LAB 12

TOPICS:

- Linear models

```
library(MASS)
library(car)
```

```
library(rgl)
library(glmnet)
```

1.

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Problem of collinearity
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Example 1: Multiple linear regression
### ------------------------------------------------------------------------
### Dataset cars: distance taken to stop [ft] as a function of velocity [mph]
### for some cars in the 1920s

cars
```

```
##    speed dist
## 1      4    2
## 2      4   10
## 3      7    4
## 4      7   22
## 5      8   16
## 6      9   10
## 7     10   18
## 8     10   26
## 9     10   34
## 10    11   17
## 11    11   28
## 12    12   14
## ..
## 41    20   52
## 42    20   56
## 43    20   64
## 44    22   66
## 45    23   54
## 46    24   70
## 47    24   92
## 48    24   93
## 49    24  120
## 50    25   85
```

```
plot(cars, xlab='Speed', ylab='Stopping distance', las=1)

n          <- dim(cars)[[1]]
distance   <- cars$dist
speed1     <- cars$speed
speed2     <- cars$speed^2

### Model:
### distance = beta_0 + beta_1 * speed + beta_2 * speed^2 + Eps
### (linear in the parameters!)

fm <- lm(distance ~ speed1 + speed2)
summary(fm)
```

(see the plot later on)

Goal: predict the stopping distance using the speed and speed^2

$$Y = \beta_0 + \beta_1 \text{speed} + \beta_2 \text{speed}^2 + \varepsilon$$

```
##
## Call:
## lm(formula = distance ~ speed1 + speed2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.720  -9.184  -3.188   4.628  45.152
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.47014   14.81716   0.167    0.868
## speed1       0.91329    2.03422   0.449    0.656
## speed2       0.09996    0.06597   1.515    0.136
##
## Residual standard error: 15.18 on 47 degrees of freedom
## Multiple R-squared:  0.6673, Adjusted R-squared:  0.6532
## F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

high p-values due to collinearity of the variables

```
# Note: collinearity

# Variance inflation factor
help(vif)
```

(to check collinearity)

```
vif(fm)
```

```
##   speed1   speed2
## 24.61489 24.61489
```

```
# recall vif formula:
1/(1- summary(lm(speed1 ~ speed2))$r.squared )
```

very high (>10 is considered high)

```
## [1] 24.61489
```

```
1/(1- summary(lm(speed2 ~ speed1))$r.squared )
```

```
## [1] 24.61489
```

```
### A possible solution to collinearity: PCA
speed.pc <- princomp(cbind(speed1,speed2), scores=TRUE)
summary(speed.pc)
```

```
## Importance of components:
##                           Comp.1       Comp.2
## Standard deviation     161.4944487 1.054526e+00
## Proportion of Variance   0.9999574 4.263644e-05
## Cumulative Proportion    0.9999574 1.000000e+00
```

2 principal components since the dimension of the space is 2

we see that the first principal component explains almost all the variability in our dataset

```
speed.pc$load
```

```
##
## Loadings:
##        Comp.1 Comp.2
## speed1         0.999
## speed2 0.999
##
##                Comp.1 Comp.2
## SS loadings       1.0    1.0
## Proportion Var    0.5    0.5
## Cumulative Var    0.5    1.0
```

component 1 corresponds basically to speed 2

component 2 corresponds basically to speed 1

⇓

we will end up in an analysis that tells us to use only component 1

Even if we notice that we should use just the first principal component, let's use them both:

extraction of the scores of the 2 PCA (we obtain the projection of the original data on PC1 and PC2)

```
sp1.pc <- speed.pc$scores[,1]
sp2.pc <- speed.pc$scores[,2]

# Now we estimate the model by inserting the PCs instead of the
# original regressors
# Model: y = b0 + b1*PC1+ b2*PC2 + eps, eps~N(0,sigma^2)
fm.pc <- lm(distance ~ sp1.pc + sp2.pc)

summary(fm.pc)
```

$$Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \varepsilon$$

we fit the model with both the principal components

```
##
## Call:
## lm(formula = distance ~ sp1.pc + sp2.pc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.720  -9.184  -3.188   4.628  45.152
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.98000    2.14622  20.026  < 2e-16 ***   ✓
## sp1.pc       0.12890    0.01329   9.700 8.49e-13 ***   ✓
## sp2.pc       0.98965    2.03525   0.447    0.657       ✗
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.18 on 47 degrees of freedom
## Multiple R-squared:  0.6673, Adjusted R-squared:  0.6532   ✓
## F-statistic: 47.14 on 2 and 47 DF,  p-value: 5.852e-12
```

```
# Note: we could have performed dimensionality reduction before
# estimating the model and then considered only the first PC.

# We can re-write the model as:
# Model:
# y= b0 + b1*      PC1                + b2*      PC2                + eps =
#  = b0 + b1*(e11*(X1-m1)+e21*(X2-m2)) + b2*(e12*(X1-m1)+e22*(X2-m2)) + eps =
#  = b0 - b1*e11*m1 - b2*e12*m1 - b1*e21*m2 - b2*e22*m2 +
#                   + (b1*e11+b2*e12)*X1 + (b1*e21+b2*e22)*X2 + eps
# where e.ij are the loadings, i=1,2, j=1,2.
# => We can compute the coefficients of the model which used the original
#    regressors
m1 <- mean(speed1)
m2 <- mean(speed2)
beta0 <- coefficients(fm.pc)[1] -
  coefficients(fm.pc)[2]*speed.pc$load[1,1]*m1 -
  coefficients(fm.pc)[3]*speed.pc$load[1,2]*m1 -
  coefficients(fm.pc)[2]*speed.pc$load[2,1]*m2 -
  coefficients(fm.pc)[3]*speed.pc$load[2,2]*m2
beta1 <- coefficients(fm.pc)[2]*speed.pc$load[1,1] +
  coefficients(fm.pc)[3]*speed.pc$load[1,2]
beta2 <- coefficients(fm.pc)[2]*speed.pc$load[2,1] +
  coefficients(fm.pc)[3]*speed.pc$load[2,2]

c(beta0=as.numeric(beta0),beta1=as.numeric(beta1),beta2=as.numeric(beta2))
```

when we use PC_j regressors we can go back to the original variables (in the original space) using the expressions of PC_j (since they're linear combinations of the original variables)

```
##      beta0      beta1      beta2
## 2.4701378 0.9132876 0.0999593
```

Using both PC_1 and PC_2 we obtain basically the same coefficients as with lm

(same original coefficients, not the coefficients associated with PCA)

=

```
fm$coefficients
```

```
## (Intercept)      speed1      speed2
##   2.4701378   0.9132876   0.0999593
```

```
x <- seq(0, 25, len=100)
plot(cars, xlab='Speed', ylab='Stopping distance', las=1)
lines(x, beta0+beta1*x+beta2*x^2)
```

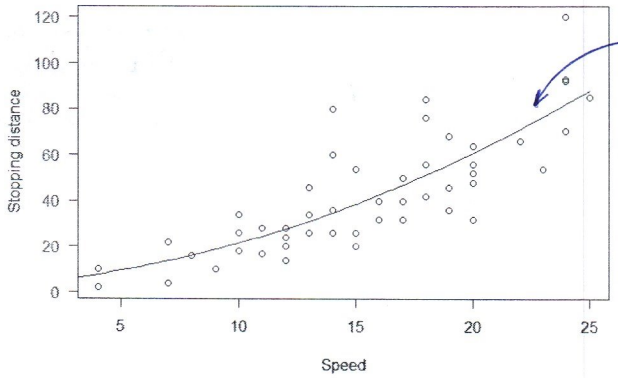
$$Y = \beta_0 + \beta_1 speed + \beta_2 speed^2 + \varepsilon$$

$$Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC + \varepsilon$$

$$\Rightarrow Y = \hat{\beta_0} + \hat{\beta_1} speed + \hat{\beta_2} speed^2 + \varepsilon$$

(what are shown are the $\hat{\beta_j}$)

The β_j do not differ from $\hat{\beta_j}$ because we've using all the principal components (so we've using again all the features in the same way)



regression line with PC_1 and PC_2 (showed in the original space)

```
# Reduce the model:
fm.pc <- lm(distance ~ sp1.pc)
summary(fm.pc)
```

(using the PCA: look how many variables we really need)

we saw that is enough to use just the first PC

```
##
## Call:
## lm(formula = distance ~ sp1.pc)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.449  -9.205  -3.595   5.074  45.859
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.98000    2.12825  20.195  < 2e-16 ***
## sp1.pc       0.12890    0.01318   9.781  5.2e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.05 on 48 degrees of freedom
## Multiple R-squared:  0.6659, Adjusted R-squared:  0.659
## F-statistic: 95.68 on 1 and 48 DF,  p-value: 5.195e-13
```

```
# We can re-write the model as:
# Model: y = b0 + b1*      PC1                  + eps =
#          = b0 + b1*(e11*(X-m1)1+e21*(X2-m2)) + eps =
#          = b0 - b1*e11*m1 - b2*e21*m2 + b1*e11*X1 + b1*e21*X2 + eps
beta0 <- coefficients(fm.pc)[1] -
  coefficients(fm.pc)[2]*speed.pc$load[1,1]*m1 -
  coefficients(fm.pc)[2]*speed.pc$load[2,1]*m2
beta1 <- coefficients(fm.pc)[2]*speed.pc$load[1,1]
beta2 <- coefficients(fm.pc)[2]*speed.pc$load[2,1]

c(beta0=as.numeric(beta0),beta1=as.numeric(beta1),beta2=as.numeric(beta2))
```

we obtain $\hat{\beta}_1$ and $\hat{\beta}_2$ because PC_1 is the linear combination of the two

```
##       beta0       beta1       beta2
## 8.831088116 0.004092603 0.128839907
```

Here we're considering and so we obtain: $Y = \beta_0 + \beta_1 PC_1 + \varepsilon$

$Y = \hat{\beta_0} + \hat{\beta_1}\,speed + \hat{\beta_2}\,speed^2 + \varepsilon$

≠

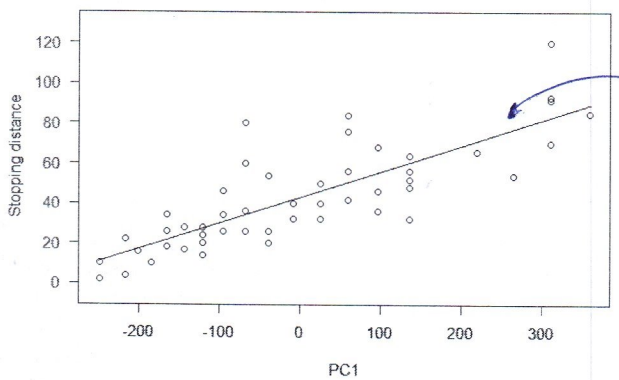
```
fm$coefficients
```

```
## (Intercept)      speed1      speed2
##   2.4701378   0.9132876   0.0999593
```

