Regression 03

A. Shrinkage (regularization) methods

B. Orthogonalization methods

Josep Fortiana 2019-10-15

Both families of methods are applicable when there are many predictors, possibly multicollinear.

Shrinkage, or regularization, methods replace the ordinary least squares condition with penalized least squares, where the penalty term purpose is to diminish the regression coefficients variance (dispersion, unstability). This is the shrinkage in the name.

Orthogonalization methods replace the set of observed predictor variables with a new set of orthogonal variables, derived as linear combinations of the old ones in such a way that the *prediction space,* that is, the space of columns of X, the regression matrix is conserved.

In this laboratory we see two shrinkage methods: Ridge regression and the Lasso, and two orthogonalization methods, Principal Components Regression (PCR) and Partial Least Squares (PLS).

A1. Ridge regression

1. Longley dataset and the `lm.ridge` function in the `MASS` package

```
require(MASS)
```

```
## Loading required package: MASS
```

```
data(longley)
str(longley)
```

```
## 'data.frame':    16 obs. of  7 variables:
##  $ GNP.deflator: num  83 88.5 88.2 89.5 96.2 ...
##  $ GNP         : num  234 259 258 285 329 ...
##  $ Unemployed  : num  236 232 368 335 210 ...
##  $ Armed.Forces: num  159 146 162 165 310 ...
##  $ Population  : num  108 109 110 111 112 ...
##  $ Year        : int  1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 ...
##  $ Employed    : num  60.3 61.1 60.2 61.2 63.2 ...
```

```
longley.ridge.1<-lm.ridge(Employed ~ .,data=longley)
str(longley.ridge.1)
```

```
## List of 9
##  $ coef  : Named num [1:6] 0.157 -3.447 -1.828 -0.696 -0.344 ...
##   ..- attr(*, "names")= chr [1:6] "GNP.deflator" "GNP" "Unemployed" "Armed.Forces" ...
##  $ scales: Named num [1:6] 10.45 96.24 90.48 67.38 6.74 ...
##   ..- attr(*, "names")= chr [1:6] "GNP.deflator" "GNP" "Unemployed" "Armed.Forces" ...
##  $ Inter : int 1
##  $ lambda: num 0
##  $ ym    : num 65.3
##  $ xm    : Named num [1:6] 102 388 319 261 117 ...
##   ..- attr(*, "names")= chr [1:6] "GNP.deflator" "GNP" "Unemployed" "Armed.Forces" ...
##  $ GCV   : Named num 0.00836
##   ..- attr(*, "names")= chr "0"
```

```
##  $ kHKB  : num 0.00428
##  $ kLW   : num 0.0323
##  - attr(*, "class")= chr "ridgelm"
```

```r
coefficients(longley.ridge.1)
```

```
##                GNP.deflator           GNP    Unemployed  Armed.Forces
## -3.482259e+03  1.506187e-02 -3.581918e-02 -2.020230e-02 -1.033227e-02
##    Population          Year
## -5.110411e-02  1.829151e+00
```

```r
longley.ridge.1$scales
```

```
## GNP.deflator          GNP   Unemployed Armed.Forces   Population
##    10.448877    96.238735    90.479112    67.382126     6.735216
##         Year
##     4.609772
```

```r
print(longley.ridge.1)
```

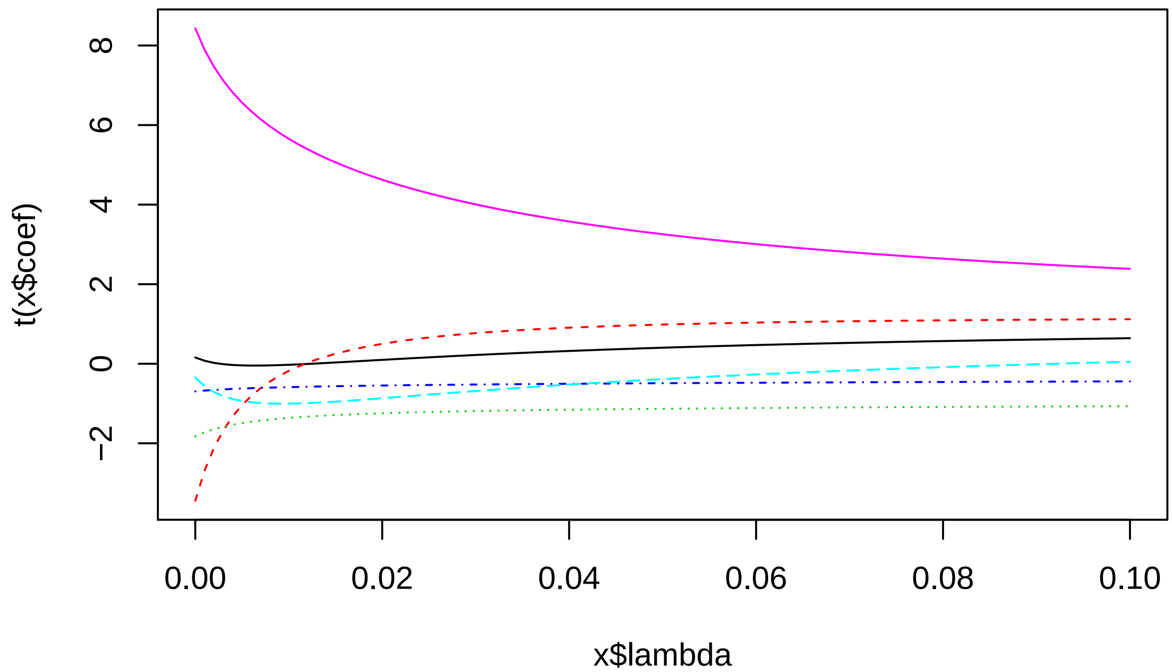
```
##                GNP.deflator           GNP    Unemployed  Armed.Forces
## -3.482259e+03  1.506187e-02 -3.581918e-02 -2.020230e-02 -1.033227e-02
##    Population          Year
## -5.110411e-02  1.829151e+00
```

```r
summary(longley.ridge.1)
```

```
##        Length Class  Mode
## coef   6      -none- numeric
## scales 6      -none- numeric
## Inter  1      -none- numeric
## lambda 1      -none- numeric
## ym     1      -none- numeric
## xm     6      -none- numeric
## GCV    1      -none- numeric
## kHKB   1      -none- numeric
## kLW    1      -none- numeric
```

`kHKB` is an estimate of the optimal λ, proposed by Hoerl, Kennard and Baldwin (1975). `kLW` is another estimate, proposed by Lawless, Wang (1976). `GCV` is the Generalized Cross-Validation statistic evaluated for each of the λ values being tested.

```r
longley.ridge<-lm.ridge(Employed ~ .,data=longley,lambda=seq(0,0.1,by=0.001))
options(repr.plot.width=5, repr.plot.height=5)
plot(longley.ridge)
```




```r
select(longley.ridge)
```

```
## modified HKB estimator is 0.004275357
## modified L-W estimator is 0.03229531
## smallest value of GCV  at 0.003
```

The `broom` package has functions to gather and visualize the output of `lm.ridge`

```r
#install.packages("broom",dependencies=TRUE,repos="https://cloud.r-project.org")
require(broom)
```

```
## Loading required package: broom
```

```r
# tidy(longley.ridge)
# long output
```

```r
glance(longley.ridge)
```

```
## # A tibble: 1 x 3
##      kHKB    kLW lambdaGCV
##     <dbl>  <dbl>     <dbl>
## 1 0.00428 0.0323     0.003
```

2. Acetylene dataset and the `genridge` package by Michael Friendly

```r
#install.packages("genridge",dependencies=TRUE,repos="https://cloud.r-project.org")
#install.packages("car",dependencies=TRUE,repos="https://cloud.r-project.org")
require(genridge)
```

```
## Loading required package: genridge
## Loading required package: car
## Loading required package: carData
```

```
require(car)
```

