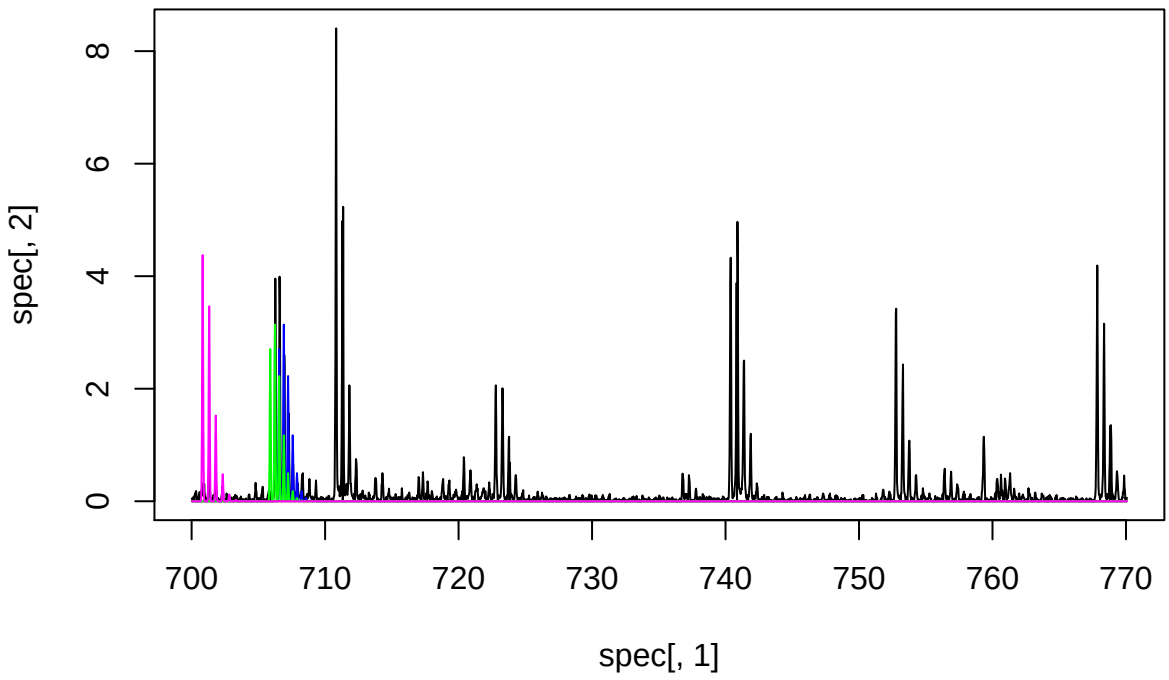
```
rssq_fr4
```

```
## [1] 62.09568
```

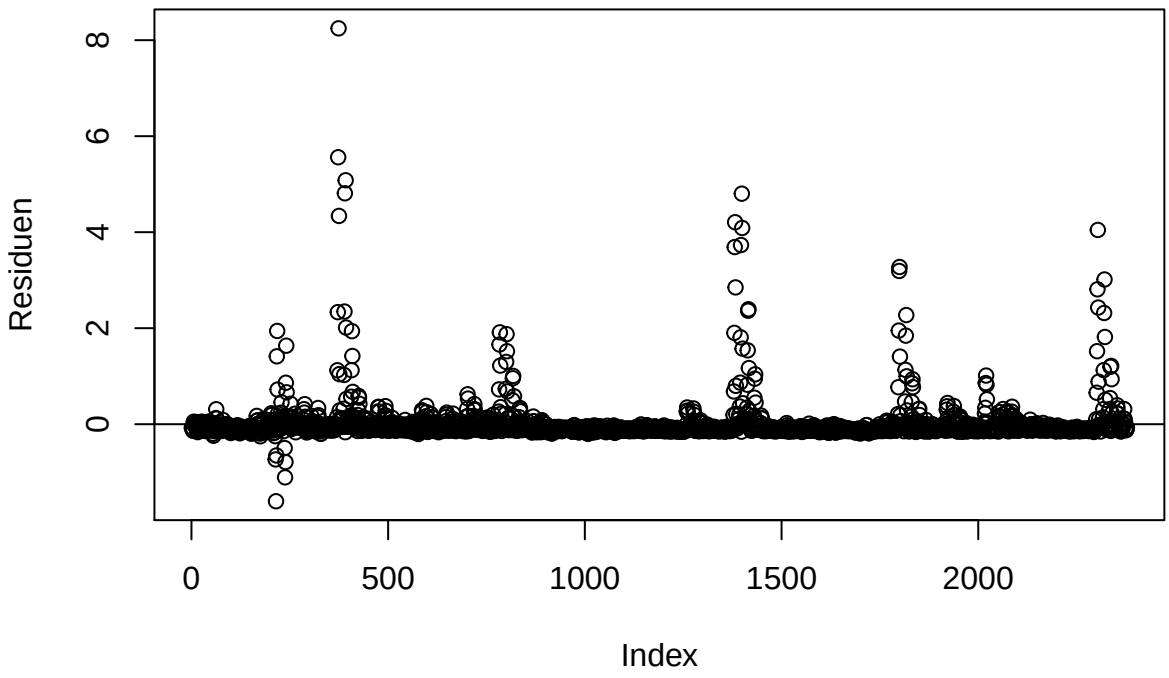
```
rssq_ols
```

```
## [1] 44.16313
```