Here we're considering $Y = \beta_0 + \beta_1\,speed + \beta_2\,speed^2 + \varepsilon$

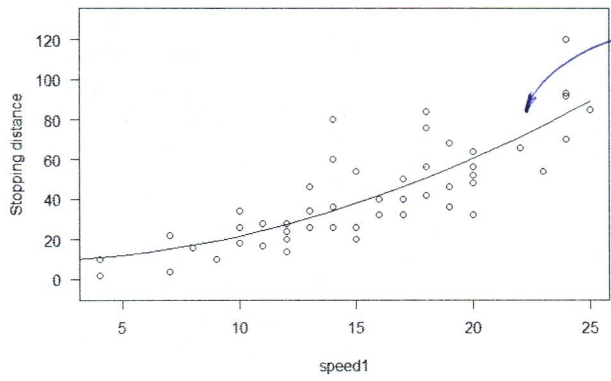
```
plot(sp1.pc,distance, xlab='PC1', ylab='Stopping distance', las=1, xlim=c(-250,361), ylim=c(-5,130))
x <- seq(-250,361,by=1)
b <- coef(fm.pc)
lines(x, b[1]+b[2]*x)
```

β_j and $\hat{\beta_j}$ this time are different because, to obtain $\hat{\beta_j}$ we use only PC_1



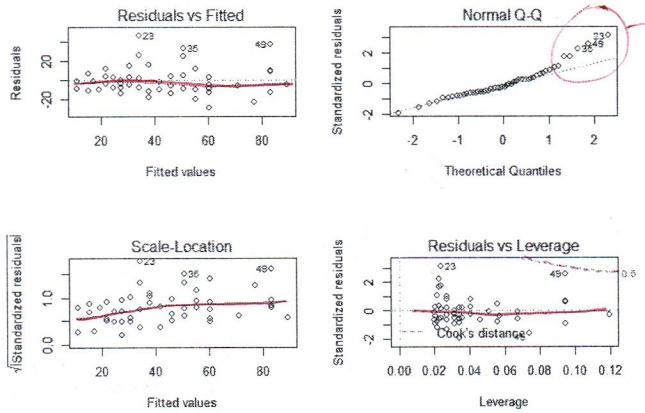
regression line with PC_1 (only) (showed in the space of PC_1)

```
plot(speed1,distance, ylab='Stopping distance', las=1, ylim=c(-5,130))
x <- seq(0,25,by=1)
lines(x, beta0 + beta1*x + beta2*x^2)
```



regression line with PC_1 (only!) (showed in the space of original variables)

```
# diagnostics of the residuals
par(mfrow=c(2,2))
plot(fm.pc)
```



gaussianity is not completely satisfied

```
shapiro.test(residuals(fm.pc))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fm.pc)
## W = 0.93034, p-value = 0.005694
```

```
dev.off()
```

we can do a principal component regression or just see what PCA suggests to use as regressors (even if this second method is not recommended by someone, anyway it's a good thing if we have some prior knowledge)

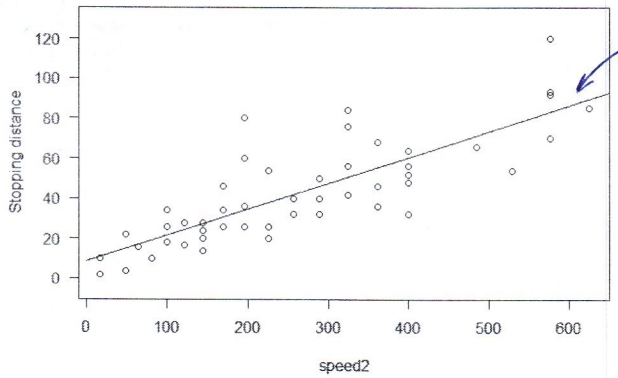
```
# The first PC is basically speed2 (centered with respect to the mean),
# hence we could just consider the regressor speed2

fm.2 <- lm(distance ~ speed2)
summary(fm.2)
```

```
##
## Call:
## lm(formula = distance ~ speed2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.448  -9.211  -3.594   5.076  45.862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.86005    4.08633   2.168   0.0351 *
## speed2       0.12897    0.01319   9.781  5.2e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.05 on 48 degrees of freedom
## Multiple R-squared:  0.6659, Adjusted R-squared:  0.6589
## F-statistic: 95.67 on 1 and 48 DF,  p-value: 5.2e-13
```

because of:

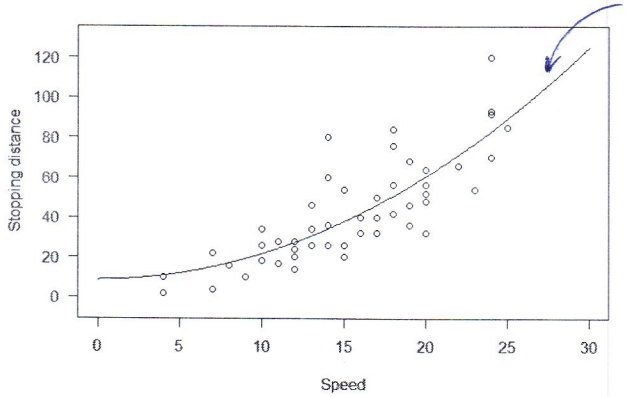
```
# Note: when using as regressors polynomials of the same variable (i.e., in
# the framework of polynomial regression), some authors recommend instead to
# keep all the orders lower than the maximum that one wants to keep (e.g., if
# keeping the maximum order 3, they recommend keeping also all the terms
# of order 2 and 1, for a "hierarchy" principle)

x11()
plot(speed2, distance, xlab='speed2', ylab='Stopping distance', las=1, xlim=c(15,626), ylim=c(-5,130))
x <- seq(0,650,by=1)
b <- coef(fm.2)
lines(x, b[1]+b[2]*x)
```

we have X, X^2, X^3, X^4 and we want to take X^3? some suggest to take X, X^2, X^3 (but if we have prior knowledge and for example we know a law that is quadratic, then we don't take the linear order too



model with just speed^2

```
plot(cars, xlab='Speed', ylab='Stopping distance', las=1, xlim=c(0,30), ylim=c(-5,130))
x <- seq(0,30,by=0.1)
b <- coef(fm.2)
lines(x, b[1]+b[2]*x^2)
```



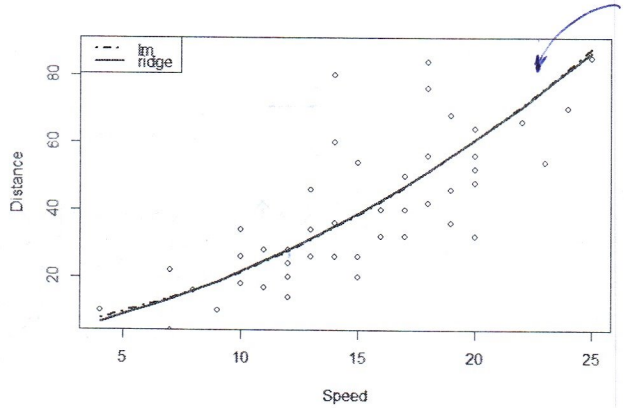
model with speed^2 but showed in the original space

```
dev.off()
```

```
### Otherwise, another possible solution to collinearity: ridge regression
help(lm.ridge)

# Fix lambda
lambda <- .5
fit.ridge <- lm.ridge(distance ~ speed1 + speed2, lambda = lambda)
# Note: to fit the model, R automatically centers X and Y
# with respect to their mean.

coef.ridge <- coef(fit.ridge)
yhat.lm <- cbind(rep(1,n), speed1, speed2)%*%coef(fm)  # LM fitted values
yhat.r <- cbind(rep(1,n), speed1, speed2)%*%coef.ridge # ridge fitted values

x11()
plot(speed1, yhat.lm, type='l', lty=4, lwd=2, ylab='Distance',xlab='Speed')
points(speed1, distance, pch=1, cex=.8)
matlines(speed1, yhat.r, type='l', lty=1,col=grey.colors(length(lambda)), lwd=2)
legend("topleft",c("lm","ridge"),lty=c(4,1),col=c("black",grey.colors(length(lambda))),lwd=2)
```

penalization of the L^2 norm of the vector of the coefficients β

comparison with linear model

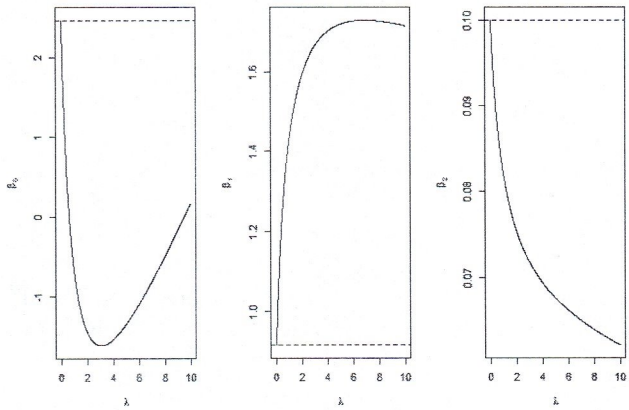


the two models are very similar

```
# Repeat for a grid of lambda's
lambda.c <- seq(0,10,0.01)
fit.ridge <- lm.ridge(distance ~ speed1 + speed2, lambda = lambda.c)

x11(width=14, height=5)
par(mfrow=c(1,3))
plot(lambda.c,coef(fit.ridge)[,1], type='l', xlab=expression(lambda),
     ylab=expression(beta[0]))
abline(h=coef(fm)[1], lty=2)
plot(lambda.c,coef(fit.ridge)[,2], type='l', xlab=expression(lambda),
     ylab=expression(beta[1]))
abline(h=coef(fm)[2], lty=2)
plot(lambda.c,coef(fit.ridge)[,3], type='l', xlab=expression(lambda),
     ylab=expression(beta[2]))
abline(h=coef(fm)[3], lty=2)
```

← same function, we just have to provide a vector of λ instead of a single value of λ

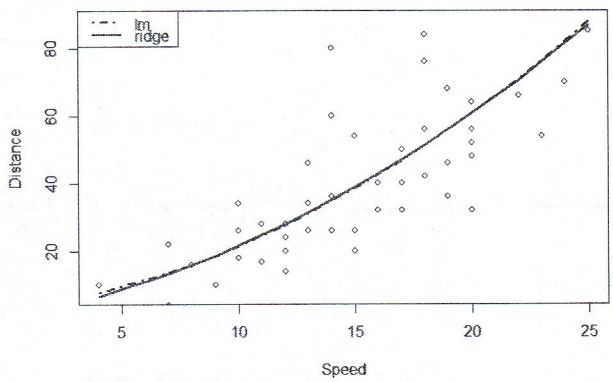
Graphical representation of β_0, β_1, β_2 as λ changes:




```
dev.off()
```