The `genridge` package includes the `Acetylene` dataset, with new variable names. We recover the linear model we tried above on these data and then we try a second linear model with quadratic terms. As a matter of fact this dataset originates from the paper: Marquardt, Donald W. and Snee, Ronald D. (1975), *"Ridge Regression in Practice"*, The American Statistician, Vol. 29, No. 1, pp. 3-20. Un this paper the authors start with the model with all six quadratic terms:

$$\text{temp}^2,\ \ \text{ratio}^2,\ \ \text{time}^2,\ \ \text{temp}\cdot\text{ratio},\ \ \text{temp}\cdot\text{time},\ \ \text{ratio}\cdot\text{time}.$$

```
data(Acetylene)
str(Acetylene)
```

```
## 'data.frame':    16 obs. of  4 variables:
##  $ yield: num  49 50.2 50.5 48.5 47.5 44.5 28 31.5 34.5 35 ...
##  $ temp : int  1300 1300 1300 1300 1300 1300 1200 1200 1200 1200 ...
##  $ ratio: num  7.5 9 11 13.5 17 23 5.3 7.5 11 13.5 ...
##  $ time : num  0.012 0.012 0.0115 0.013 0.0135 0.012 0.04 0.038 0.032 0.026 ...
```

```
# Same model as above, with only linear terms (main effects)
Acetylene.lm1<-lm(yield~temp+ratio+time,data=Acetylene)
summary(Acetylene.lm1)
```

```
##
## Call:
## lm(formula = yield ~ temp + ratio + time, data = Acetylene)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.920 -1.856  0.234  2.074  6.948
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -121.26962   55.43571  -2.188   0.0492 *
## temp           0.12685    0.04218   3.007   0.0109 *
## ratio          0.34816    0.17702   1.967   0.0728 .
## time         -19.02170  107.92824  -0.176   0.8630
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.767 on 12 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.8998
## F-statistic: 45.88 on 3 and 12 DF,  p-value: 7.522e-07
```

```
vif(Acetylene.lm1)
```

```
##      temp     ratio      time
## 12.225045  1.061838 12.324964
```

```
X.Acetylene.lm1<-model.matrix(Acetylene.lm1)
kappa(X.Acetylene.lm1)
```

```
## [1] 201893.3
```

```r
# Model from the original paper by Marquardt and Snee
Acetylene.lm2 <- lm(yield ~ temp + ratio + time + I(temp^2)+ I(ratio^2)+ I(time^2)
                    + temp:ratio+temp:time+ratio:time, data=Acetylene)
summary(Acetylene.lm2)
```

```
##
## Call:
## lm(formula = yield ~ temp + ratio + time + I(temp^2) + I(ratio^2) +
##     I(time^2) + temp:ratio + temp:time + ratio:time, data = Acetylene)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3499 -0.3411  0.1297  0.5011  0.6720
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.617e+03  3.136e+03  -1.153  0.29260
## temp         5.324e+00  4.879e+00   1.091  0.31706
## ratio        1.924e+01  4.303e+00   4.472  0.00423 **
## time         1.377e+04  1.045e+04   1.318  0.23572
## I(temp^2)   -1.927e-03  1.896e-03  -1.016  0.34874
## I(ratio^2)  -3.034e-02  1.168e-02  -2.597  0.04084 *
## I(time^2)   -1.158e+04  7.699e+03  -1.504  0.18318
## temp:ratio  -1.414e-02  3.212e-03  -4.404  0.00455 **
## temp:time   -1.058e+01  8.241e+00  -1.283  0.24666
## ratio:time  -2.103e+01  9.241e+00  -2.276  0.06312 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9014 on 6 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9943
## F-statistic: 289.7 on 9 and 6 DF,  p-value: 3.225e-07
```

```r
vif(Acetylene.lm2)
```

```
##         temp        ratio         time    I(temp^2)   I(ratio^2)
## 2.856749e+06 1.095614e+04 2.017163e+06 2.501945e+06 6.573359e+01
##    I(time^2)   temp:ratio    temp:time   ratio:time
## 1.266710e+04 9.802903e+03 1.428092e+06 2.403594e+02
```

```r
X.Acetylene.lm2<-model.matrix(Acetylene.lm2)
kappa(X.Acetylene.lm2)
```

```
## [1] 125655134398
```

```r
# A third model, with fewer quadratic terms, used by Michael Friendly to illustrate genridge
Acetylene.lm3 <- lm(yield ~ temp + ratio + time + I(time^2) + temp:time, data=Acetylene)
summary(Acetylene.lm3)
```

```
##
## Call:
## lm(formula = yield ~ temp + ratio + time + I(time^2) + temp:time,
##     data = Acetylene)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -7.3186 -1.2320  0.2038  2.2028  5.6327
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -121.9766   138.3525  -0.882   0.3987
## temp            0.1298     0.1033   1.257   0.2373
## ratio           0.3518     0.1871   1.880   0.0895 .
## time         2209.1184  3506.2107   0.630   0.5428
## I(time^2)   -2091.3422  5966.8212  -0.350   0.7332
## temp:time      -1.8758     2.6044  -0.720   0.4879
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.965 on 10 degrees of freedom
## Multiple R-squared:  0.926,  Adjusted R-squared:  0.889
## F-statistic: 25.02 on 5 and 10 DF,  p-value: 2.368e-05
```

```
vif(Acetylene.lm3)
```

```
##         temp        ratio         time    I(time^2)    temp:time
##    66.144125     1.070829 11743.892195   393.392844  7373.542427
```

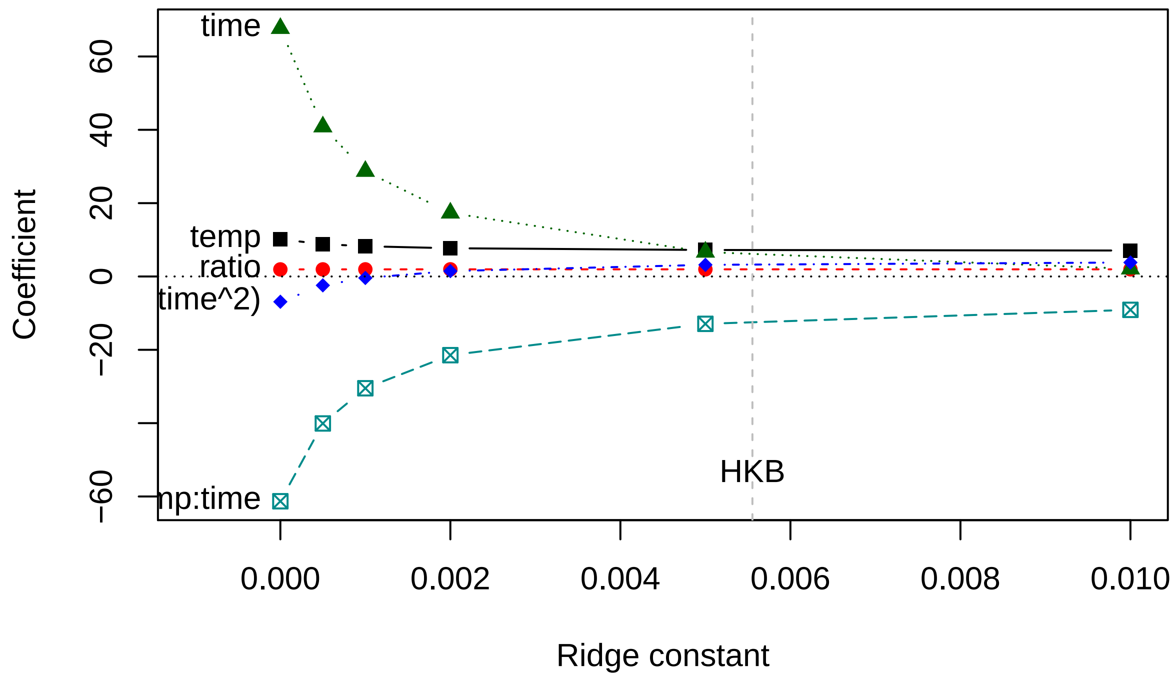
```
X.Acetylene.lm3<-model.matrix(Acetylene.lm3)
kappa(X.Acetylene.lm3)
```

```
## [1] 9431117
```