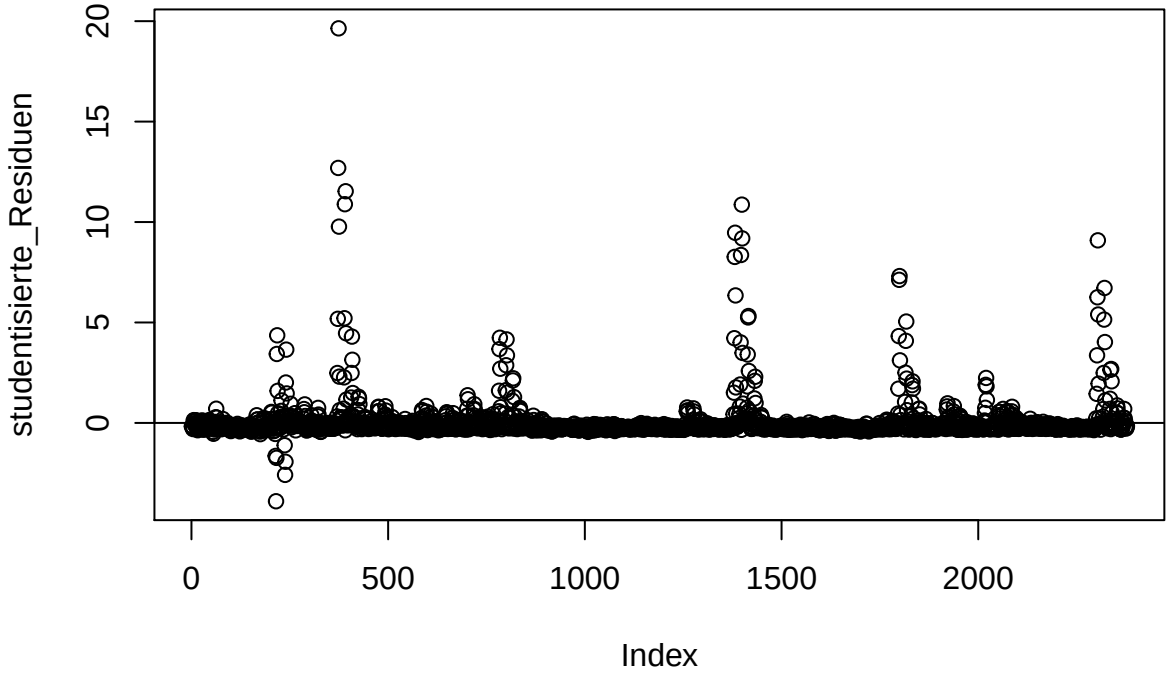
```
# Mit kleinen lambdas (kein grosser bias) erhaelt man aehnliche RSSQ-Werte
# wie beim normalen kleinste Quadrate Modell.
# Das ist zu erwarten -> Bias/Variance Tradeoff
# Ein hoeherer Bias fuehrt zu groesseren Residuen


# Aufgabe 15
# Spec
load("C:/Users/Janina/Desktop/FU/4. Semester/Statistik II/Uebungen/spec.rda")
# 1. Spalte: Massenpositionen, 2. Spalte: gemessene Intensitaeten
# Model
load("C:/Users/Janina/Desktop/FU/4. Semester/Statistik II/Uebungen/model.rda")