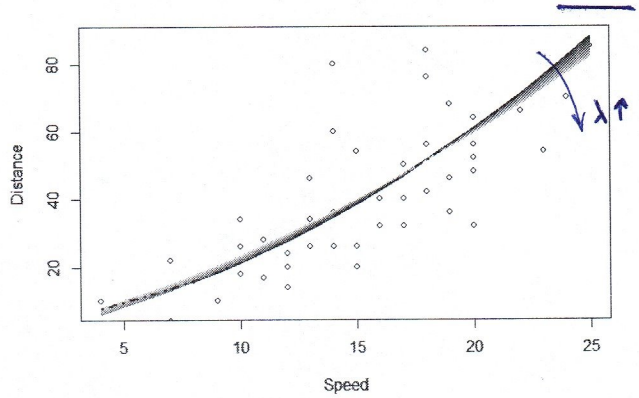
We plot all the possible models (linear model and all the models with those λ's)

```
yhat.lm <- cbind(rep(1,n), speed1, speed2)%*%coef(fm)
```




```
plot(speed1, yhat.lm, type='l', lty=1, lwd=2, ylab='Distance',
     xlab='Speed')
points(speed1, distance, pch=1, cex=.8)
yhat.r <- NULL
for(i in 1:length(lambda.c))
  yhat.r=cbind(yhat.r, cbind(rep(1,n), speed1, speed2)%*%coef(fit.ridge)[i,])
matlines(speed1, yhat.r, type='l', lty=1,
         col=grey.colors(length(lambda.c)))
lines(speed1, yhat.lm, type='l', lty=4, lwd=2, ylab='Distance',
      xlab='Speed')
```

---- linear model
ridge models (they're 1000)



: increasing the λ we increase the penalization of L^2 norm of the β's, so increasing λ we decrease the length of the vector of β's ⇒ FLATTERING OF THE CURVE

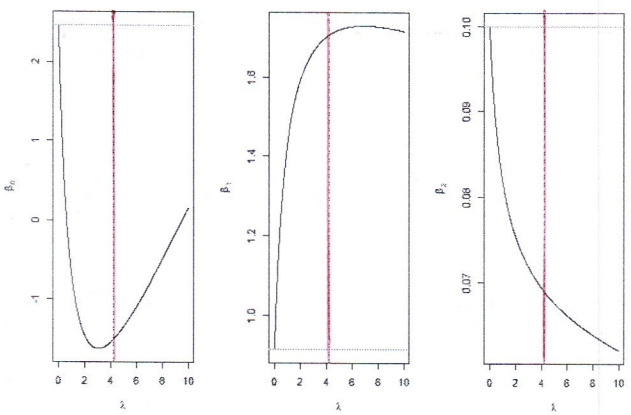
```
# Choice of the optimal lambda, e.g., via cross-validation
select(fit.ridge)
```

```
## modified HKB estimator is 0
## modified L-W estimator is 0
## smallest value of GCV  at 4.26
```

```
# or
lambda.opt <- lambda.c[which.min(fit.ridge$GCV)]
lambda.opt
```

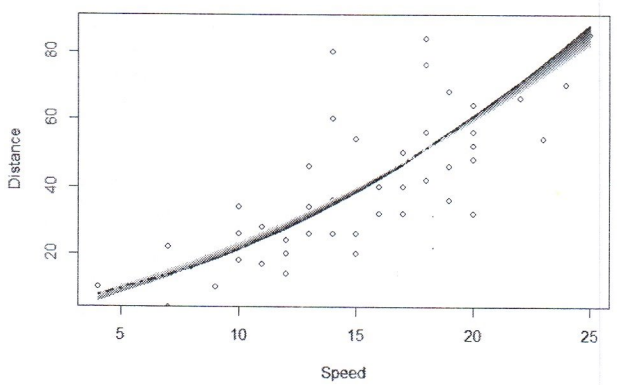
```
## [1] 4.26
```

```
x11(width=14, height=5)
par(mfrow=c(1,3))
plot(lambda.c,coef(fit.ridge)[,1], type='l', xlab=expression(lambda),
     ylab=expression(beta[0]))
abline(h=coef(fm)[1], lty=1, col='grey')
abline(v=lambda.opt, col=2, lty=2)
plot(lambda.c,coef(fit.ridge)[,2], type='l', xlab=expression(lambda),
     ylab=expression(beta[1]))
abline(h=coef(fm)[2], lty=1, col='grey')
abline(v=lambda.opt, col=2, lty=2)
plot(lambda.c,coef(fit.ridge)[,3], type='l', xlab=expression(lambda),
     ylab=expression(beta[2]))
abline(h=coef(fm)[3], lty=1, col='grey')
abline(v=lambda.opt, col=2, lty=2)
```

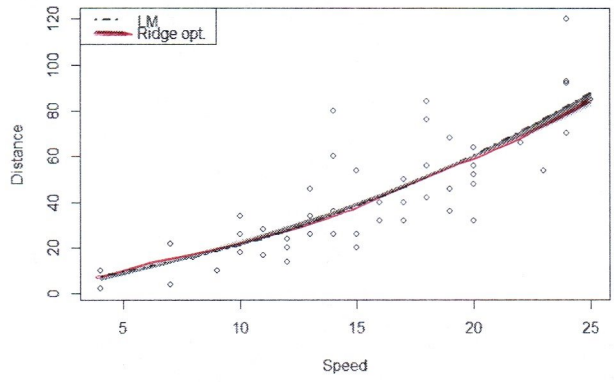



```
dev.off()
```

```
plot(speed1, distance, pch=1, cex=.8, ylab='Distance',
     xlab='Speed')
```




```
matlines(speed1, yhat.r, type='l', lty=1,
         col=grey.colors(length(lambda.c)))
lines(speed1, yhat.lm, type='l', lty=4, lwd=2, ylab='Distance',
      xlab='Speed')
lines(speed1, yhat.r[,which.min(fit.ridge$GCV)], type='l', lty=1, lwd=2,
      col=2, ylab='Distance', xlab='Speed')
legend("topleft", c('LM', 'Ridge opt.' ), lty=c(4,1), col=c(1,2), lwd=2)
```




```
coef.ridge <- coef(fit.ridge)[which.min(fit.ridge$GCV),]
coef.ridge
```

← we can extract the optimal model's coefficients

```
##                speed1      speed2
## -1.51415846  1.70485617  0.06894229
```

```
### Otherwise, another possible solution to collinearity: Lasso regression
help(glmnet)

# Build the matrix of predictors
x <- model.matrix(distance~speed1+speed2)[,-1]
# Build the vector of response
y <- distance

# Let's set a grid of candidate lambda's for the estimate
lambda.grid <- 10^seq(5,-3,length=100)
fit.lasso <- glmnet(x,y, lambda = lambda.grid) # default: alpha=1 -> lasso
# [note: if alpha=0 -> ridge regression]

plot(fit.lasso,xvar='lambda',label=TRUE, col = rainbow(dim(x)[2]))
legend('topright', dimnames(x)[[2]], col =  rainbow(dim(x)[2]), lty=1, cex=1)
```

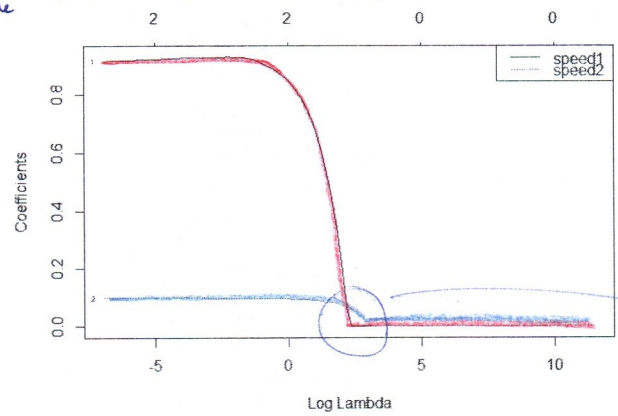
← penalization of the L^1 norm of the coefficients β (of the vector of coefficients β)

← the penalization of this model is a weighted sum of the penalization of the lasso regression and the penalization of Ridge regression

$\alpha = 1 \Rightarrow$ lasso
$\alpha = 0 \Rightarrow$ Ridge

The default option for the graphical representation of glmnet is the behaviour of the coefficients $\beta_0, \beta_1, \beta_2$ as a function of $\log(\lambda)$

(In this case we see only β_1 and β_2)



Ridge: $\alpha = 0$; Lasso: $\alpha = 1$

everything in between: $0 < \alpha < 1$

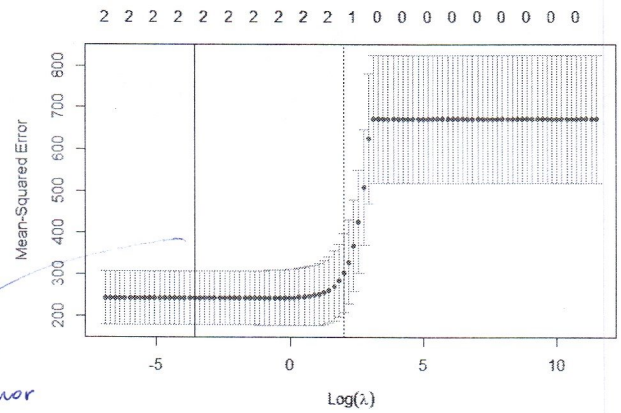
→ in Ridge we have a smooth behaviour, in lasso regression we have a stronger penalization
⇒ the behaviour of β's is not smooth
(at some point the coefficients go to zero, there is no delicate curve)

```
# Let's set lambda via cross validation
cv.lasso <- cv.glmnet(x,y,lambda=lambda.grid) # default: 10-fold CV

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

```
## [1] 0.02848036
```

```
plot(cv.lasso)
abline(v=log(bestlam.lasso), lty=1)
```

plot(< cv.glmnet >) gives the values of the mean square errors as a function of $\log(\lambda)$



this line get the $\log(\hat{\lambda})$ such that the mean square error is minimum

```
# Get the coefficients for the optimal lambda
coef.lasso <- predict(fit.lasso, s=bestlam.lasso, type = 'coefficients')[1:3,]
coef.lasso
```

