Regularization with R

Data Mining
Master Degree in Computer Science
University of Padova

a.y. 2017/2018

Annamaria Guolo

1 Hitters dataset

Consider dataset `Hitters` inside package `ISLR` we have already used when talking about model selection.

```
library(ISLR)
data(Hitters)
## omit missing data
hitters <- na.omit(Hitters)
dim(hitters)

## [1] 263  20

names(hitters)

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

hitters$Salary <- log(hitters$Salary)
## using the transformation applied in the previous analysis of the data
```

Consider a model relating the seasonal income to the other covariates.
Ridge regression and lasso can be applied using functions inside library `glmnet`.

```
library(glmnet)
```

Start with ridge regression. We estimate the model through function `glmnet()`, which needs the matrix of covariates `X` and the vector of observations from `y`. Parameter `alpha` is set to 0 (indicator of ridge regression).

```r
y <- hitters$Salary
X <- model.matrix(Salary ~ ., data=hitters)[,-1]
```

Function `model.matrix()` creates the matrix with covariates and in the meanwhile it transforms qualitative variables into dummies. Remember to eliminate the first column corresponding to the intercept.

```r
m.ridge <- glmnet(X, y, alpha=0)
```

Output

```r
m.ridge

##
## Call:  glmnet(x = X, y = y, alpha = 0)
##
##        Df      %Dev    Lambda
##   [1,] 19 7.070e-36 551.20000
##   [2,] 19 1.214e-02 502.20000
##   [3,] 19 1.330e-02 457.60000
##   [4,] 19 1.456e-02 417.00000
##   [5,] 19 1.595e-02 379.90000
##   [6,] 19 1.746e-02 346.20000
##   [7,] 19 1.911e-02 315.40000
##   [8,] 19 2.091e-02 287.40000
##   [9,] 19 2.288e-02 261.90000
##  [10,] 19 2.503e-02 238.60000
##  [11,] 19 2.736e-02 217.40000
##  [12,] 19 2.990e-02 198.10000
##  [13,] 19 3.267e-02 180.50000
##  [14,] 19 3.568e-02 164.50000
##  [15,] 19 3.895e-02 149.90000
##  [16,] 19 4.249e-02 136.50000
##  [17,] 19 4.634e-02 124.40000
##  [18,] 19 5.050e-02 113.40000
##  [19,] 19 5.500e-02 103.30000
##  [20,] 19 5.985e-02  94.11000
##  [21,] 19 6.509e-02  85.75000
##  [22,] 19 7.073e-02  78.13000
##  [23,] 19 7.679e-02  71.19000
##  [24,] 19 8.329e-02  64.87000
##  [25,] 19 9.025e-02  59.11000
##  [26,] 19 9.768e-02  53.85000
##  [27,] 19 1.056e-01  49.07000
##  [28,] 19 1.140e-01  44.71000
##  [29,] 19 1.229e-01  40.74000
##  [30,] 19 1.324e-01  37.12000
```

```
##  [31,] 19 1.423e-01  33.82000
##  [32,] 19 1.528e-01  30.82000
##  [33,] 19 1.637e-01  28.08000
##  [34,] 19 1.751e-01  25.59000
##  [35,] 19 1.869e-01  23.31000
##  [36,] 19 1.991e-01  21.24000
##  [37,] 19 2.117e-01  19.35000
##  [38,] 19 2.247e-01  17.63000
##  [39,] 19 2.378e-01  16.07000
##  [40,] 19 2.512e-01  14.64000
##  [41,] 19 2.647e-01  13.34000
##  [42,] 19 2.783e-01  12.16000
##  [43,] 19 2.918e-01  11.08000
##  [44,] 19 3.053e-01  10.09000
##  [45,] 19 3.186e-01   9.19500
##  [46,] 19 3.317e-01   8.37800
##  [47,] 19 3.444e-01   7.63400
##  [48,] 19 3.568e-01   6.95600
##  [49,] 19 3.687e-01   6.33800
##  [50,] 19 3.802e-01   5.77500
##  [51,] 19 3.912e-01   5.26200
##  [52,] 19 4.016e-01   4.79400
##  [53,] 19 4.114e-01   4.36800
##  [54,] 19 4.206e-01   3.98000
##  [55,] 19 4.293e-01   3.62700
##  [56,] 19 4.374e-01   3.30400
##  [57,] 19 4.448e-01   3.01100
##  [58,] 19 4.518e-01   2.74300
##  [59,] 19 4.582e-01   2.50000
##  [60,] 19 4.641e-01   2.27800
##  [61,] 19 4.695e-01   2.07500
##  [62,] 19 4.745e-01   1.89100
##  [63,] 19 4.790e-01   1.72300
##  [64,] 19 4.832e-01   1.57000
##  [65,] 19 4.871e-01   1.43000
##  [66,] 19 4.906e-01   1.30300
##  [67,] 19 4.939e-01   1.18800
##  [68,] 19 4.969e-01   1.08200
##  [69,] 19 4.997e-01   0.98590
##  [70,] 19 5.023e-01   0.89830
##  [71,] 19 5.048e-01   0.81850
##  [72,] 19 5.070e-01   0.74580
##  [73,] 19 5.091e-01   0.67960
##  [74,] 19 5.111e-01   0.61920
##  [75,] 19 5.130e-01   0.56420
##  [76,] 19 5.148e-01   0.51410
##  [77,] 19 5.165e-01   0.46840
```

```
##  [78,] 19 5.181e-01   0.42680
##  [79,] 19 5.197e-01   0.38890
##  [80,] 19 5.212e-01   0.35430
##  [81,] 19 5.227e-01   0.32280
##  [82,] 19 5.241e-01   0.29420
##  [83,] 19 5.255e-01   0.26800
##  [84,] 19 5.268e-01   0.24420
##  [85,] 19 5.281e-01   0.22250
##  [86,] 19 5.294e-01   0.20280
##  [87,] 19 5.307e-01   0.18470
##  [88,] 19 5.319e-01   0.16830
##  [89,] 19 5.331e-01   0.15340
##  [90,] 19 5.343e-01   0.13980
##  [91,] 19 5.355e-01   0.12730
##  [92,] 19 5.366e-01   0.11600
##  [93,] 19 5.377e-01   0.10570
##  [94,] 19 5.388e-01   0.09633
##  [95,] 19 5.399e-01   0.08777
##  [96,] 19 5.409e-01   0.07997
##  [97,] 19 5.420e-01   0.07287
##  [98,] 19 5.430e-01   0.06639
##  [99,] 19 5.439e-01   0.06050
## [100,] 19 5.449e-01   0.05512
```