# Jede Spalte: Intensitaeten eines anderen Modells fuer die Massenpositionen im Spektrum

# A
# OLS Regression mit Intensitaeten des Spektrums durch Modelle 200-210 als Variablen erklaert
fit_spec <- lm(spec[,2]~model[,200]+model[,201]+model[,202]+model[,203]+model[,204]+model[,205]+model[,
# Modellannahmen ueberpruefen
# Plot des Spektrums
plot(spec[,1],spec[,2],type='l')
# einige Modelle
lines(spec[,1],model[,210]*10,col='blue')
lines(spec[,1],model[,200]*10,col='red')
lines(spec[,1],model[,208]*10,col='green')
lines(spec[,1],model[,207]*10,col='magenta')
```



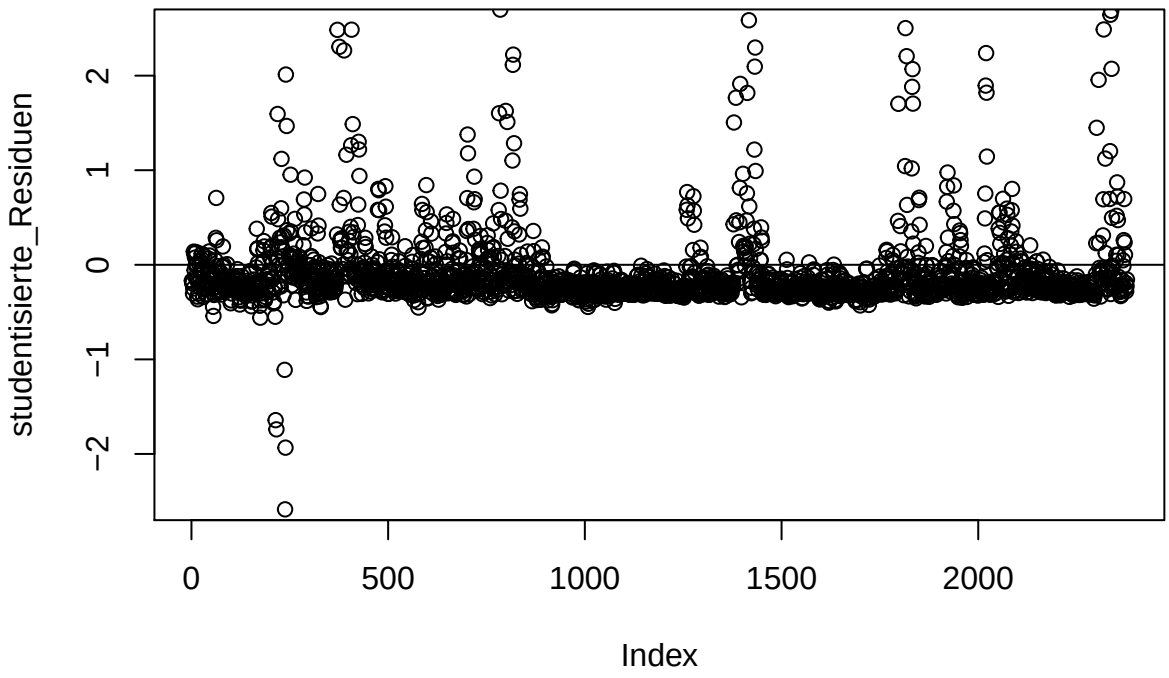

```
# Diagramm der Residuen
Residuen = fit_spec$residuals
plot(Residuen)
abline(0,0)
```



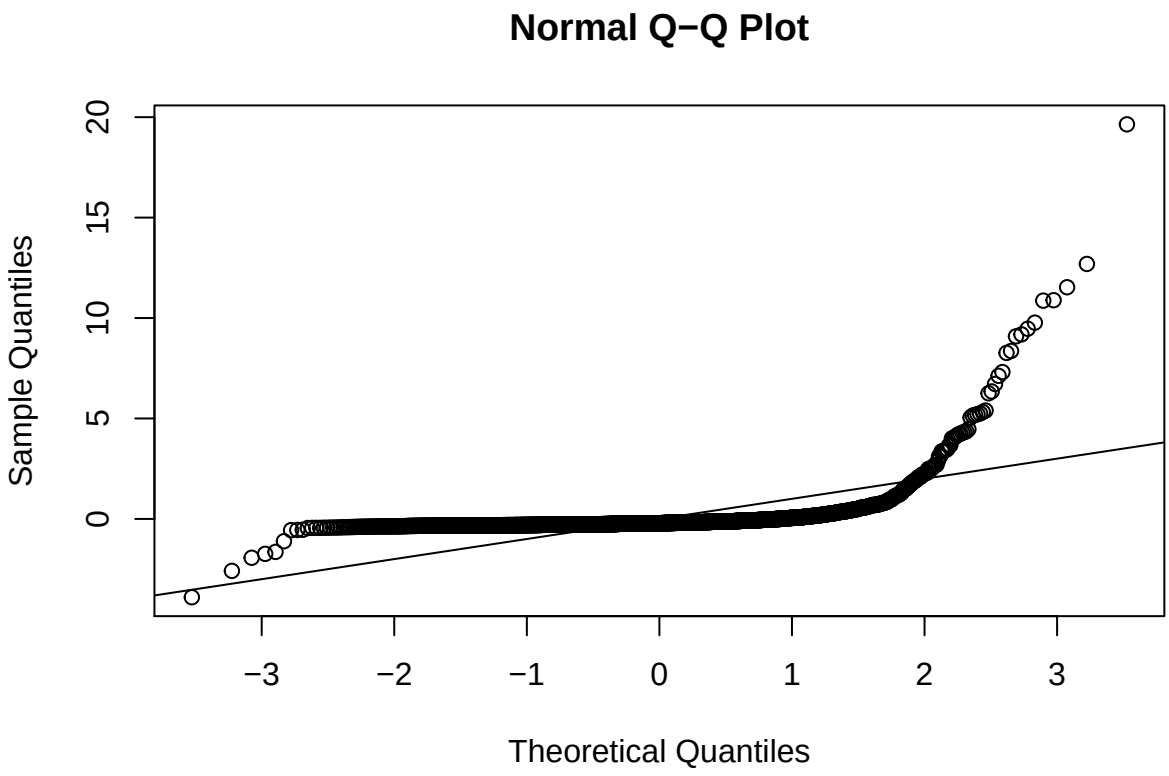

```
# Diagramm der studentisierten Residuen
studentisierte_Residuen = rstudent(fit_spec)
plot(studentisierte_Residuen)
abline(0,0)
```



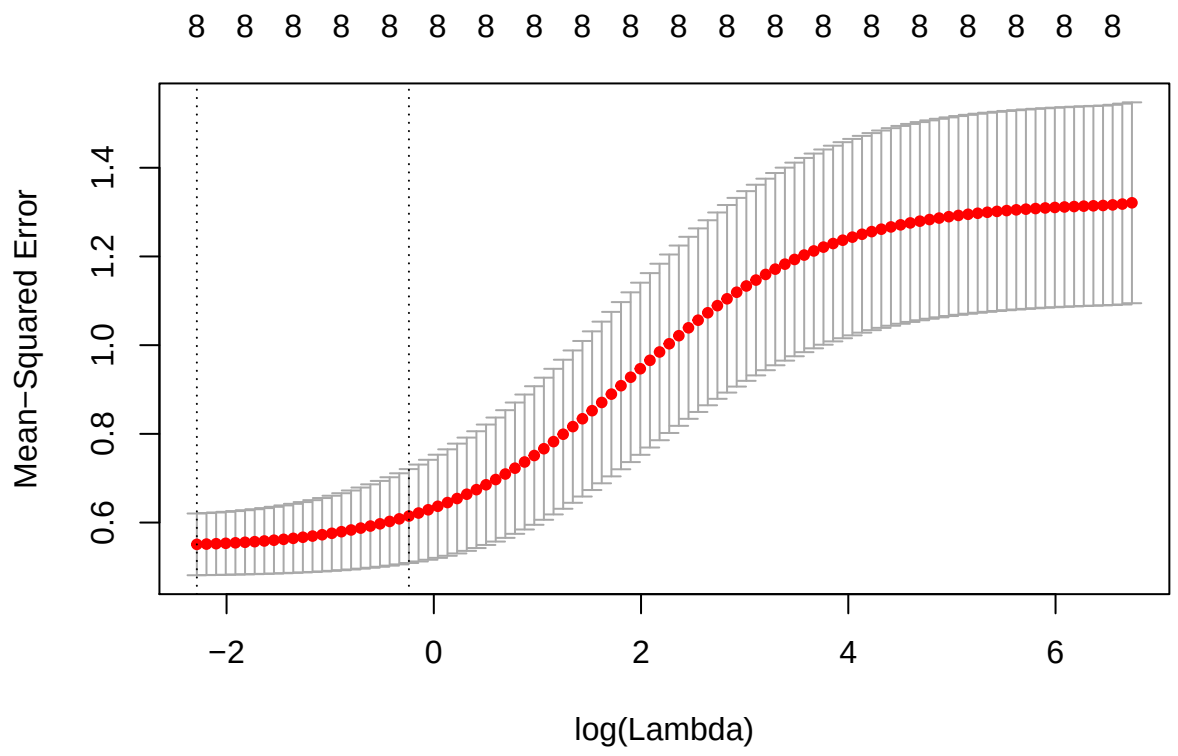

```
# 95% der Residuen muessen zwischen 2 und -2 sein
plot(studentisierte_Residuen,ylim=c(-2.5,2.5))
abline(0,0)
```