```
## (Intercept)      speed1      speed2
##  2.39559667  0.93061333  0.09923253
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Multiple linear regression
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------

data <- read.table('concrete.txt', header=T)

head(data)
```

X_1 X_2 X_3 X_4 Y

```
##   Alluminium Silicate Alluminium_ferrite Silicate_bicalcium Hardness_concrete
## 1          7       16                  6                 60              78.5
## 2          1       19                 15                 52              74.3
## 3         11       56                  8                 20             104.3
## 4         11       31                  8                 47              87.6
## 5          7       52                  6                 33              95.9
## 6         11       55                  9                 22             109.2
```

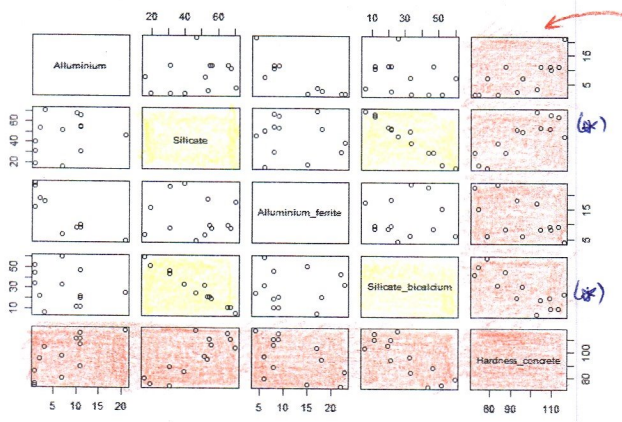
```
dim(data)
```

```
## [1] 13  5
```

```
names(data)
```

```
## [1] "Alluminium"         "Silicate"           "Alluminium_ferrite"
## [4] "Silicate_bicalcium" "Hardness_concrete"
```

```
pairs(data)
```

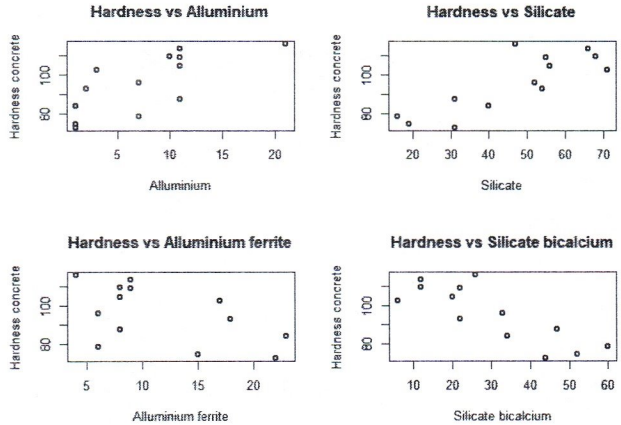


Scatterplots where Y is one of the two variables, we see that there are two of them (*) which have a good correlation

Most important: we look at the other scatterplots to have an idea of the correlation between covariates and in particular we spot a problem of colinearity ( )

```
attach(data)

X1 <- Alluminium
X2 <- Silicate
X3 <- Alluminium_ferrite
X4 <- Silicate_bicalcium
Y <- Hardness_concrete

detach(data)

par(mfrow=c(2,2))
plot(X1, Y, main='Hardness vs Alluminium', lwd=2,
     xlab='Alluminium', ylab='Hardness concrete')
plot(X2, Y, main='Hardness vs Silicate', lwd=2,
     xlab='Silicate', ylab='Hardness concrete')
plot(X3, Y, main='Hardness vs Alluminium ferrite', lwd=2,
     xlab='Alluminium ferrite', ylab='Hardness concrete')
plot(X4, Y, main='Hardness vs Silicate bicalcium', lwd=2,
     xlab='Silicate bicalcium', ylab='Hardness concrete')
```

(Plot of the , but closer :)




```
### ---------------------------------------------------------------------------
### Multiple linear regression
### ---------------------------------------------------------------------------
result <- lm(Y ~ X1 + X2 + X3 + X4)
summary(result)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0990 -1.7178  0.2919  1.2055  3.7526
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 124.4809    26.7557   4.653  0.00164 **
## X1            0.9739     0.2835   3.435  0.00889 **
## X2           -0.1405     0.2891  -0.486  0.63996
## X3           -0.4974     0.2751  -1.808  0.10820
## X4           -0.7974     0.3214  -2.481  0.03805 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.484 on 8 degrees of freedom
## Multiple R-squared:  0.9818, Adjusted R-squared:  0.9727
## F-statistic:   108 on 4 and 8 DF,  p-value: 5.385e-07
```

✓ (symmetric)

✓ ✓ ✗ ✗ ✓

← because of what we saw in the scatterplot it may be a problem of collinearity ; let's check it with **VIF**

✓ ✓ ($\exists \beta_j \neq 0$)

```
vif(result)
```

```
##       X1       X2       X3       X4
##  5.408208 52.818202  6.035697 56.260964
```

← very large for X_2 and X_4 ($>> 10$)

```
### ---------------------------------------------------------------------------
### PCA regression
### ---------------------------------------------------------------------------
result.pc <- princomp(cbind(X1,X2,X3,X4), scores=TRUE)
summary(result.pc)
```

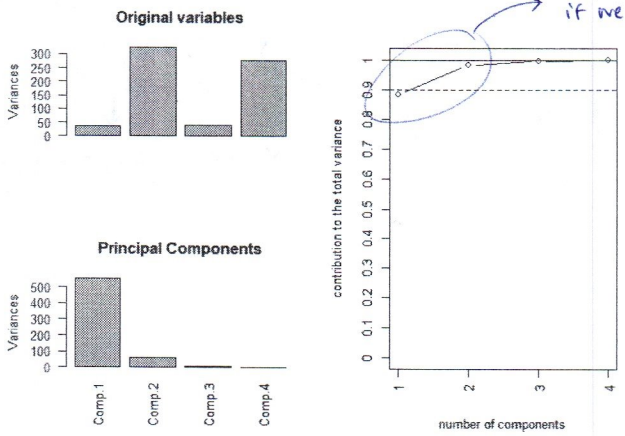
} principal component analysis

```
## Importance of components:
##                           Comp.1     Comp.2     Comp.3      Comp.4
## Standard deviation     23.5887588 7.89074012 2.84498593 1.312101401
## Proportion of Variance  0.8853169 0.09906587 0.01287801 0.002739198
## Cumulative Proportion   0.8853169 0.98438279 0.99726080 1.000000000
```

```
result.pc$load
```

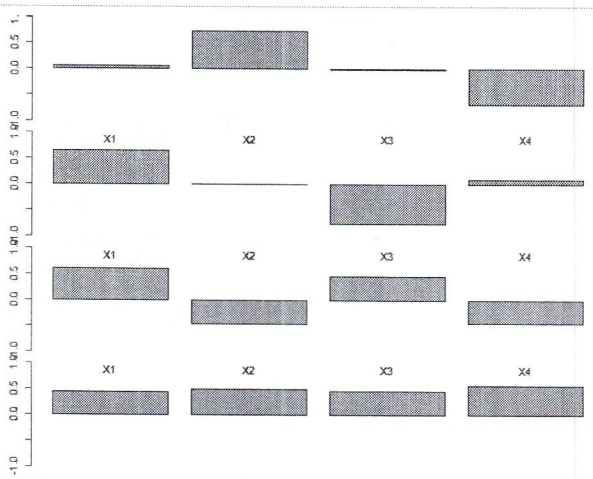
```
##
## Loadings:
##    Comp.1 Comp.2 Comp.3 Comp.4
## X1         0.649  0.611  0.448
## X2  0.732        -0.455  0.507
## X3        -0.755  0.467  0.461
## X4 -0.678        -0.449  0.574
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

```
# Explained variance
layout(matrix(c(2,3,1,3),2,byrow=T))
barplot(result.pc$sdev^2, las=2, main='Principal Components', ylab='Variances')
barplot(c(sd(X1),sd(X2),sd(X3),sd(X4))^2, las=2, main='Original variables', ylab='Variances')
plot(cumsum(result.pc$sdev^2)/sum(result.pc$sde^2), type='b', axes=F, xlab='number of components', ylab='contribution to the
total variance', ylim=c(0,1))
abline(h=1, col='blue')
abline(h=0.9, lty=2, col='blue')
box()
axis(2,at=0:10/10,labels=0:10/10)
axis(1,at=1:4,labels=1:4,las=2)
```



if we consider just PC_1 we have almost **0.9**,
if we consider PC_1 and PC_2 we have almost **1**
→ let's take PC_1 and PC_2

```
# Loadings
par(mar = c(1,4,0,2), mfrow = c(4,1))
for(i in 1:4)barplot(result.pc$load[,i], ylim = c(-1, 1))
```




```
dev.off()
```

```
# Dimensionality reduction: select first two PCs:
pc1 <- result.pc$scores[,1]
pc2 <- result.pc$scores[,2]

# Now we estimate the model using the first two PCs as regressors.
# Model: y = b0 + b1*PC1+ b2*PC2 + eps, eps~N(0,sigma^2)
fm.pc <- lm(Y ~ pc1 + pc2)

summary(fm.pc)
```

$$Y = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \varepsilon$$

```
##
## Call:
## lm(formula = Y ~ pc1 + pc2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.623 -2.245 -1.312  1.587  5.094
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 95.42308    0.94690 100.775 2.27e-16 ***
## pc1          0.51195    0.04014  12.754 1.64e-07 ***
## pc2          0.93214    0.12000   7.768 1.52e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.414 on 10 degrees of freedom
## Multiple R-squared:  0.9571, Adjusted R-squared:  0.9485
## F-statistic: 111.5 on 2 and 10 DF,  p-value: 1.456e-07
```

✓ ✓ ✓ ✓ ✓

```
# We can re-write the model as:
# Model:
# y= b0 + b1*PC1 + b2*PC2 + eps =
#  = b0 + b1*(e11*(X1-m1)+e21*(X2-m2)+e31*(X3-m3)+e41*(X4-m4)) +
#       + b2*(e12*(X1-m1)+e22*(X2-m2)+e32*(X3-m3)+e42*(X4-m4)) + eps =
#  = b0 - b1*e11*m1 - b2*e12*m1 - b1*e21*m2 - b2*e22*m2 +
#       - b1*e31*m3 - b2*e32*m3 - b1*e41*m4 - b2*e42*m4 +
#       + (b1*e11+b2*e12)*X1 + (b1*e21+b2*e22)*X2 +
#       + (b1*e31+b2*e32)*X3 + (b1*e41+b2*e42)*X4 + eps
# where e.ij are the loadings, i=1,2,3,4, j=1,2.
# => We can compute the coefficients of the model which used the original
#    regressors
m1 <- mean(X1)
m2 <- mean(X2)
m3 <- mean(X3)
m4 <- mean(X4)
beta0 <- coefficients(fm.pc)[1] -
  coefficients(fm.pc)[2]*result.pc$load[1,1]*m1 -
  coefficients(fm.pc)[3]*result.pc$load[1,2]*m1 -
  coefficients(fm.pc)[2]*result.pc$load[2,1]*m2 -
  coefficients(fm.pc)[3]*result.pc$load[2,2]*m2 -
  coefficients(fm.pc)[2]*result.pc$load[3,1]*m3 -
  coefficients(fm.pc)[3]*result.pc$load[3,2]*m3 -
  coefficients(fm.pc)[2]*result.pc$load[4,1]*m4 -
  coefficients(fm.pc)[3]*result.pc$load[4,2]*m4
beta1 <- coefficients(fm.pc)[2]*result.pc$load[1,1] +
  coefficients(fm.pc)[3]*result.pc$load[1,2]
beta2 <- coefficients(fm.pc)[2]*result.pc$load[2,1] +
  coefficients(fm.pc)[3]*result.pc$load[2,2]
beta3 <- coefficients(fm.pc)[2]*result.pc$load[3,1] +
  coefficients(fm.pc)[3]*result.pc$load[3,2]
beta4 <- coefficients(fm.pc)[2]*result.pc$load[4,1] +
  coefficients(fm.pc)[3]*result.pc$load[4,2]

c(beta0=as.numeric(beta0),beta1=as.numeric(beta1),beta2=as.numeric(beta2),beta3=as.numeric(beta3),beta4=as.numeric(beta4))
```