The output reports the value fo the deviance for each value of λ. Actually, the estimated object includes many other quantities

```
names(m.ridge)

##  [1] "a0"         "beta"       "df"         "dim"        "lambda"     "dev.ratio" "nulldev"
##  [8] "npasses"    "jerr"       "offset"     "call"       "nobs"
```

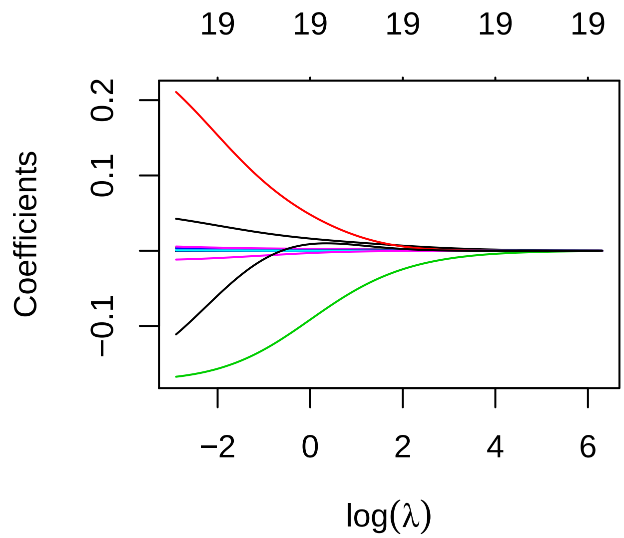
- `a0`: estimated intercept for each model fitted with a different λ
- `beta`: $p \times$ (number of λ) matrix with the estimates of the coefficients
- `lambda`: values of λ
- `dev.ratio`: 1- model deviance/null deviance;
- `nulldev`: null deviance

How many λ are considered?

```
length(m.ridge$lambda)

## [1] 100
```

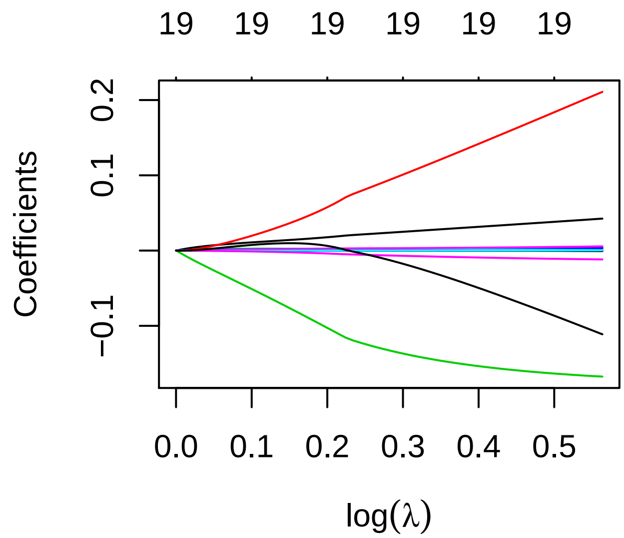
Graphical evaluation of the coefficients associated to the covariates

```r
plot(m.ridge, xvar='lambda', xlab=expression(log(lambda)))
```