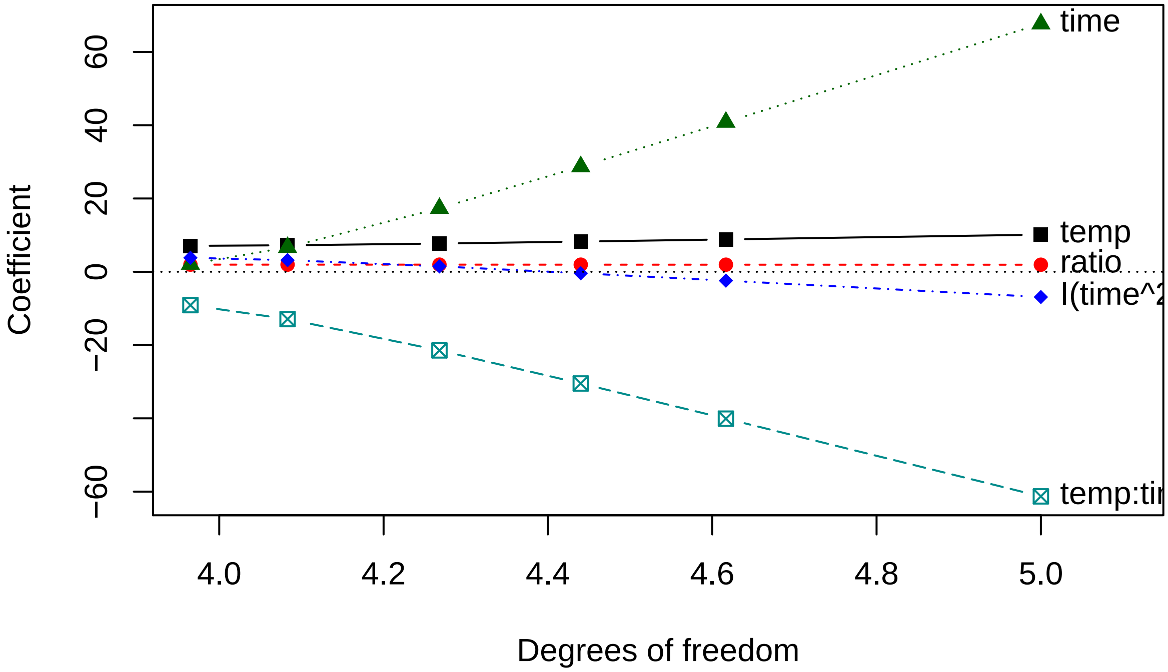
```
# Ridge regression with the ridge function from genridge
y<- Acetylene[,"yield"]
X0<-X.Acetylene.lm3[,-1]
lambda <- c(0, 0.0005, 0.001, 0.002, 0.005, 0.01)
Acetylene.ridge.1 <- ridge(y, X0, lambda=lambda)
summary(Acetylene.ridge.1)
```

```
##        Length Class  Mode
## lambda  6     -none- numeric
## df      6     -none- numeric
## coef   30     -none- numeric
## cov     6     -none- list
## mse     6     -none- numeric
## scales  5     -none- numeric
## kHKB    1     -none- numeric
## kLW     1     -none- numeric
## GCV     6     -none- numeric
## kGCV    1     -none- numeric
## svd.D   5     -none- numeric
## svd.U  80     -none- numeric
## svd.V  25     -none- numeric
```

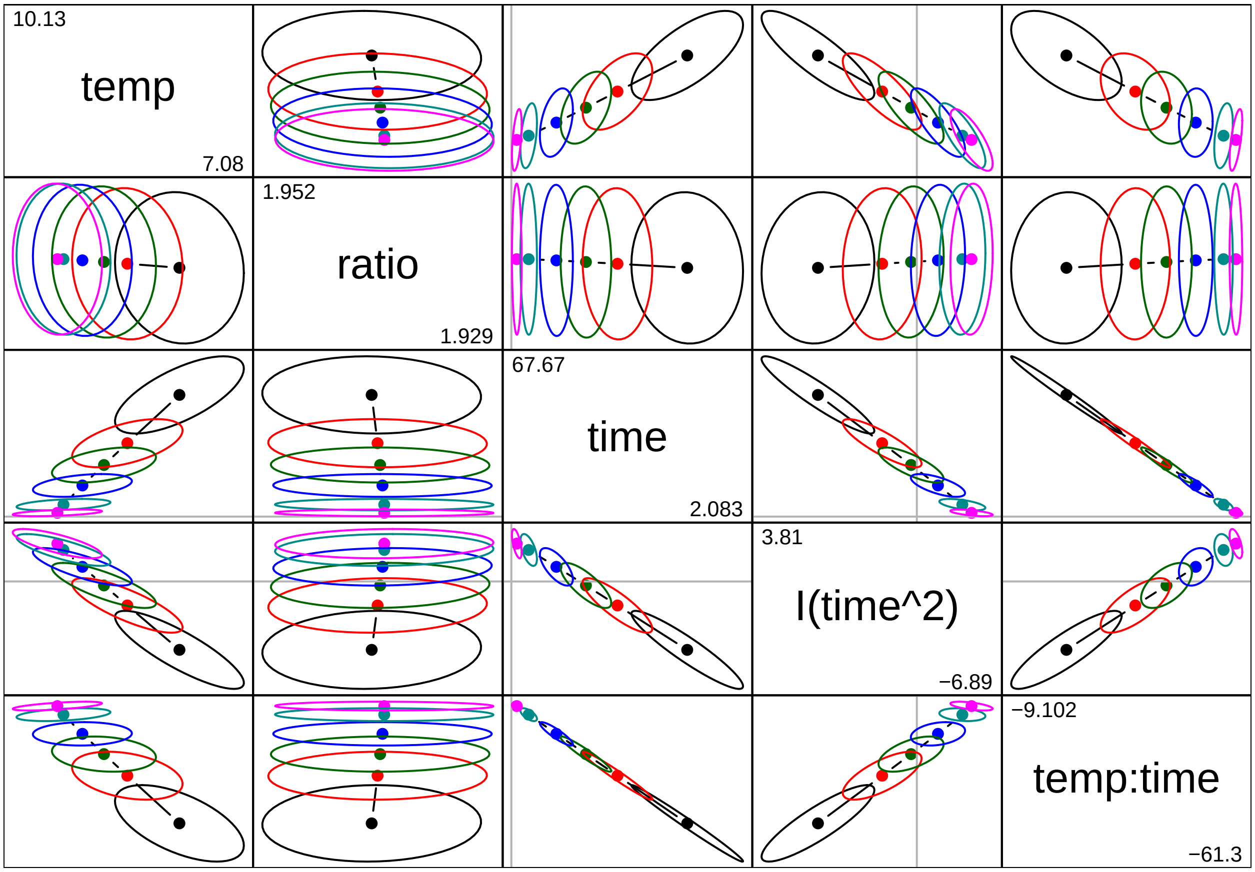
```
traceplot(Acetylene.ridge.1)
```




```
traceplot(Acetylene.ridge.1, X="df")
```




```
pairs(Acetylene.ridge.1, radius=0.2)
```



3. The Fearn dataset

A dataset from the paper by Fearn, T. (1983), *A Misuse of Ridge Regression in the Calibration of a Near Infrared Reflectance Instrument,* Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 32, No. 1(1983), pp. 73-79. This paper, with intended controversial title and contents, found its rebuttal in the paper by Hoerl, Arthur E., Kennard, Robert W. and Hoerl, Roger W. (1985), *Practical Use of Ridge Regression: A Challenge Met,* Journal of the Royal Statistical Society. Series C (Applied Statistics), Vol. 34, No. 2(1985), pp. 114-120.

```
Fearn.1<-read.table("Fearn.data.1.txt", header=TRUE)
Fearn.2<-read.table("Fearn.data.2.txt", header=TRUE)
str(Fearn.1)
```

```
## 'data.frame':    24 obs. of  7 variables:
##  $ y : num  9.23 8.01 10.95 11.67 10.41 ...
##  $ x1: int  468 458 457 450 464 499 463 462 488 483 ...
##  $ x2: int  123 112 118 115 119 147 119 115 134 141 ...
##  $ x3: int  246 236 240 236 243 273 242 238 258 264 ...
##  $ x4: int  374 368 359 352 366 404 370 370 393 384 ...
##  $ x5: int  386 383 353 340 371 433 377 353 377 398 ...
##  $ x6: int  -11 -15 -16 -15 -16 5 -12 -13 -5 -2 ...
```

```
str(Fearn.2)
```

```
## 'data.frame':    26 obs. of  7 variables:
##  $ y : num  8.66 7.9 9.27 11.77 9.7 ...
##  $ x1: int  486 485 482 443 478 449 461 503 493 368 ...
##  $ x2: int  144 136 136 112 134 113 121 155 146 40 ...
```

```
##  $ x3: int  266 260 260 232 257 233 243 280 271 158 ...
##  $ x4: int  393 393 388 346 382 351 366 403 390 275 ...
##  $ x5: int  373 395 423 355 390 343 378 414 378 250 ...
##  $ x6: int  26 6 -2 -18 -5 -18 -14 6 -3 -63 ...
```

Adjust the regression `y~x1+x2+x3+x4+x5+x6` with the Fearn dataset and:

1. Ordinary Least Squares (OLS), selecting the best predictors subset
2. Ridge regression

Compare prediction errors. Which one is better?