We can go back to the original system:

$$Y = \hat{\beta}_0 + \hat{\beta}_1 PC_1 + \hat{\beta}_2 PC_2 + \varepsilon$$

$$\downarrow$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \varepsilon$$

```
##      beta0      beta1      beta2      beta3      beta4
## 89.1310486  0.6383398  0.3804914 -0.7150276 -0.2597465
```

← from PC regression

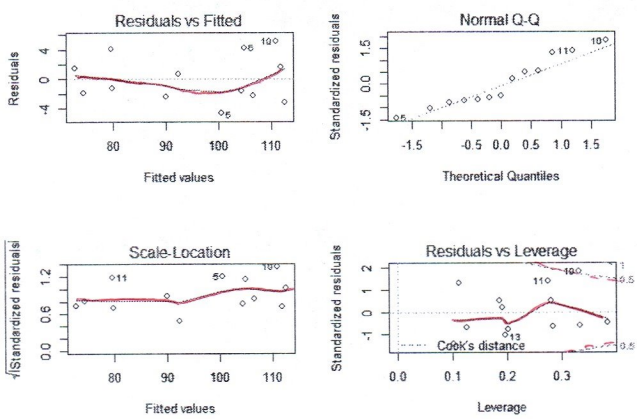
```
result$coefficients
```

≠

```
## (Intercept)          X1          X2          X3          X4
##  124.4808741   0.9739087  -0.1405100  -0.4973948  -0.7973591
```

← from linear regression

```
# diagnostics of the residuals
x11()
par(mfrow=c(2,2))
plot(fm.pc)
```



✓

```
shapiro.test(residuals(fm.pc))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(fm.pc)
## W = 0.92873, p-value = 0.328
```

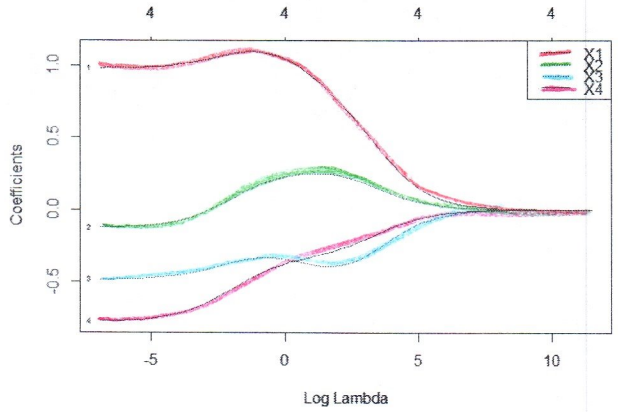
✓

```
dev.off()
```

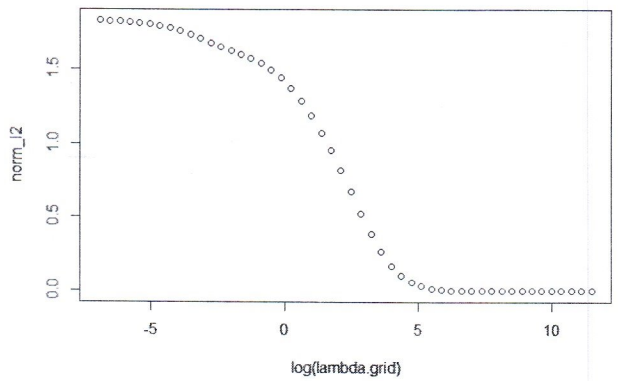
```
### ------------------------------------------------------------------------
### Ridge regression (with glmnet)
### ------------------------------------------------------------------------
x <- model.matrix(Y ~ X1 + X2 + X3 + X4)[,-1] # matrix of predictors
y <- Y # vector of response
lambda.grid <- 10^seq(5,-3,length=50)

# Ridge regression
fit.ridge <- glmnet(x,y, lambda = lambda.grid, alpha=0) # alpha=0 -> ridge

x11()
plot(fit.ridge,xvar='lambda',label=TRUE, col = rainbow(dim(x)[2]))
legend('topright', dimnames(x)[[2]], col =  rainbow(dim(x)[2]), lty=1, cex=1)
```

→ we'll choose a λ by cross validation through a grid

Graphical representation of the behaviour of the coefficients $\beta_1, \beta_2, \beta_3, \beta_4$ as a function of $\log(\lambda)$:




```
norm_l2 <- NULL
for(i in 1:50)
  norm_l2 <- c(norm_l2,sum((fit.ridge$beta[,i])^2))

x11()
plot(log(lambda.grid),norm_l2)
```



We can look at the behaviour of L^2 norm of this sequence of vectors of coefficients β's and observe what happens changing λ

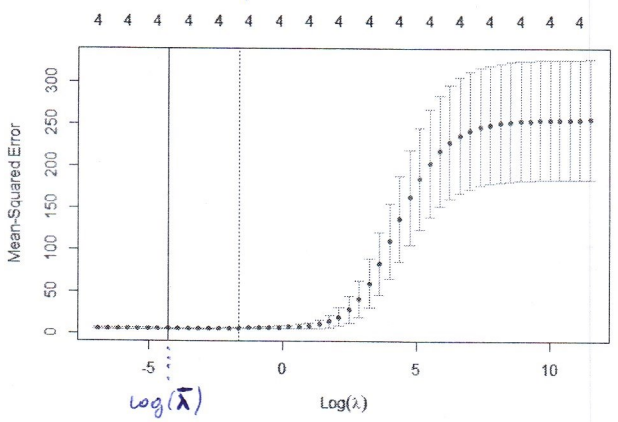
(With Ridge we penalize the L^2 norm of the vector of β's so by increasing λ we get a stronger penalization and this leads to an L^2 norm smaller and smaller ($\Rightarrow$ smaller and smaller β's))

```
# Let's set lambda via CV
set.seed(1)
cv.ridge <- cv.glmnet(x,y,alpha=0,nfolds=3,lambda=lambda.grid)

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

```
## [1] 0.01389495
```

```
x11()
plot(cv.ridge)
abline(v=log(bestlam.ridge), lty=1)
```

Behaviour of the mean square error as a function of $\log(\lambda)$

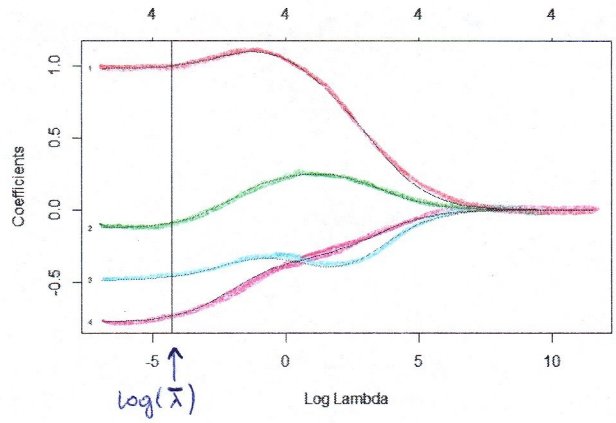


$\bar{\lambda}$ = Lambda that minimizes the mean square error (best λ for the model)

```
# Get the coefficients for the optimal lambda
coef.ridge <- predict(fit.ridge, s=bestlam.ridge, type = 'coefficients')[1:5,]
coef.ridge
```

```
##  (Intercept)           X1           X2           X3           X4
## 119.64979497   1.00319395  -0.08774654  -0.46421077  -0.73861152
```

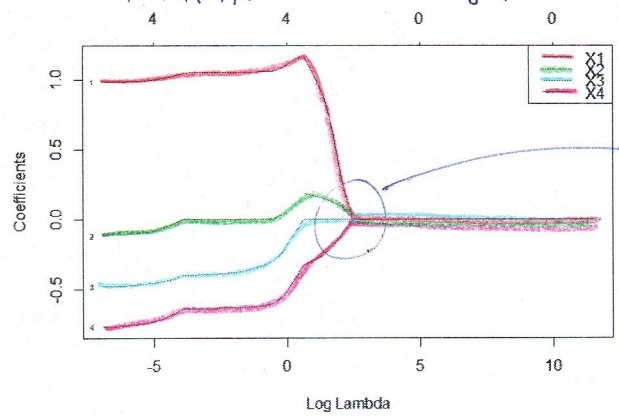
```
x11()
plot(fit.ridge,xvar='lambda',label=TRUE, col = rainbow(dim(x)[2]))
abline(v=log(bestlam.ridge))
```



$\log(\lambda)$

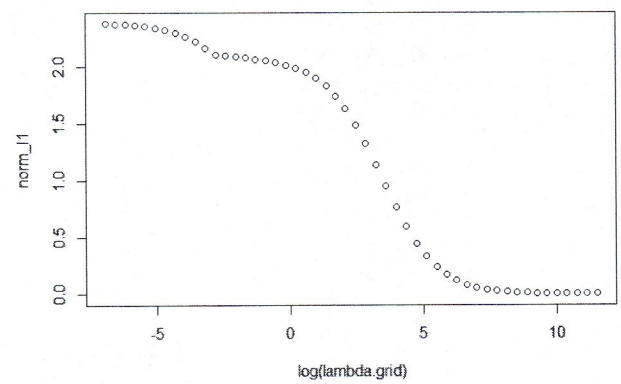
```
### ------------------------------------------------------------------------
### Lasso regression (with glmnet)
### ------------------------------------------------------------------------
fit.lasso <- glmnet(x,y, lambda = lambda.grid, alpha=1) # alpha=1 -> lasso

x11()
plot(fit.lasso,xvar='lambda',label=TRUE, col = rainbow(dim(x)[2]))
legend('topright', dimnames(x)[[2]], col =  rainbow(dim(x)[2]), lty=1, cex=1)
```

Behaviour of $\beta_1, \beta_2, \beta_3, \beta_4$ as function of $\log(\lambda)$:



again, not smooth