Option `xvar='lambda'` specifies that the x-axis is expressed in terms of λ. Alternatives are deviance values and L1-norm values. See, for example

```r
plot(m.ridge, xlab=expression(log(lambda)))
```



Option `xlab=expression(log(lambda))` insert the mathematical symbol for λ in the axis. Numbers (19, repeated) over the graph indicate the number of covariates entering the model as λ varies: 19 is repeated, as ridge regression is not a selection method.
Look for the best λ using cross validation, using function `cv.glmnet()`, with a syntax similar to that in `glmnet()`.
Fix the seed

```r
set.seed(2906)
```

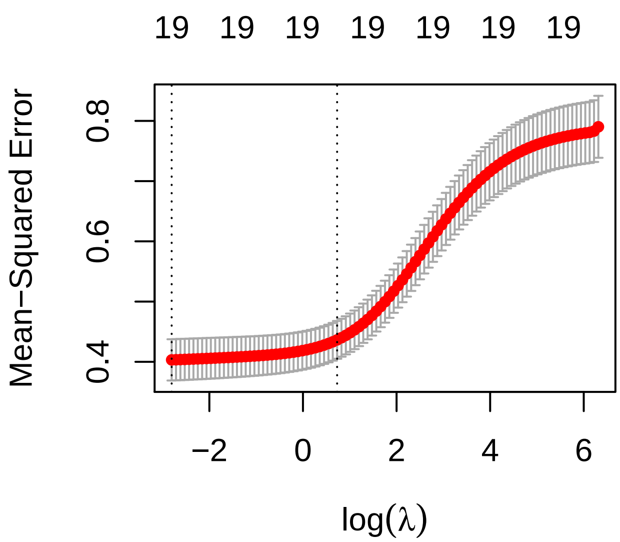
```r
cv.ridge <- cv.glmnet(X, y, alpha=0)
```

For default, R considers 10−fold cross validation. The resulting object includes several quantities, such as

- `lambda`: values of λ
- `cvm`: the MSE for each λ
- `cvsd`: the estimate of the standard error of `cvm`
- `lambda.min`: the value of λ associated to the minimum `cvm`
- `lambda.1se`: the values of λ associated to the minimum `cvm` within 1 standard error.

Graphical representation

```r
plot(cv.ridge, xlab=expression(log(lambda)))
```



The plots shows the values of `cvm` for each $\log(\lambda)$ together with the associated confidence interval. The two dashed lines are the values of log-`lambda.min` and log-`lambda.1se`.
λ from cross validation

```r
best.lambda <- cv.ridge$lambda.min
best.lambda

## [1] 0.060496
```

Find the minimum MSE

```
cv.ridge$cvm[cv.ridge$lambda==best.lambda]

## [1] 0.403175

## or, equivalently,
min(cv.ridge$cvm)

## [1] 0.403175
```

Re-estimate the model using the best λ

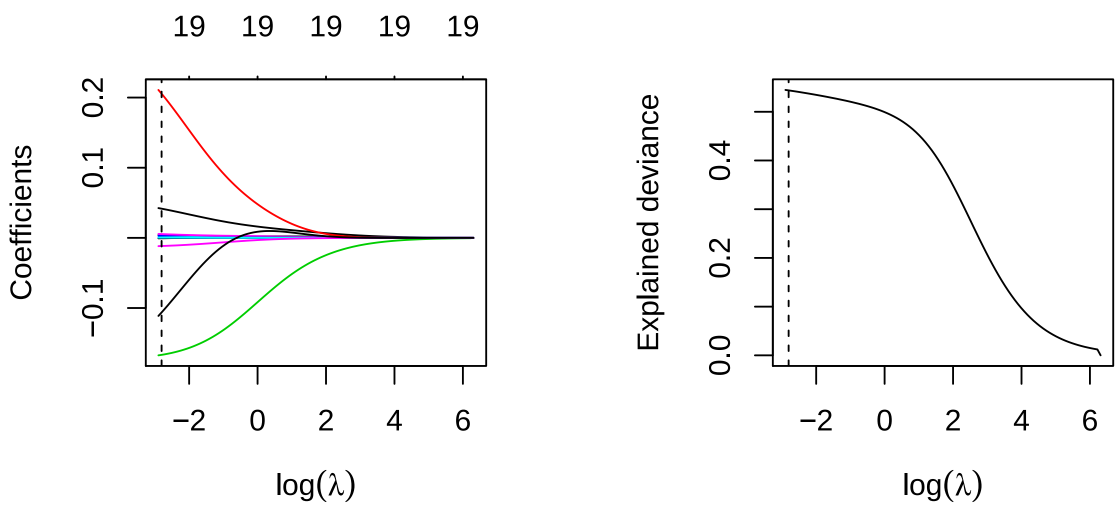
```
m.ridge.min <- glmnet(X, y, alpha=0, lambda=best.lambda)
m.ridge.min

##
## Call:  glmnet(x = X, y = y, alpha = 0, lambda = best.lambda)
##
##      Df   %Dev Lambda
## [1,] 19 0.5439 0.0605
```