3. After working through the following section on the lasso, repeat with this method.

NOTE: the data frames `Fearn.1` and `Fearn.2` were used as train and test subsets in the original paper. You may choose to follow this selection or merge both subsets and partition the joint dataset in some other way.

4. The `Hitters` dataset in the `ISLR` package

Ridge regression following ISLR - Chap 6 - Laboratory 2 - Using the `glmnet` package

Code from the ISLR website

```r
#install.packages("ISLR",dependencies=TRUE,repos="https://cloud.r-project.org")
require(ISLR)
```

```
## Loading required package: ISLR
```

```r
#fix(Hitters)
names(Hitters)
```

```
##  [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"
##  [6] "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"
## [11] "CRuns"     "CRBI"      "CWalks"    "League"    "Division"
## [16] "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
```

```r
dim(Hitters)
```

```
## [1] 322  20
```

```r
sum(is.na(Hitters$Salary))
```

```
## [1] 59
```

```r
Hitters=na.omit(Hitters)
dim(Hitters)
```

```
## [1] 263  20
```

```r
sum(is.na(Hitters))
```

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

```r
# Prepare x, y for the glmnet syntax
x<-model.matrix(Salary~.,Hitters)[,-1]
y<-Hitters$Salary
```

```r
#install.packages("glmnet",dependencies=TRUE,repos="https://cloud.r-project.org")
require(glmnet)
```

```
## Loading required package: glmnet
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach

## Loaded glmnet 2.0-18
```

A grid of lambda values

```r
# When lambda goes to infinity penalization on coefficients beta01 through beta19 is so high
# that it pushes all of them down to zero, resulting in a model with no predictors, only the intercept
#
# Syntax:
# alpha=0 is for ridge regression
# alpha=1 is for 'lasso'  regression (cfr. below)
#
grid<-10^seq(10,-2,length=100)
ridge.mod<-glmnet(x,y,alpha=0,lambda=grid)
str(ridge.mod)
```

```
## List of 12
##  $ a0       : Named num [1:100] 536 536 536 536 536 ...
##   ..- attr(*, "names")= chr [1:100] "s0" "s1" "s2" "s3" ...
##  $ beta     :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:1900] 0 1 2 3 4 5 6 7 8 9 ...
##   .. ..@ p       : int [1:101] 0 19 38 57 76 95 114 133 152 171 ...
##   .. ..@ Dim     : int [1:2] 19 100
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:19] "AtBat" "Hits" "HmRun" "Runs" ...
##   .. .. ..$ : chr [1:100] "s0" "s1" "s2" "s3" ...
##   .. ..@ x       : num [1:1900] 5.44e-08 1.97e-07 7.96e-07 3.34e-07 3.53e-07 ...
##   .. ..@ factors : list()
##  $ df       : int [1:100] 19 19 19 19 19 19 19 19 19 19 ...
##  $ dim      : int [1:2] 19 100
##  $ lambda   : num [1:100] 1.00e+10 7.56e+09 5.72e+09 4.33e+09 3.27e+09 ...
##  $ dev.ratio: num [1:100] 2.76e-07 3.64e-07 4.82e-07 6.37e-07 8.42e-07 ...
##  $ nulldev  : num 53319113
##  $ npasses  : int 2130
##  $ jerr     : int 0
##  $ offset   : logi FALSE
##  $ call     : language glmnet(x = x, y = y, alpha = 0, lambda = grid)
##  $ nobs     : int 263
##  - attr(*, "class")= chr [1:2] "elnet" "glmnet"
```

```r
# Compare the beta regression coefficients with a large lambda (small absolute values)
# and with a smaller lambda (larger absolute values).
dim(coef(ridge.mod))
```

```
## [1]  20 100
```

```r
round(ridge.mod$lambda[50],2)
```

```
## [1] 11497.57
```

```r
round(coef(ridge.mod)[,50],2)
```

```
## (Intercept)       AtBat        Hits       HmRun        Runs         RBI
##      407.36        0.04        0.14        0.52        0.23        0.24
##       Walks       Years      CAtBat       CHits      CHmRun       CRuns
##        0.29        1.11        0.00        0.01        0.09        0.02
```

```
##        CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##        0.02         0.03         0.09        -6.22         0.02         0.00
##      Errors   NewLeagueN
##       -0.02         0.30
```

```
round(sqrt(sum(coef(ridge.mod)[-1,50]^2)),2)
```

```
## [1] 6.36
```

```
round(ridge.mod$lambda[60],2)
```

```
## [1] 705.48
```

```
round(coef(ridge.mod)[,60],2)
```

```
## (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##       54.33         0.11         0.66         1.18         0.94         0.85
##       Walks        Years       CAtBat        CHits       CHmRun        CRuns
##        1.32         2.60         0.01         0.05         0.34         0.09
##        CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##        0.10         0.07        13.68       -54.66         0.12         0.02
##      Errors   NewLeagueN
##       -0.70         8.61
```

```
round(sqrt(sum(coef(ridge.mod)[-1,60]^2)),2)
```

```
## [1] 57.11
```

```
# We extract now the regression coefficients with the 'predict' function
round(predict(ridge.mod,s=50,type="coefficients")[1:20,],2)
```

```
## (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##       48.77        -0.36         1.97        -1.28         1.15         0.80
##       Walks        Years       CAtBat        CHits       CHmRun        CRuns
##        2.72        -6.22         0.01         0.11         0.62         0.22
##        CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##        0.22        -0.15        45.93      -118.20         0.25         0.12
##      Errors   NewLeagueN
##       -3.28        -9.50
```

```
# Split randomly the dataset into 'train' and 'test' subsets
set.seed(1)
train<-sample(1:nrow(x), nrow(x)/2)
test<-(-train)
y.test<-y[test]
```

```
# Adjust model with the 'train' subset
ridge.mod<-glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh=1e-12)
#
# Then we evaluate prediction error (sum of squares) on the 'test' subset for three lambda values
# (lambda=4, lambda=1.0e10, lambda=0)
#
ridge.pred<-predict(ridge.mod,s=4,newx=x[test,])
round(mean((ridge.pred-y.test)^2),2)
```

```
## [1] 142199.1
```

```
# The model with no predictors (other than the intercept) has always a predicted value equal to the mea
# With a large lambda, the model tends to the no predictor one
```

```r
round(mean((mean(y[train])-y.test)^2),2)
```

```
## [1] 224669.9
```

```r
ridge.pred<-predict(ridge.mod,s=1e10,newx=x[test,])
round(mean((ridge.pred-y.test)^2))
```

```
## [1] 224670
```

```r
# With lambda equal to zero, the ridge regression model reduces to ordinary least squares
#
## Warning
#
# predict.glmnet with 'exact' computation requires re-entering the original training dataset
#
ridge.pred<-predict(ridge.mod,x=x[train,],y=y[train],s=0,newx=x[test,],exact=TRUE)
round(mean((ridge.pred-y.test)^2),2)
```

```
## [1] 168588.6
```

```r
# Same, with no 'exact' computation
#
ridge.pred<-predict(ridge.mod,s=0,newx=x[test,])
round(mean((ridge.pred-y.test)^2),2)
```

```
## [1] 167789.8
```

```r
# Compare an ordinary least squares regression with ridge regression with lambda=0
ols<-lm(Salary~.,data=Hitters, subset=train)
summary(ols)
```

```
##
## Call:
## lm(formula = Salary ~ ., data = Hitters, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -755.40 -172.21  -16.12  148.81 1709.58
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  274.0145   125.3304   2.186   0.0309 *
## AtBat         -0.3521     0.9547  -0.369   0.7130
## Hits          -1.6377     3.7435  -0.437   0.6626
## HmRun          5.8145     9.5466   0.609   0.5437
## Runs           1.5424     4.5241   0.341   0.7338
## RBI            1.1243     3.8265   0.294   0.7694
## Walks          3.7287     2.6005   1.434   0.1544
## Years        -16.3773    17.4006  -0.941   0.3487
## CAtBat        -0.6412     0.2499  -2.565   0.0116 *
## CHits          3.1632     1.1572   2.733   0.0073 **
## CHmRun         3.4008     2.9882   1.138   0.2575
## CRuns         -0.9739     1.1832  -0.823   0.4122
## CRBI          -0.6005     1.1839  -0.507   0.6130
## CWalks         0.3379     0.5657   0.597   0.5515
## LeagueN      119.1486   117.7810   1.012   0.3139
## DivisionW   -144.0831    55.8401  -2.580   0.0112 *
```

```
## PutOuts         0.1976     0.1078   1.833   0.0694 .
## Assists         0.6804     0.3054   2.228   0.0279 *
## Errors         -4.7128     6.4677  -0.729   0.4677
## NewLeagueN   -71.0951   117.4263  -0.605   0.5461
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 297 on 111 degrees of freedom
## Multiple R-squared:  0.5862, Adjusted R-squared:  0.5154
## F-statistic: 8.276 on 19 and 111 DF,  p-value: 7.206e-14
```

```
ols.yhat<-predict.lm(ols,newdata=Hitters[test,],type="response")
str(ols.yhat)
```

```
##  Named num [1:132] 763 1160 522 211 404 ...
##  - attr(*, "names")= chr [1:132] "-Alvin Davis" "-Andre Dawson" "-Andres Galarraga" "-Alfredo Griffi
```