```
norm_l1 <- NULL
for(i in 1:50)
  norm_l1 <- c(norm_l1,sum(abs(fit.ridge$beta[,i])))

x11()
plot(log(lambda.grid),norm_l1)
```

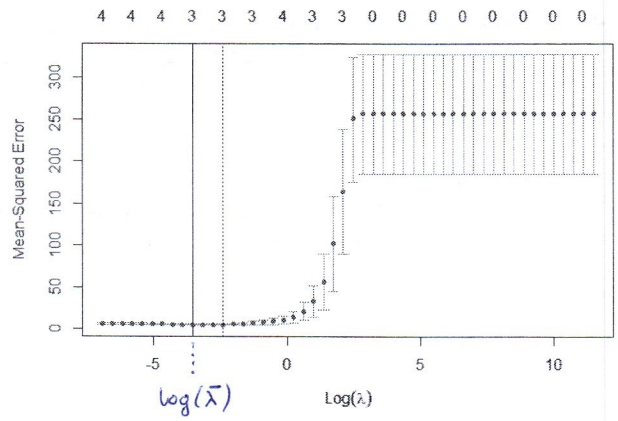


behaviour of L^1 norm of the sequence of vectors of β's as λ changes

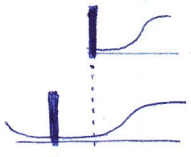
```
# Let's set lambda via CV
set.seed(1)
cv.lasso <- cv.glmnet(x,y,alpha=1,nfolds=3,lambda=lambda.grid)

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

```
## [1] 0.02947052
```

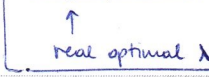
```
x11()
plot(cv.lasso)
abline(v=log(bestlam.lasso), lty=1)
```



NOTE: if $\bar{\lambda}$ is such that the black continuous line is at the extrem ⇒ change the grid of λ:



: maybe changing grid we find that:

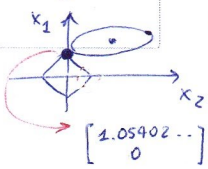


↑ real optimal λ

```
# Get the coefficients for the optimal Lambda
coef.lasso <- predict(fit.lasso, s=bestlam.lasso, type = 'coefficients')[1:5,]
coef.lasso
```

```
## (Intercept)          X1          X2          X3          X4
## 111.5358126   1.0540267   0.0000000  -0.4037632  -0.6408472
```

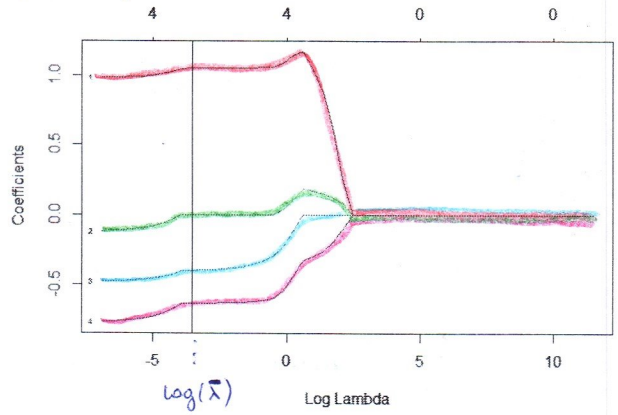
in this case lasso performed a variable selection too; probably happened something like:



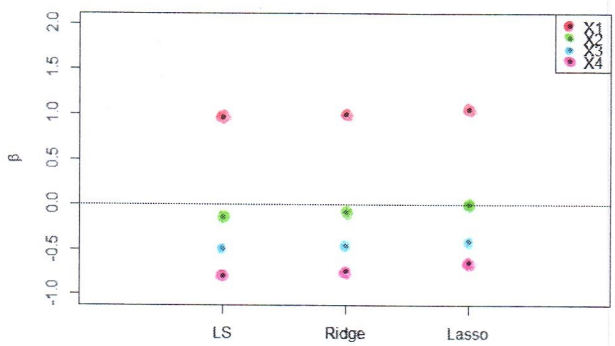
(but in higher dimension)

```
x11()
plot(fit.lasso,xvar='lambda',label=TRUE, col = rainbow(dim(x)[2]))
abline(v=log(bestlam.lasso))
```

Graphical repr. of $\bar{\lambda}$




```
### ------------------------------------------------------------------------
# Compare coefficients estimates for LS, Ridge and Lasso
x11()
plot(rep(0, dim(x)[2]), coef(lm(y~x))[-1], col=rainbow(dim(x)[2]), pch=20, xlim=c(-1,3), ylim=c(-1,2), xlab='', ylab=express
ion(beta),
     axes=F)
points(rep(1, dim(x)[2]), coef.ridge[-1], col=rainbow(dim(x)[2]), pch=20)
points(rep(2, dim(x)[2]), coef.lasso[-1], col=rainbow(dim(x)[2]), pch=20)
abline(h=0, col='grey41', lty=1)
box()
axis(2)
axis(1, at=c(0,1,2), labels = c('LS', 'Ridge', 'Lasso'))
legend('topright', dimnames(x)[[2]], col =  rainbow(dim(x)[2]), pch=20, cex=1)
```




```
# L2 norm
sum((coef(lm(y~x))[-1])^2) # LS
```

```
## [1] 1.851424
```

```
sum((coef.ridge[-1])^2)    # ridge
```

```
## [1] 1.775136
```

← with Ridge we obtain a vector of β's which L^2 norm is smaller than the LS' one

```
# L1 norm
sum(abs(coef(lm(y~x))[-1])) # LS
```

```
## [1] 2.409173
```

```
sum(abs(coef.lasso[-1]))    # lasso
```

```
## [1] 2.098637
```

← with lasso we obtain a vector of β's which L^1 norm is smaller than the LS' s one

```
### ------------------------------------------------------------
### Variable selection
result <- lm(Y ~ X1 + X2 + X3 + X4)
summary(result)
```

(from the LS model)

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3 + X4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0990 -1.7178  0.2919  1.2055  3.7526
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 124.4809    26.7557   4.653  0.00164 **
## X1            0.9739     0.2835   3.435  0.00889 **
## X2           -0.1405     0.2891  -0.486  0.63996
## X3           -0.4974     0.2751  -1.808  0.10820
## X4           -0.7974     0.3214  -2.481  0.03805 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.484 on 8 degrees of freedom
## Multiple R-squared:  0.9818, Adjusted R-squared:  0.9727
## F-statistic:   108 on 4 and 8 DF,  p-value: 5.385e-07
```

✓ ✓ ✗ → worse, we remove this as first ✗ ✓

```
result1 <- lm(Y ~ X1 + X3 + X4)
summary(result1)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X3 + X4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9323 -1.8090  0.4806  1.1398  3.7771
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 111.68441    4.56248  24.479 1.52e-09 ***
## X1            1.05185    0.22368   4.702  0.00112 **
## X3           -0.41004    0.19923  -2.058  0.06969 .
## X4           -0.64280    0.04454 -14.431 1.58e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.377 on 9 degrees of freedom
## Multiple R-squared:  0.9813, Adjusted R-squared:  0.975
## F-statistic: 157.3 on 3 and 9 DF,  p-value: 4.312e-08
```

✓ ✓ ✗ → ✓

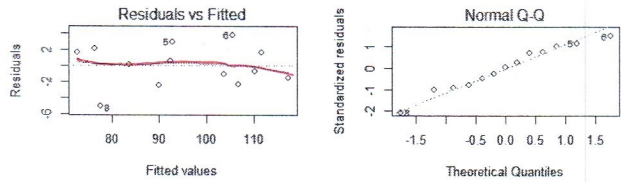
we remove this,
NOTICE that the p-value changed,
we have to remove ONE AT THE TIME
(Maybe 2 variables are not necessary together but at least one of them is strongly necessary (because both of them give a lot of informations, but the same information!))

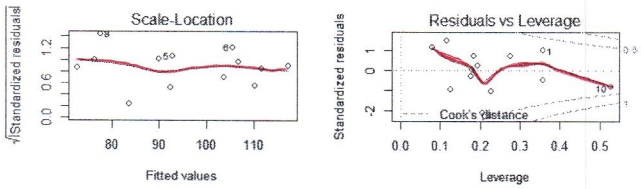
```
result2 <- lm(Y ~ X1 + X4)
summary(result2)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0234 -1.4737  0.1371  1.7305  3.7701
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 103.09738    2.12398   48.54 3.32e-13 ***
## X1            1.43996    0.13842   10.40 1.11e-06 ***
## X4           -0.61395    0.04864  -12.62 1.81e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.734 on 10 degrees of freedom
## Multiple R-squared:  0.9725, Adjusted R-squared:  0.967
## F-statistic: 176.6 on 2 and 10 DF,  p-value: 1.581e-08
```

✓ ✓ ✓

```
# diagnostics of the residuals
x11()
par(mfrow=c(2,2))
plot(result2)
```






```
shapiro.test(residuals(result2))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(result2)
## W = 0.9751, p-value = 0.9468
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Hitters dataset
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------

library(ISLR)

help(Hitters)
names(Hitters)
```

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

```
dim(Hitters)
```

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

```
# remove NA's
sum(is.na(Hitters$Salary))
```

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

```
Hitters <- na.omit(Hitters)
dim(Hitters)
```

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

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

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

```
### ------------------------------------------------------------------------
###  Subset Selection Methods
### ------------------------------------------------------------------------

library(leaps)
help(regsubsets)

# Best Subset Selection (exhaustive search)
regfit.full <- regsubsets(Salary~., data=Hitters)
summary(regfit.full)
```

: if we have many regressors we can't procede with manual removal / addition of features, we need something "automatized"

we check ALL the possible models and for each number of variables admittable ($k = 1, \dots, 19$ in this case) we choose the best variables

Here we're checking 2^{19} models

That's why here we don't have nested models: maybe with $k=1$ we consider X_1 and for $k=2$ we consider X_2, X_3 as best variables. This is different from backward selection / forward selection because here we've exploring ALL POSSIBLE MODELS, with forward/backward selection we're exploring just one path!

(In forward selection once we select something we don't get rid of it anymore)

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 8
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    " "   " "    " "   "*"
## 7  ( 1 ) " "   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   " "
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
```

response: salary
possible covariates: all the other

one line for each degree of complexity →

⇒ if we can take just one regressor then we take "CRBI"

⇒ if we can take two regressors then we take "CRBI" and "Hits"

Note: the models are not nested
If we're allowed to take 7 regressors we don't take "CRBI"

```
regfit.full <- regsubsets(Salary~., data=Hitters, nvmax=19)
summary(regfit.full)
```