Coefficients of the model

```
coef(m.ridge.min)

## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)  4.595305e+00
## AtBat       -5.160388e-04
## Hits         4.396714e-03
## HmRun        3.359157e-03
## Runs         3.110850e-03
## RBI          7.308564e-04
## Walks        5.371948e-03
## Years        4.171532e-02
## CAtBat       3.892745e-05
## CHits        1.913195e-04
## CHmRun      -5.602232e-05
## CRuns        2.972823e-04
## CRBI         1.652623e-04
## CWalks      -3.891595e-04
## LeagueN      2.052706e-01
## DivisionW   -1.666745e-01
## PutOuts      2.925873e-04
## Assists      3.949817e-04
## Errors      -1.161143e-02
## NewLeagueN  -1.060981e-01
```

Graphical representation of the coefficients for the best λ and model deviance

```r
par(mfrow=c(1,2))
plot(m.ridge, xvar='lambda', xlab=expression(log(lambda)))
## add on the line corresponding to the best lambda
abline(v=log(best.lambda), lty=2)
## deviance
plot(log(m.ridge$lambda), m.ridge$dev.ratio, type='l',
        xlab=expression(log(lambda)), ylab='Explained deviance')
abline(v=log(best.lambda), lty=2)
```



The maximum explained deviance is obtained for the minimum (best) λ and it is equal to

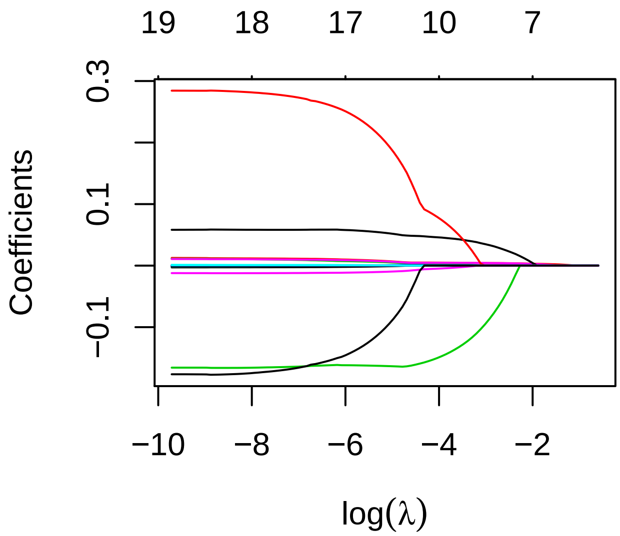
```r
max(m.ridge$dev.ratio)

## [1] 0.5448529
```

Now move to lasso. The syntax is similar to that for ridge regression, but specifying `alpha=1`

```r
m.lasso <- glmnet(X, y, alpha=1)
```

Graphical representation of the coefficients

```r
plot(m.lasso, xvar='lambda', xlab=expression(log(lambda)))
```



Look for λ that minimizes the MSE

```
## fix the seed to the same value used for ridge regression
set.seed(2906)
cv.lasso <- cv.glmnet(X, y, alpha=1)
```

Minimum λ from cross validation

```
best.lambda.lasso <- cv.lasso$lambda.min
```

Minimum MSE

```
min(cv.lasso$cvm)

## [1] 0.4026953
```

On the basis of MSE, the model fitted with lasso is preferable. In addition, the resulting model with lasso is simplest.
Re-estimate the model using the best λ from cross-validation

```
m.lasso.min <- glmnet(X, y, alpha=1, lambda=best.lambda.lasso)
```