```
# Auswertung: Es gibt einflussreiche Ausreisser im Bereich 5-20 bei den Residuen,
# allerdings liegen die meisten (~95%) im Bereich -2,2.
# Die Residuen haeufen sich an einigen Stellen -> systematischer Fehler erkennbar
# --> Unabhaegigkeit ist nicht gegeben

# QQ-Plot zur Ueberpruefung der Verteilungsannahme
qqnorm(studentisierte_Residuen)
abline(0,1)
```




```
# Die Kurve des QQ Plot verlaeuft nicht annaehernd wie die 45° Gerade, sondern ist sigmoidal/zeitweise
# --> Normalverteilungsannahme ist nicht erfuellt

# B
cv.ridge <- cv.glmnet(x,Prostate$lpsa,alpha=0,family='gaussian',nfolds=5,type.measure = 'mse')
plot(cv.ridge)
```




```
cv.ridge$lambda.min
```

```
## [1] 0.1015911
```

```
# Ridge mit Funktion
#coef_ridge <- coef(cv.ridge,s=cv.ridge$lambda.min)
#coef_ridge
# Ridge mit Formel
ridge.regression(x,y,cv.ridge$lambda.min)
```

```
##              [,1]
## [1,]  0.577937337
## [2,]  0.494495068
## [3,] -0.017424927
## [4,]  0.096636528
## [5,]  0.765330428
## [6,] -0.105663944
## [7,]  0.107385443
## [8,]  0.003413167
```

```
# Vergleich zu A
coef(fit_spec)
```

```
##   (Intercept)  model[, 200]  model[, 201]  model[, 202]  model[, 203]
```

```
##  1.411516e-01 -1.056176e+15 -3.986848e+15 -2.029329e+09 -9.266460e+10
##  model[, 204]  model[, 205]  model[, 206]  model[, 207]  model[, 208]
##  7.018765e+09 -1.621127e+02 -1.350758e+02  3.693110e-01 -8.604124e-01
##  model[, 209]  model[, 210]
##  1.318088e+01 -1.963061e+00
```

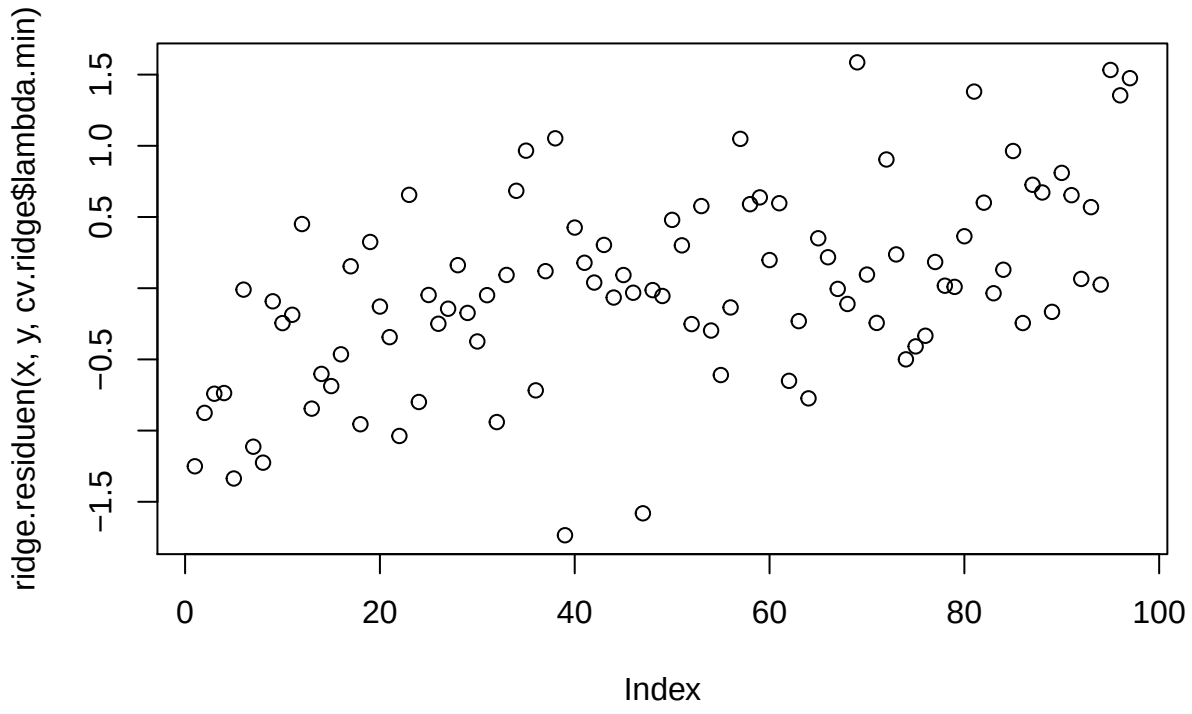
```
sum(resid(fit_spec)^2)
```

```
## [1] 485.4734
```

```
ridge.rssq(x,y,cv.ridge$lambda.min)
```

```
## [1] 44.29773
```

```
# -> Die Ridge Regression liefert deutlich kleinere Residuen

# Hilfsfunktion fuer Residuen
ridge.residuen <- function(x,y,lambda){