← we go on until 19 variables

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 7  ( 1 ) " "   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   " "
## 9  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   "*"
## 10 ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   "*"
## 11 ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    "*"   "*"    " "   "*"
## 12 ( 1 ) "*"   "*"  " "   " "  "*" " "   "*"   " "    "*"   "*"    " "   "*"
## 13 ( 1 ) "*"   "*"  " "   "*"  "*" " "   "*"   " "    "*"   "*"    " "   "*"
## 14 ( 1 ) "*"   "*"  "*"   "*"  " " " "   "*"   " "    "*"   "*"    " "   "*"
## 15 ( 1 ) "*"   "*"  "*"   "*"  " " " "   "*"   " "    "*"   "*"    "*"   "*"
## 16 ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   " "    "*"   "*"    "*"   "*"
## 17 ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
## 18 ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
## 19 ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 10 ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11 ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12 ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19 ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

```
reg.summary <- summary(regfit.full)
names(reg.summary)
```

```
## [1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
```

```
reg.summary$which
```

```
##    (Intercept) AtBat  Hits HmRun  Runs   RBI Walks Years CAtBat CHits CHmRun
## 1         TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  FALSE
## 2         TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  FALSE
## 3         TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  FALSE
## 4         TRUE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  FALSE
## 5         TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  FALSE FALSE  FALSE
## 6         TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  FALSE FALSE  FALSE
## 7         TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE   TRUE  TRUE   TRUE
## 8         TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE  FALSE FALSE   TRUE
## 9         TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE   TRUE FALSE  FALSE
## 10        TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE   TRUE FALSE  FALSE
## 11        TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE   TRUE FALSE  FALSE
## 12        TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE   TRUE FALSE  FALSE
## 13        TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE   TRUE FALSE  FALSE
## 14        TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE   TRUE FALSE  FALSE
## 15        TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE   TRUE  TRUE  FALSE
## 16        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE   TRUE  TRUE  FALSE
## 17        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE   TRUE  TRUE  FALSE
## 18        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE   TRUE  TRUE  FALSE
## 19        TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE   TRUE  TRUE   TRUE
##    CRuns  CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  FALSE  TRUE  FALSE   FALSE     FALSE   FALSE   FALSE  FALSE      FALSE
## 2  FALSE  TRUE  FALSE   FALSE     FALSE   FALSE   FALSE  FALSE      FALSE
## 3  FALSE  TRUE  FALSE   FALSE     FALSE    TRUE   FALSE  FALSE      FALSE
## 4  FALSE  TRUE  FALSE   FALSE      TRUE    TRUE   FALSE  FALSE      FALSE
## 5  FALSE  TRUE  FALSE   FALSE      TRUE    TRUE   FALSE  FALSE      FALSE
## 6  FALSE  TRUE  FALSE   FALSE      TRUE    TRUE   FALSE  FALSE      FALSE
## 7  FALSE FALSE  FALSE   FALSE      TRUE    TRUE   FALSE  FALSE      FALSE
## 8   TRUE FALSE   TRUE   FALSE      TRUE    TRUE   FALSE  FALSE      FALSE
## 9   TRUE  TRUE   TRUE   FALSE      TRUE    TRUE   FALSE  FALSE      FALSE
## 10  TRUE  TRUE   TRUE   FALSE      TRUE    TRUE    TRUE  FALSE      FALSE
## 11  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE  FALSE      FALSE
## 12  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE  FALSE      FALSE
## 13  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE      FALSE
## 14  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE      FALSE
## 15  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE      FALSE
## 16  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE      FALSE
## 17  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE       TRUE
## 18  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE       TRUE
## 19  TRUE  TRUE   TRUE    TRUE      TRUE    TRUE    TRUE   TRUE       TRUE
```

```
reg.summary$rsq   # r-squared
```

R^2

```
##  [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227
##  [8] 0.5285569 0.5346124 0.5404950 0.5426153 0.5436302 0.5444570 0.5452164
## [15] 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```

```
reg.summary$adjr2 # adjusted r-squared
```

R^2_{adj}

```
##  [1] 0.3188503 0.4208024 0.4450753 0.4672734 0.4808971 0.4972001 0.5007849
##  [8] 0.5137083 0.5180572 0.5222606 0.5225706 0.5217245 0.5206736 0.5195431
## [15] 0.5178661 0.5162219 0.5144464 0.5126097 0.5106270
```

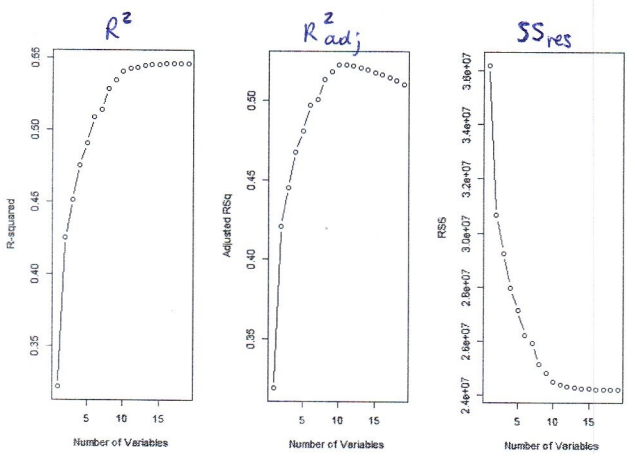
```
reg.summary$rss   # residual sum of squares
```

SS_{Res}

```
##  [1] 36179679 30646560 29249297 27970852 27149899 26194904 25906548 25136930
##  [9] 24814051 24500402 24387345 24333232 24289148 24248660 24235177 24219377
## [17] 24209447 24201837 24200700
```

```
x11(height=7,width=14)
par(mfrow=c(1,3))
plot(reg.summary$rsq,xlab="Number of Variables",ylab="R-squared",type="b")
plot(reg.summary$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="b")
plot(reg.summary$rss,xlab="Number of Variables",ylab="RSS",type="b")
```

R^2 R^2_{adj} SS_{res}




```
# extract coefficient estimates associated with the models
which.max(reg.summary$adjr2)
```

← we want the model with max R^2_{adj}

```
## [1] 11
```

← this model has 11 covariates

```
coef(regfit.full,11)
```

← we extract the coefficients with the model with 11 covariates

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  135.7512195   -2.1277482   6.9236994   5.6202755   -0.1389914   1.4553310
##         CRBI       CWalks      LeagueN   DivisionW      PutOuts      Assists
##    0.7852528   -0.8228559  43.1116152 -111.1460252   0.2894087   0.2688277
```

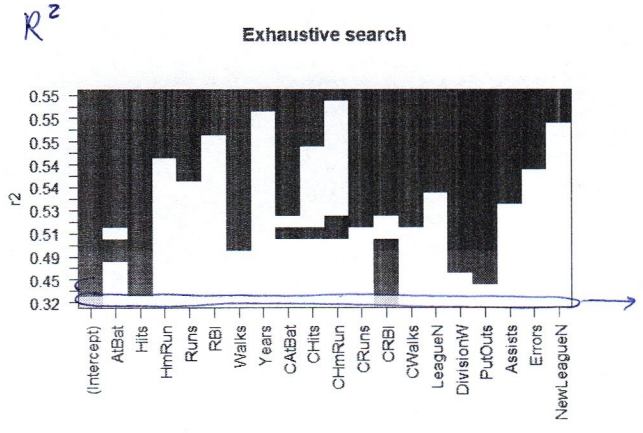
```
coef(regfit.full,6)
```

← coeff. for the model with 6 covariates

```
##  (Intercept)        AtBat         Hits        Walks         CRBI    DivisionW
##   91.5117981   -1.8685892   7.6043976   3.6976468   0.6430169 -122.9515338
##      PutOuts
##    0.2643076
```

```
# graphical table of best subsets
help(plot.regsubsets)

x11()
plot(regfit.full,scale="r2",main="Exhaustive search")
```

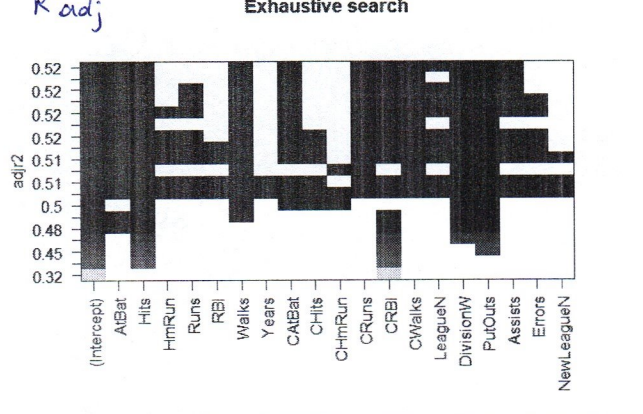
R^2



Interpretation:
the model with $R^2 = 0.32$ has only the intercept and CRBI

```
x11()
plot(regfit.full,scale="adjr2",main="Exhaustive search")
```

R^2_{adj}

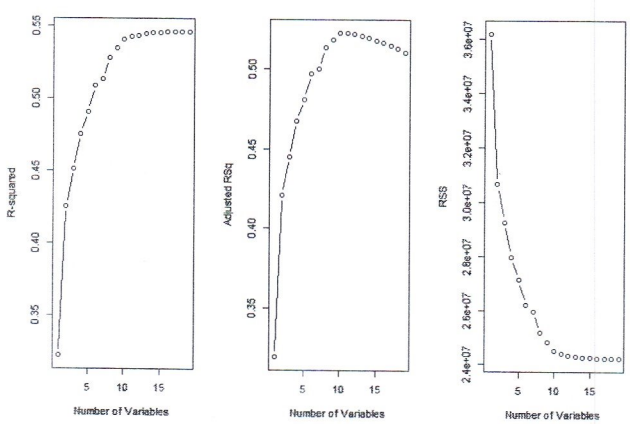



```
# Forward and Backward Stepwise Selection
regfit.fwd <- regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
summary(regfit.fwd)
```