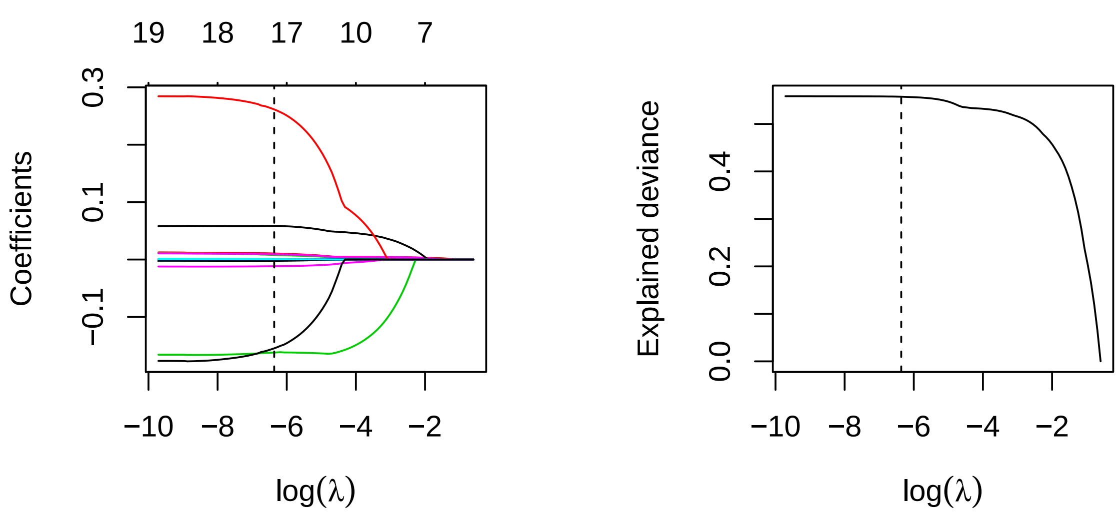
Coefficients

```
coef(m.lasso.min)

## 20 x 1 sparse Matrix of class "dgCMatrix"
##                         s0
## (Intercept)   4.593462e+00
## AtBat        -2.311993e-03
## Hits          1.056073e-02
```

```
## HmRun        7.973244e-03
## Runs          .
## RBI         -2.285376e-04
## Walks        9.294713e-03
## Years        5.881411e-02
## CAtBat       2.017179e-05
## CHits        4.843161e-05
## CHmRun       2.194080e-04
## CRuns        1.068970e-03
## CRBI          .
## CWalks      -1.089961e-03
## LeagueN      2.615124e-01
## DivisionW   -1.616904e-01
## PutOuts      3.204166e-04
## Assists      5.616621e-04
## Errors      -1.166310e-02
## NewLeagueN  -1.543706e-01
```

Some of the coefficients are zero, so the lasso performed a model selection.
Graphical representation of the coefficients for the best λ and model deviance

```
par(mfrow=c(1,2))
plot(m.lasso, xvar='lambda', xlab=expression(log(lambda)))
## add on the line corresponding to the best lambda
abline(v=log(best.lambda.lasso), lty=2)
## deviance
plot(log(m.lasso$lambda), m.lasso$dev.ratio, type='l',
        xlab=expression(log(lambda)), ylab='Explained deviance')
abline(v=log(best.lambda.lasso), lty=2)
```



The maximum explained deviance is obtained for the minimum (best) λ and it is equal to

```r
max(m.lasso$dev.ratio)

## [1] 0.5584737
```

2 Leukemia dataset

Consider the Leukemia data about the gene expression in cancer cells obtained from 72 subjects with acute myeloid leukemia and acute lymphoblastic leukemia.
Data are available in the R workspace *Leukemia.RData*. Upload the data

```r
load("Leukemia.RData")
ls()

##  [1] "best.lambda"       "best.lambda.lasso" "cv.lasso"          "cv.ridge"
##  [5] "hitters"           "Hitters"           "Leukemia"          "m.lasso"
##  [9] "m.lasso.min"       "m.ridge"           "m.ridge.min"       "X"
## [13] "y"

names(Leukemia)

## [1] "x" "y"
```

There are the objects x and y. Response y is categorical (diseased/nondiseased)

```r
table(Leukemia$y)

##
##  0  1
## 47 25
```

Object x is the matrix with the observations for the covariates

```r
dim(Leukemia$x)

## [1]   72 3571
```

There are 3571 covariates. Clearly, a standard logistic regression model cannot work here. In fact, look at the output