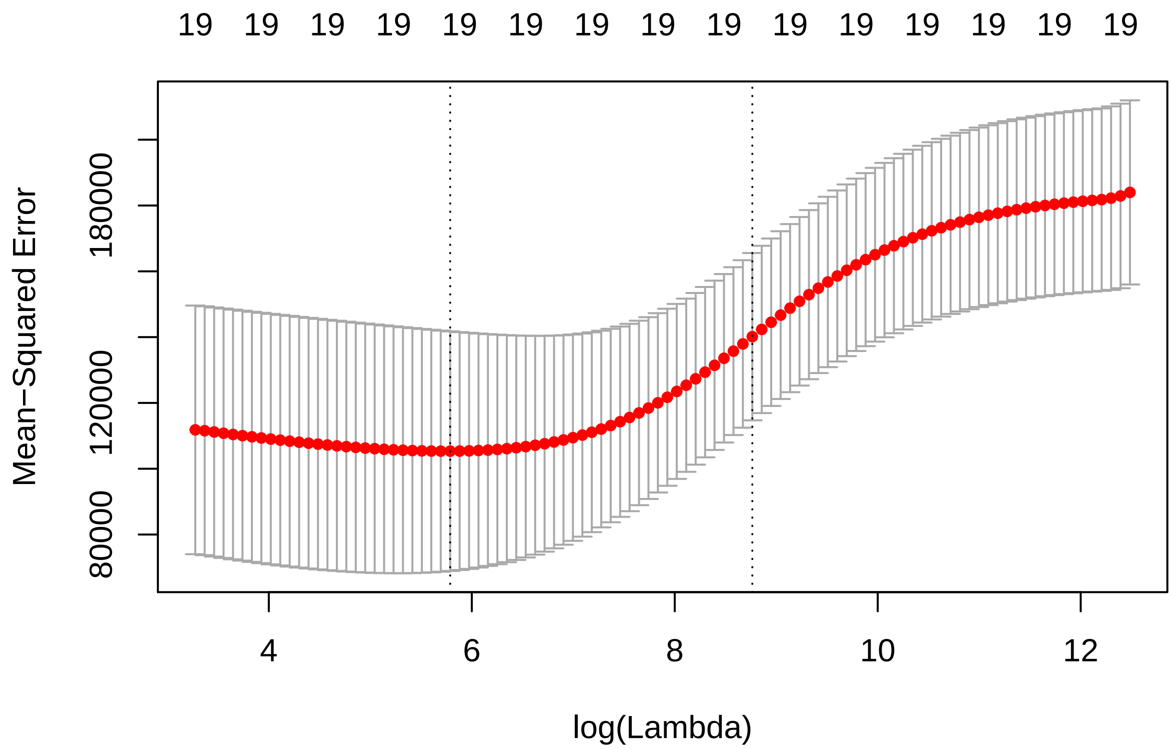
```
ols.residuals<-ols.yhat-y.test
round(mean(ols.residuals^2),2)
```

```
## [1] 168593.3
```

```
ridge.yhat<-predict(ridge.mod,x=x[train,],y=y[train],s=0,newx=x[test,],exact=TRUE)
#ridge.yhat<-predict(ridge.mod,s=0,newx=x[test,],type="response")
ridge.residuals<-ridge.yhat-y.test
round(mean(ridge.residuals^2),2)
```

```
## [1] 168588.6
```

```
# There is a k-fold cross-validation feature in the glmnet package which we can take advantege of
#
# By default k=10
set.seed(1)
cv.out<-cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.out)
```



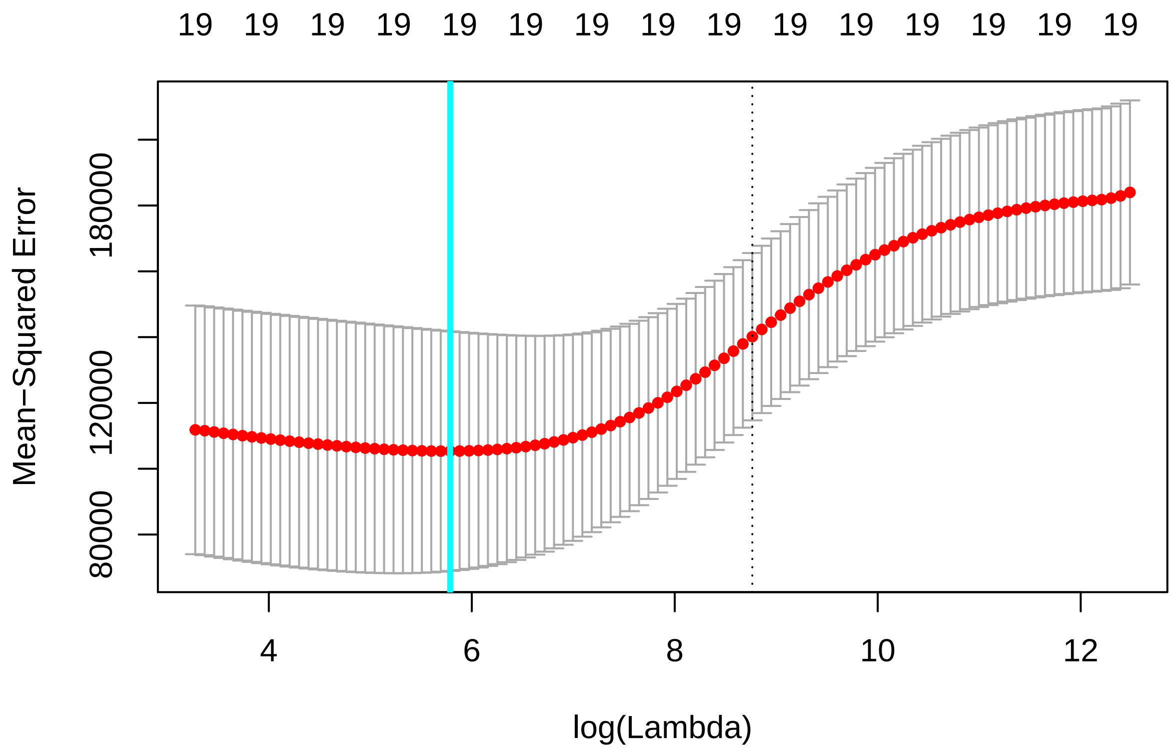

```r
bestlam<-cv.out$lambda.min
round(bestlam,3)
```

```
## [1] 326.083
```

```r
round(log(bestlam),3)
```

```
## [1] 5.787
```

```r
plot(cv.out)
abline(v=log(bestlam),lwd=3,col="cyan")
```




```r
# Mean quadratic error with the optimal lambda and the full dataset
#
# Coefficients of this model
#
# We observe that none of these coefficients is zero, hence there is no variable selectioin in ridge re
# To be compared with the lasso below.
ridge.pred<-predict(ridge.mod,s=bestlam,newx=x[test,])
round(mean((ridge.pred-y.test)^2),3)
```

```
## [1] 139856.6
```

```r
out<-glmnet(x,y,alpha=0)
round(predict(out,type="coefficients",s=bestlam)[1:20,],3)
```