  beta <- ridge.regression(x,y,lambda)
  yquer <- x%*%beta
  residuen <- y-yquer
  return (residuen)
}
plot(ridge.residuen(x,y,cv.ridge$lambda.min))
```



```r
# Ohne studentisierte Residuen und QQ Plot sieht man, dass durch den Bias
# die Residuen jetzt eine 'normalisierte' Varianz haben


# Aufgabe 16
# A
# Lasso-Loesung fuer das Regressionsproblem mit Intercept. Betrachtet werden 15 zuerst gewaehlte Kovari
library(lars)
```

```
## Warning: package 'lars' was built under R version 3.3.2
```

```
## Loaded lars 1.2
```

```r
methods(class='lars')
```

```
## [1] coef    plot    predict print   summary
## see '?methods' for accessing help and source code
```

```r
#x <- cbind(model[,200],model[,201],model[,202],model[,203],model[,204],model[,205],model[,206],model[,
cv.lasso <- cv.glmnet(model,spec[,2],alpha=0,family='gaussian',nfolds=5,type.measure = 'mse')
cv.lasso$lambda.min
```

```
## [1] 0.06564805
```

```r
coef_lasso <- coef(cv.lasso,s=cv.lasso$lambda.min)
#coef_lasso
cut_coef_lasso <- coef_lasso[1:15]
cut_coef_lasso
```

```
##  [1]  7.501762e-02 -1.032803e+14  1.178309e+15 -4.872256e+14  3.034790e+14
##  [6] -2.161775e+15 -5.686823e+14 -1.167007e+15  1.051981e+14  1.583342e+15
## [11] -7.874317e+13 -5.366963e+12 -3.508703e+15  1.256692e+15  1.366577e+14
```

```r
#cv.lars(model,spec[,2],5,plot.it=FALSE,type='lasso',trace=TRUE)


# B
# AIC und BIC Kriterium mit Formel
# AIC = n*log(RSS)+2*df(lambda)
# BIC = n*log*(RSS)+log(n)*df(lambda)
# fuer alle Modelle, die schrittweise mit Lasso berechnet wurden von 1 bis max 15 Einflussvariablen
# df(lambda) = summe von j=1 bis p (d_j^2/(d_j^2+lambda))



# C
# AIC und BIC visualisieren, vergleichen der jew ausgewaehlten Modelle
# Welches Modell aufgrund des AIC/BIC?
# -> Ein Modell ist umso geeigneter, je kleiner AIC/BIC ist
```