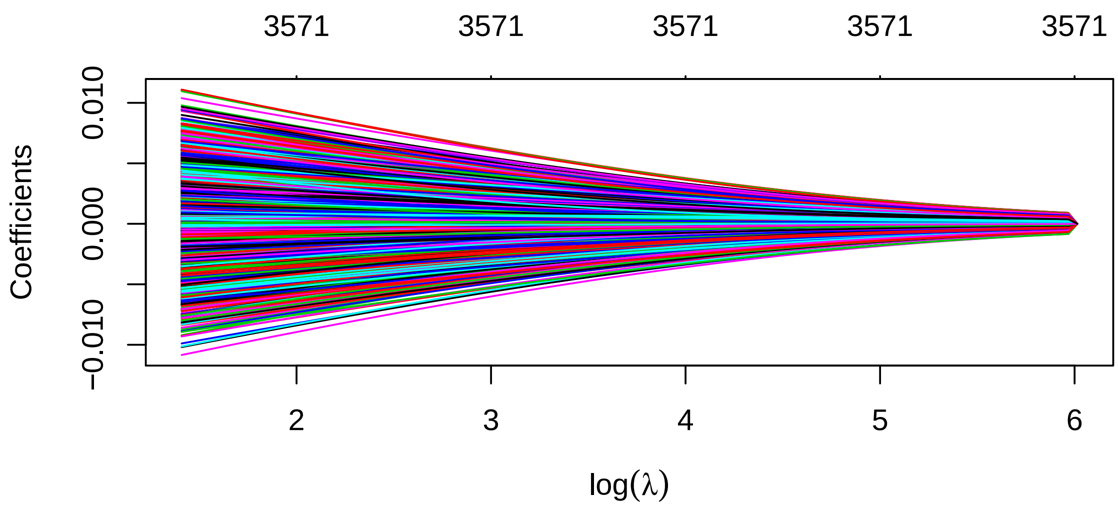
```r
m <- glm(y ~ x, data=Leukemia, family='binomial')
```

here not reported for space reason. Use ridge regression and lasso instead.
Start with ridge regression

```
leukemia.ridge <- glmnet(Leukemia$x, Leukemia$y, alpha=0, family='binomial')
```

Note that we specify `family='binomial'` as y is a binary indicator.

```
plot(leukemia.ridge, xvar='lambda', xlab=expression(log(lambda)))
```



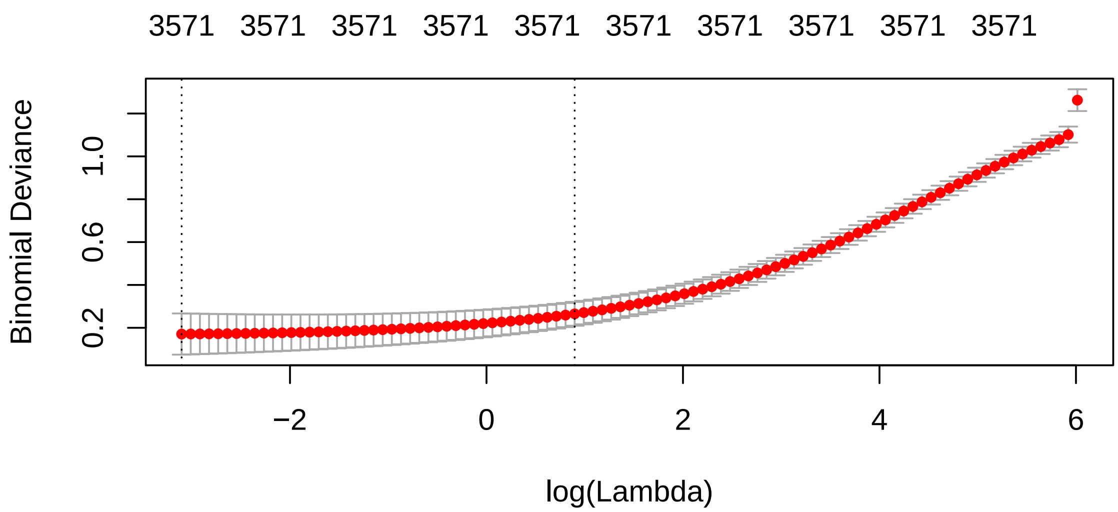
Which values of λ are used?

```
summary(leukemia.ridge$lambda)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.093  12.946  40.942  89.195 129.462 409.310
```

Select the best value of λ using cross validation, previously extending the grid of values of λ through option `lambda.min`

```
set.seed(111)
cv.leukemia.ridge <- cv.glmnet(Leukemia$x, Leukemia$y, alpha=0, family='binomial',
        lambda.min = 1e-4)
best.lambda.leukemia <- cv.leukemia.ridge$lambda.min
best.lambda.leukemia

## [1] 0.0449217

min(cv.leukemia.ridge$cvm)

## [1] 0.1708827
```

```
plot(cv.leukemia.ridge)
```