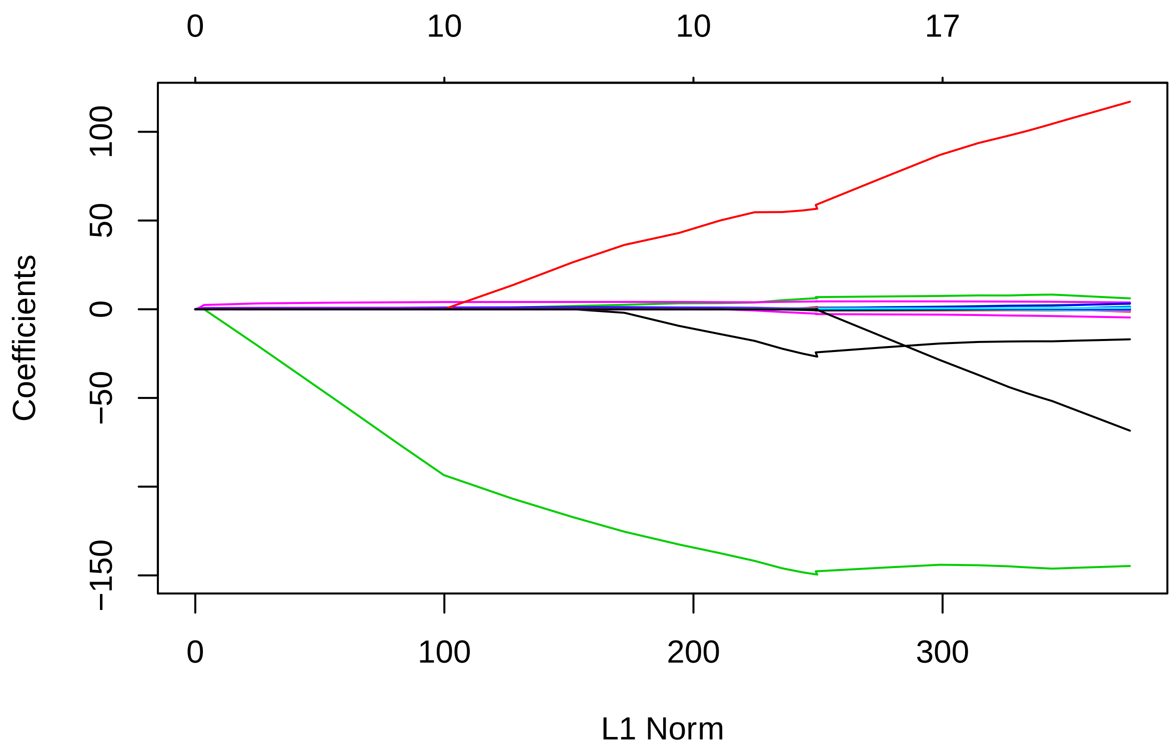
```
## (Intercept)       AtBat        Hits       HmRun        Runs         RBI
##      15.444       0.077       0.859       0.601       1.064       0.879
##       Walks       Years      CAtBat       CHits      CHmRun       CRuns
##       1.624       1.353       0.011       0.057       0.407       0.115
##        CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
##       0.121       0.053      22.091     -79.040       0.166       0.029
##      Errors  NewLeagueN
##      -1.361       9.125
```

A2. Regression with the *Lasso*

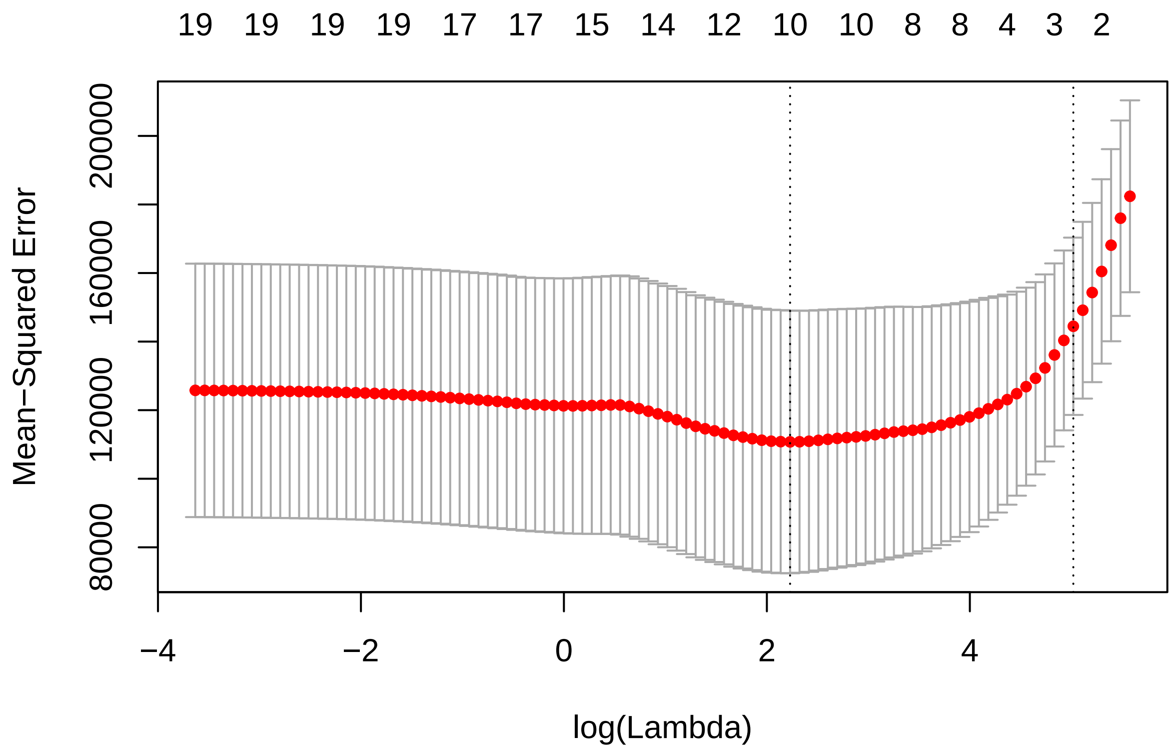
Same `Hitters` dataset as above and `glmnet`

```
# The same glmnet function performs lasso regression, setting the parameter alpha=1
#
lasso.mod<-glmnet(x[train,],y[train],alpha=1,lambda=grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```




```
set.seed(1)
cv.out<-cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
```