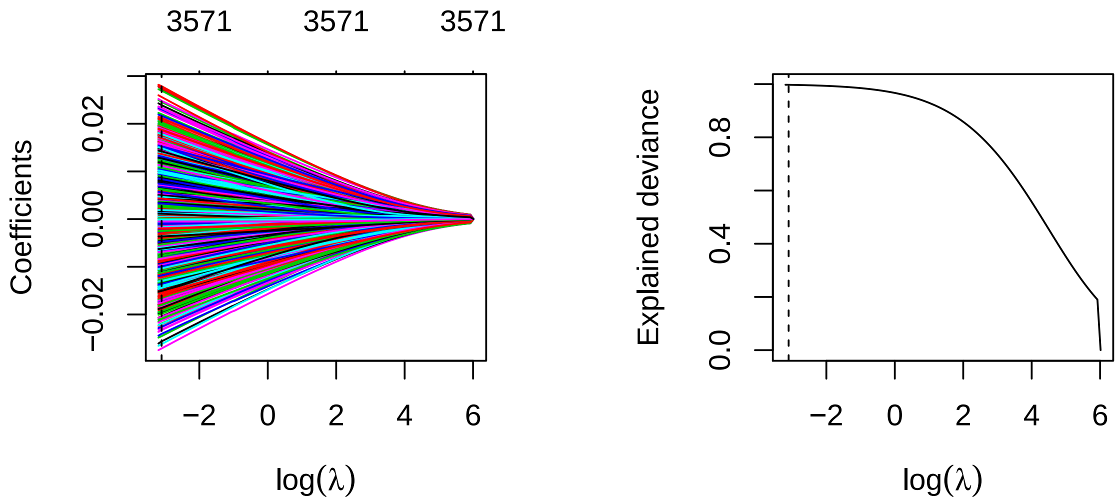
Re-estimate the model using the best λ chosen from cross-validation

```
leukemia.ridge.min <- glmnet(Leukemia$x, Leukemia$y, alpha=0, family='binomial',
        lambda=best.lambda.leukemia)
```

Graphical representation of the coefficients for the best λ and model deviance. Remember to extend the grid of values of λ

```
leukemia.ridge <- glmnet(Leukemia$x, Leukemia$y, alpha=0, family='binomial',
        lambda.min = 1e-4)
```

```
par(mfrow=c(1,2))
plot(leukemia.ridge, xvar='lambda', xlab=expression(log(lambda)))
## add on the line corresponding to the best lambda
abline(v=log(best.lambda.leukemia), lty=2)
## deviance
plot(log(leukemia.ridge$lambda), leukemia.ridge$dev.ratio, type='l',
        xlab=expression(log(lambda)), ylab='Explained deviance')
abline(v=log(best.lambda.leukemia), lty=2)
```



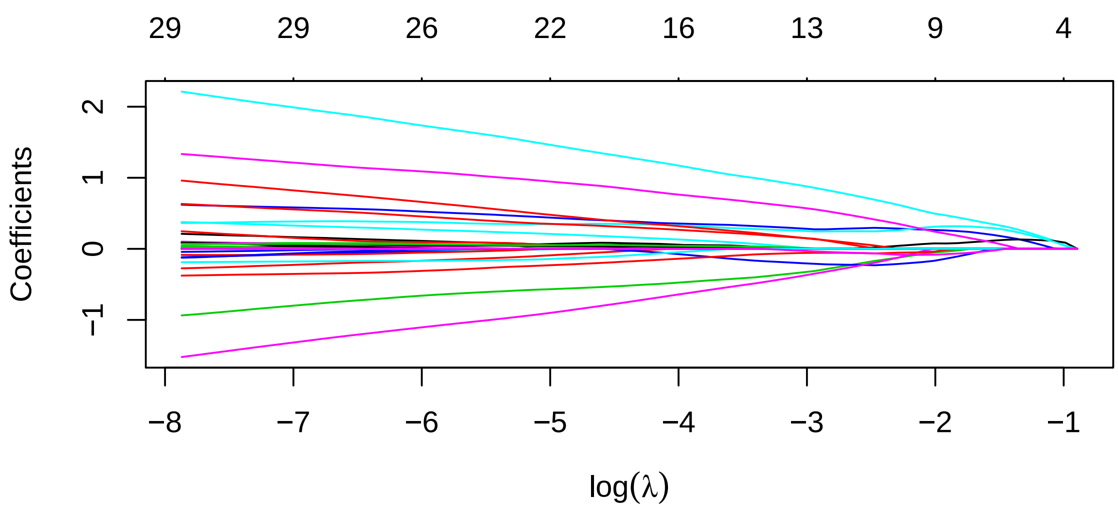
The maximum explained deviance is obtained for the minimum (best) λ and it is equal to

```
max(leukemia.ridge$dev.ratio)

## [1] 0.9975967
```