```
bestlam<-cv.out$lambda.min
```

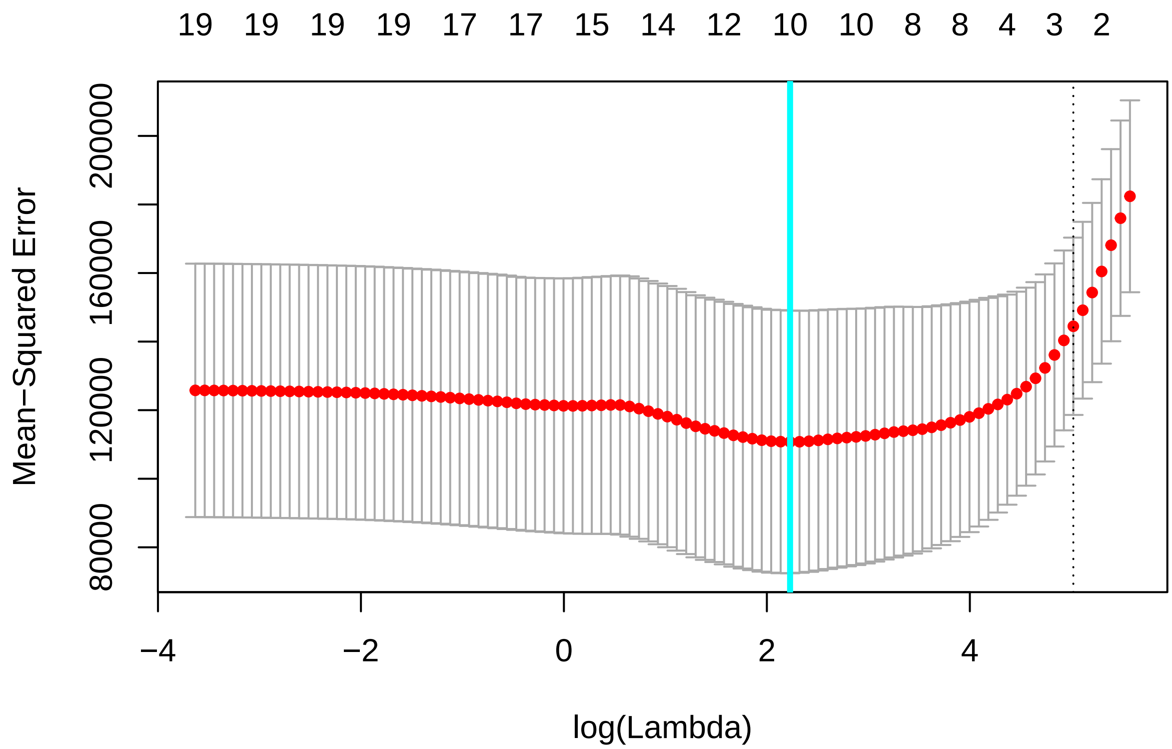
```
round(bestlam,3)
```

```
## [1] 9.287
```

```
round(log(bestlam),3)
```

```
## [1] 2.229
```

```
plot(cv.out)
abline(v=log(bestlam),lwd=3,col="cyan")
```




```
# Quadratic error on the test subset with the optimal lambda
lasso.pred<-predict(lasso.mod,s=bestlam,newx=x[test,])
round(mean((lasso.pred-y.test)^2),3)
```

```
## [1] 143673.6
```

The variable selection feature of the *Lasso*

```
# Quadratic error on the full dataset with the optimal lambda
# Coefficients in this model:
#
# Now we see there are zero coefficients: this is equivalent to discarding these variables.
#
# Compare with the ridge regression above
#
out<-glmnet(x,y,alpha=1,lambda=grid)
lasso.coef<-predict(out,type="coefficients",s=bestlam)[1:20,]
round(lasso.coef,3)
```

```
## (Intercept)       AtBat        Hits       HmRun        Runs         RBI
##       1.275      -0.055       2.180       0.000       0.000       0.000
##       Walks       Years      CAtBat       CHits      CHmRun       CRuns
##       2.292      -0.338       0.000       0.000       0.028       0.216
##        CRBI      CWalks     LeagueN   DivisionW     PutOuts     Assists
##       0.417       0.000      20.286    -116.168       0.238       0.000
##      Errors  NewLeagueN
##      -0.856       0.000
```

```r
round(lasso.coef[lasso.coef!=0],3)
```

```
## (Intercept)       AtBat        Hits       Walks       Years      CHmRun
##       1.275      -0.055       2.180       2.292      -0.338       0.028
##       CRuns        CRBI     LeagueN   DivisionW     PutOuts      Errors
##       0.216       0.417      20.286    -116.168       0.238      -0.856
```

B. Orthogonalization methods

Following ISLR - Cap 6 - Laboratory 3 - PCR and PLS

Codi de la web ISLR

```r
#install.packages("pls",dependencies=TRUE,repos="https://cloud.r-project.org")
require(pls)
```

```
## Loading required package: pls

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

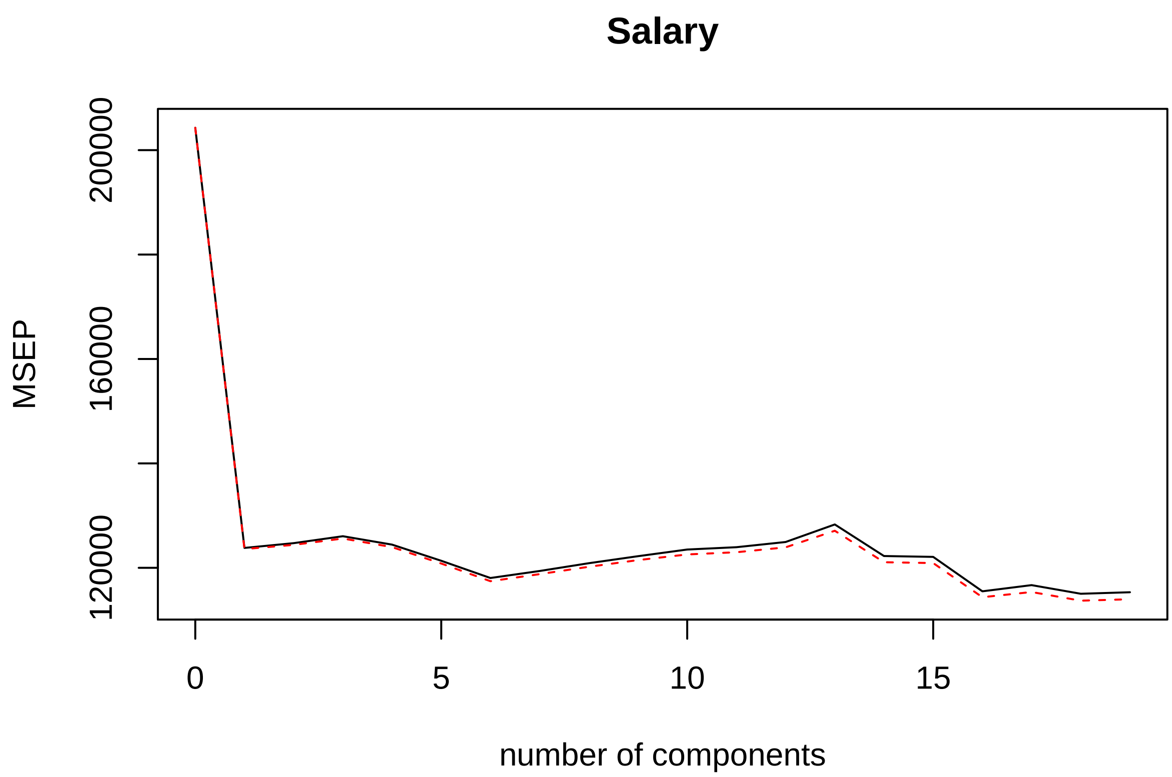
B1. Principal Components Regression (PCR)

```r
# Principal Components Regression
set.seed(2)
pcr.fit<-pcr(Salary~., data=Hitters,scale=TRUE,validation="CV")
summary(pcr.fit)
```