Lasso

```
leukemia.lasso <- glmnet(Leukemia$x, Leukemia$y, alpha=1, family='binomial',
          lambda.min = 1e-4)
```

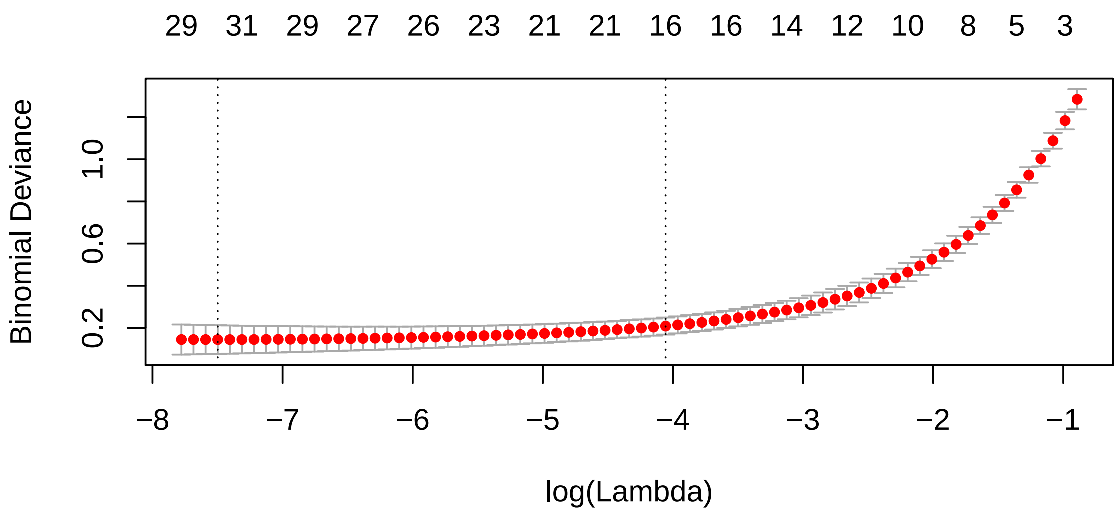
```
plot(leukemia.lasso, xvar='lambda', xlab=expression(log(lambda)))
```



Select λ from cross validation

```r
set.seed(111)
cv.leukemia.lasso <- cv.glmnet(Leukemia$x, Leukemia$y, alpha=1, family='binomial',
        lambda.min = 1e-4)
best.lambda.leukemia.lasso <- cv.leukemia.lasso$lambda.min
best.lambda.leukemia.lasso

## [1] 0.0005538157

min(cv.leukemia.lasso$cvm)

## [1] 0.1443071
```

```r
plot(cv.leukemia.lasso)
```



Re-estimate the model using the best λ from cross-validation

```r
leukemia.lasso.min <- glmnet(Leukemia$x, Leukemia$y, alpha=1, family='binomial',
        lambda=best.lambda.leukemia.lasso)
```

Pay attention to the different time consumption...
How many coefficients are set equal to zero? None in ridge regression.

```r
id.zero <- which(coef(leukemia.lasso.min)==0)
length(id.zero)

## [1] 3541

nonzero <- length(coef(leukemia.lasso.min))-length(id.zero)
nonzero

## [1] 31
```

There are 3541 zero coefficients, so lasso selects only 30 variables (we need to eliminate the intercept). The chosen variables are

```
id.nonzero <- which(coef(leukemia.lasso.min)!=0)
varnames <- rownames(coef(leukemia.lasso.min))[id.nonzero]
values <- coef(leukemia.lasso.min)[id.nonzero]
names(values) <- varnames
values
```

```
##   (Intercept)          V158          V219          V456          V657          V672
## -2.8824214763  0.0018961082 -0.2285781638 -0.9231795133 -0.1237991380 -1.4074641347
##          V888          V918          V926          V956          V979         V1007
##  0.1704091156  0.2085950745  0.0317573289  0.6160689383  2.0840545978  0.0354589050
##         V1219         V1569         V1652         V1796         V1835         V1946
## -0.3447530778  0.0101644338  0.3413735697  0.0001313399  0.0646048792  0.9194201601
##         V2230         V2239         V2481         V2727         V2831         V2859
##  0.1121487684 -0.1144706536  1.2448989053 -0.1306291781  0.0066234777 -0.0629515620
##         V2888         V2929         V3038         V3098         V3125         V3158
##  0.3532643009  0.0902908966  0.1319079465  0.5965100779  0.0142853358  0.0783746529
##         V3181
## -0.1169728169
```

Try to see what happens when using λ equal to `lambda.1se` in place of `lambda.min` from cross-validation.