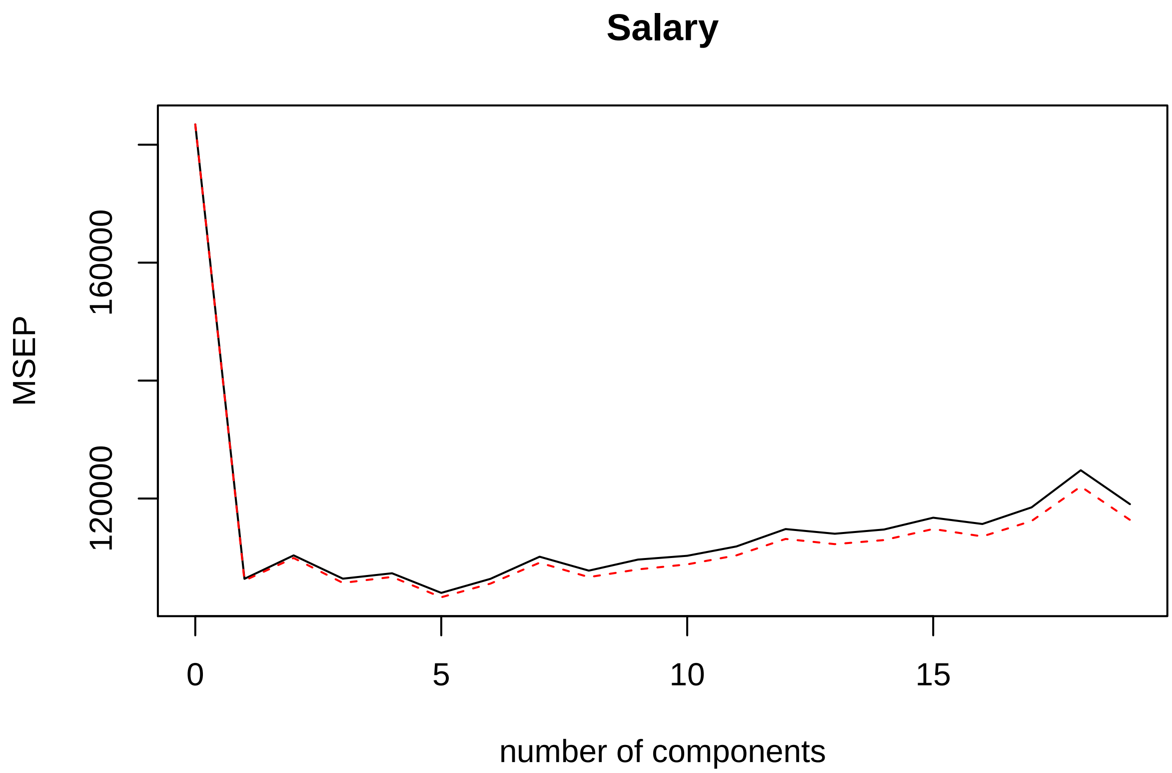
```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV             452    351.9    353.2    355.0    352.8    348.4    343.6
## adjCV          452    351.6    352.7    354.4    352.1    347.6    342.7
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       345.5    347.7    349.6     351.4     352.1     353.5     358.2
## adjCV    344.7    346.7    348.5     350.1     350.7     352.0     356.5
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        349.7     349.4     339.9     341.6     339.2     339.6
## adjCV     348.0     347.7     338.2     339.7     337.2     337.6
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         38.31    60.16    70.84    79.03    84.29    88.63    92.26
## Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69
##         8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
```

```
## X           94.96    96.28    97.26    97.98    98.65    99.15    99.47
## Salary      46.75    46.86    47.76    47.82    47.85    48.10    50.40
##          15 comps  16 comps  17 comps  18 comps  19 comps
## X           99.75     99.89     99.97     99.99    100.00
## Salary      50.55     53.01     53.85     54.61     54.61
```

```
validationplot(pcr.fit,val.type="MSEP")
```




```
# Cross-validation with hold-out
#
# Training the model, selecting number of principal components included in the model
set.seed(1)
pcr.fit<-pcr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")
validationplot(pcr.fit,val.type="MSEP")
```




```
# The minimum of the graph (optimal number of orthogonal variables) appears at 5 variables (principal c
# Fit the model for this number
pcr.pred<-predict(pcr.fit,x[test,],ncomp=5)
round(mean((pcr.pred-y.test)^2),3)
```

```
## [1] 142811.8
```

```
pcr.fit<-pcr(y~x,scale=TRUE,ncomp=5)
summary(pcr.fit)
```

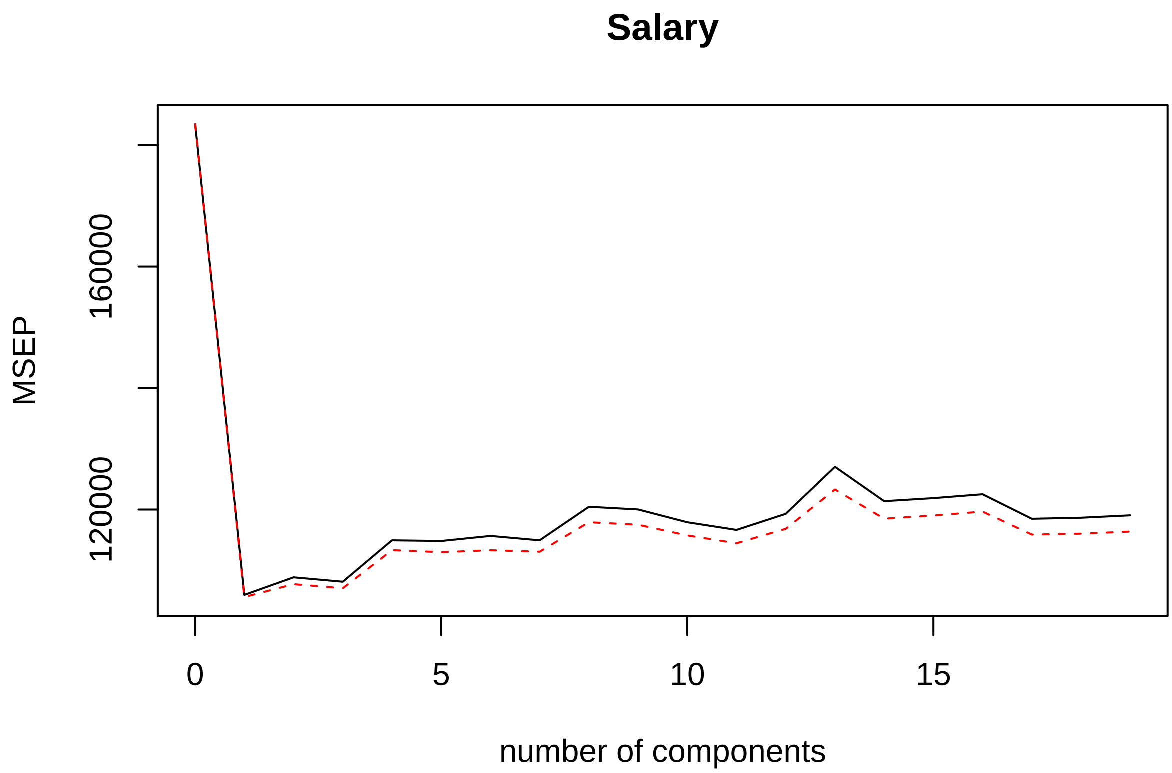
```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: svdpc
## Number of components considered: 5
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps  4 comps  5 comps
## X    38.31    60.16    70.84    79.03    84.29
## y    40.63    41.58    42.17    43.22    44.90
```

B2. Partial Least Squares (PLS)

```
# Partial Least Squares
set.seed(1)
pls.fit<-plsr(Salary~., data=Hitters,subset=train,scale=TRUE, validation="CV")
summary(pls.fit)
```

```
## Data:    X dimension: 131 19
##  Y dimension: 131 1
## Fit method: kernelpls
## Number of components considered: 19
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           428.3    325.5    329.9    328.8    339.0    338.9    340.1
## adjCV        428.3    325.0    328.2    327.2    336.6    336.1    336.6
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       339.0    347.1    346.4     343.4     341.5     345.4     356.4
## adjCV    336.2    343.4    342.8     340.2     338.3     341.8     351.1
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps
## CV        348.4     349.1     350.0     344.2     344.5     345.0
## adjCV     344.2     345.0     345.9     340.4     340.6     341.1
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         39.13    48.80    60.09    75.07    78.58    81.12    88.21
## Salary    46.36    50.72    52.23    53.03    54.07    54.77    55.05
##         8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X         90.71    93.17     96.05     97.08     97.61     97.97     98.70
## Salary    55.66    55.95     56.12     56.47     56.68     57.37     57.76
##         15 comps  16 comps  17 comps  18 comps  19 comps
## X          99.12     99.61     99.70     99.95    100.00
## Salary     58.08     58.17     58.49     58.56     58.62
```

```r
validationplot(pls.fit,val.type="MSEP")
```




```
# The minimum of the graph (optimal number of orthogonal variables) appears at 2 variables.
# Fit the model for this number
pls.pred<-predict(pls.fit,x[test,],ncomp=2)
round(mean((pls.pred-y.test)^2),3)
```

```
## [1] 145367.7
```

```
pls.fit<-plsr(Salary~., data=Hitters,scale=TRUE,ncomp=2)
summary(pls.fit)
```

```
## Data:    X dimension: 263 19
##  Y dimension: 263 1
## Fit method: kernelpls
## Number of components considered: 2
## TRAINING: % variance explained
##         1 comps  2 comps
## X         38.08    51.03
## Salary    43.05    46.40
```