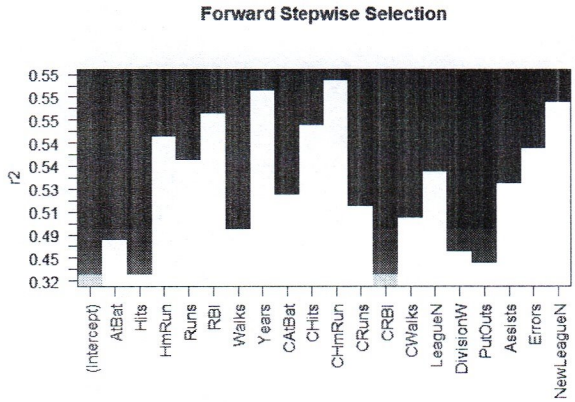
```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "forward")
## 19 Variables  (and intercept)
##             Forced in Forced out
## AtBat           FALSE      FALSE
## Hits            FALSE      FALSE
## HmRun           FALSE      FALSE
## Runs            FALSE      FALSE
## RBI             FALSE      FALSE
## Walks           FALSE      FALSE
## Years           FALSE      FALSE
## CAtBat          FALSE      FALSE
## CHits           FALSE      FALSE
## CHmRun          FALSE      FALSE
## CRuns           FALSE      FALSE
## CRBI            FALSE      FALSE
## CWalks          FALSE      FALSE
## LeagueN         FALSE      FALSE
## DivisionW       FALSE      FALSE
## PutOuts         FALSE      FALSE
## Assists         FALSE      FALSE
## Errors          FALSE      FALSE
## NewLeagueN      FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: forward
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 7  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 8  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   "*"
## 9  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " " "   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " " "   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " " "   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

This time the models are nested!

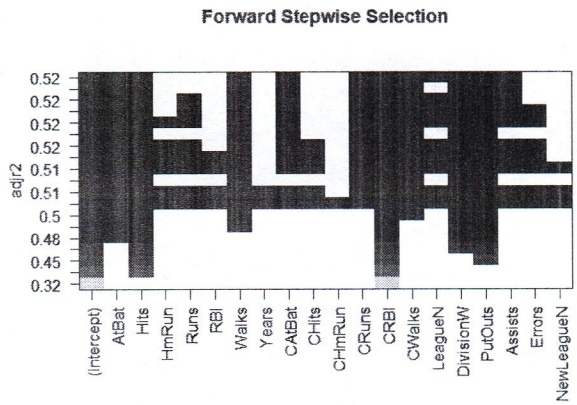
```
x11(height=7,width=14)
par(mfrow=c(1,3))
plot(summary(regfit.fwd)$rsq,xlab="Number of Variables",ylab="R-squared",type="b")
plot(summary(regfit.fwd)$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="b")
plot(summary(regfit.fwd)$rss,xlab="Number of Variables",ylab="RSS",type="b")
```




```
x11()
plot(regfit.fwd,scale="r2",main="Forward Stepwise Selection")
```




```
x11()
plot(regfit.fwd,scale="adjr2",main="Forward Stepwise Selection")
```

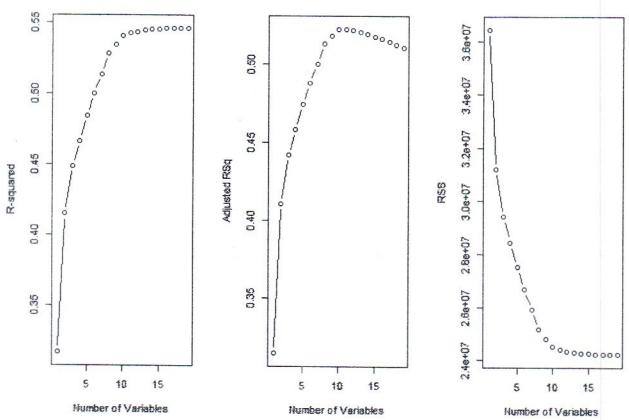



```
regfit.bwd <- regsubsets(Salary~.,data=Hitters,nvmax=19,method="backward")
summary(regfit.bwd)
```

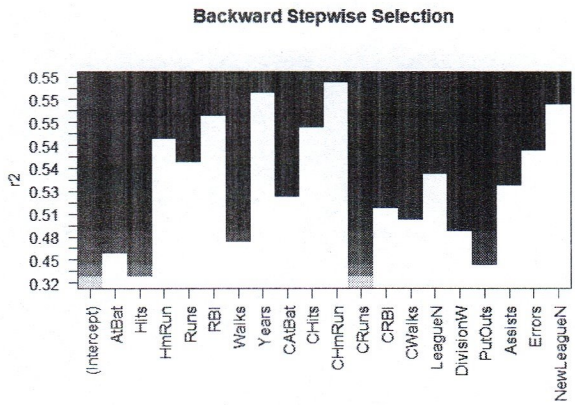
```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19, method = "backward")
## 19 Variables  (and intercept)
##             Forced in Forced out
## AtBat           FALSE      FALSE
## Hits            FALSE      FALSE
## HmRun           FALSE      FALSE
## Runs            FALSE      FALSE
## RBI             FALSE      FALSE
## Walks           FALSE      FALSE
## Years           FALSE      FALSE
## CAtBat          FALSE      FALSE
## CHits           FALSE      FALSE
## CHmRun          FALSE      FALSE
## CRuns           FALSE      FALSE
## CRBI            FALSE      FALSE
## CWalks          FALSE      FALSE
## LeagueN         FALSE      FALSE
## DivisionW       FALSE      FALSE
## PutOuts         FALSE      FALSE
## Assists         FALSE      FALSE
## Errors          FALSE      FALSE
## NewLeagueN      FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: backward
##          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 4  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    "*"   " "
## 5  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    " "   " "    "*"   " "
## 6  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    " "   " "    "*"   " "
## 7  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    " "   " "    "*"   " "
## 8  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    " "   " "    "*"   "*"
## 9  ( 1 ) "*"   "*"  " "   " "  " " " "   "*"   " "    "*"   " "    "*"   "*"
## 10  ( 1 ) "*"  "*"  " "   " "  " " " "   "*"   " "    "*"   " "    "*"   "*"
## 11  ( 1 ) "*"  "*"  " "   " "  " " " "   "*"   " "    "*"   " "    "*"   "*"
## 12  ( 1 ) "*"  "*"  " "   "*"  " " " "   "*"   " "    "*"   " "    "*"   "*"
## 13  ( 1 ) "*"  "*"  " "   "*"  " " " "   "*"   " "    "*"   " "    "*"   "*"
## 14  ( 1 ) "*"  "*"  "*"   "*"  " " " "   "*"   " "    "*"   " "    "*"   "*"
## 15  ( 1 ) "*"  "*"  "*"   "*"  " " "*"   "*"   " "    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"  "*"  "*"   "*"  "*" "*"   "*"   " "    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"  "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"  "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
## 19  ( 1 ) "*"  "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##          CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 ) " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 5  ( 1 ) " "    " "     " "       "*"     " "     " "    " "
## 6  ( 1 ) " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 ) "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"   " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"   "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"   "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"   "*"     "*"       "*"     "*"     "*"    "*"
```

This time we've starting from the bottom (including all of the variables) and we've removing one at the time

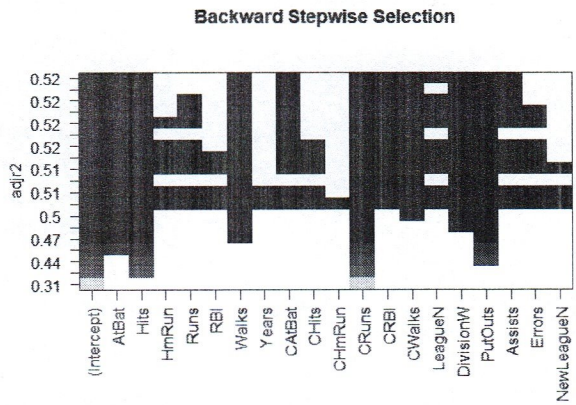
```
x11(height=7,width=14)
par(mfrow=c(1,3))
plot(summary(regfit.bwd)$rsq,xlab="Number of Variables",ylab="R-squared",type="b")
plot(summary(regfit.bwd)$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="b")
plot(summary(regfit.bwd)$rss,xlab="Number of Variables",ylab="RSS",type="b")
```




```
x11()
plot(regfit.bwd,scale="r2",main="Backward Stepwise Selection")
```




```
x11()
plot(regfit.bwd,scale="adjr2",main="Backward Stepwise Selection")
```




```
coef(regfit.full,7) # Exhaustive search
```

```
##  (Intercept)         Hits        Walks       CAtBat        CHits       CHmRun
##   79.4509472    1.2833513    3.2274264   -0.3752350    1.4957073    1.4420538
##    DivisionW      PutOuts
## -129.9866432    0.2366813
```

```
coef(regfit.fwd,7)  # Forward Stepwise Selection
```

they're different

```
##  (Intercept)         AtBat          Hits         Walks          CRBI        CWalks
##  109.7873062   -1.9588851    7.4498772    4.9131401    0.8537622   -0.3053070
##    DivisionW      PutOuts
## -127.1223928    0.2533404
```

```
coef(regfit.bwd,7)  # Backward Stepwise Selection
```

```
##  (Intercept)         AtBat          Hits         Walks         CRuns        CWalks
##  105.6487488   -1.9762838    6.7574914    6.0558691    1.1293095   -0.7163346
##    DivisionW      PutOuts
## -116.1692169    0.3028847
```

```
graphics.off()

### ------------------------------------------------------------------------
### Choosing among models using the k-fold cross-validation approach
### (exhaustive search)
### ------------------------------------------------------------------------
k <- 10
```

→ we divide the date in 10 groups (10 folds)

```
set.seed(1)
folds <- sample(1:k,nrow(Hitters),replace=TRUE)
folds
```

```
##   [1]  9  4  7  1  2  7  2  3  1  5  5 10  6 10  7  9  5  5  9  9  5  5  2 10  9
##  [26]  1  4  3  6 10 10  6  4  4 10  9  7  6  9  8  9  7  8  6 10  7  3 10  6  8
##  [51]  2  2  6  6  1  3  3  8  6  7  6  8  7  1  4  8  9  9  7  4  7  6  1  5  6
##  [76]  1  9  7  7  3  6  2 10 10  7  3  2 10  1 10 10  8 10  5  7  8  5  6  8  1
## [101]  3 10  3  1  6  6  4  9  5  1  3  6  3  7  3  3  1  9  2  8  6  1  2  7  7
## [126]  4  9  8  3  5  3  4  2  1  7  9 10 10  2  2  3  1  2  3  3  3  8  9  2 10
## [151]  8 10  4  5  9  5  7  5  6  4  2  1  3  8  9  6  1  4  5  9  5  8  4  1  9
## [176]  5  1  5  4 10 10  9  8  5  5  6  6  2  2  8  4 10  8  5  5  8  8  7  4  4
## [201]  1 10  4  9  9  9  9  6  6  4  3  3  9  9  7  9  5  7  4  4 10  8  1 10  2
## [226] 10  1  1  4  5  5  6  9  8  5  1  2  1  8  5  8 10  7  7  2  9  4  2  5  2
## [251]  4  3  6  9  7  5  5  1  1 10  1  3 10
```

```
table(folds)
```

```
## folds
##  1  2  3  4  5  6  7  8  9 10
## 28 22 24 24 30 26 25 24 31 29
```

```r
# function that performs the prediction for regsubsets
predict.regsubsets <- function(object,newdata,id){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form,newdata)
  coefi <- coef(object,id=id)
  xvars <- names(coefi)
  mat[,xvars]%*%coefi
}

cv.errors <- matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))
for(j in 1:k){
  best.fit <- regsubsets(Salary~.,data=Hitters[folds!=j,],nvmax=19)
  for(i in 1:19){
    pred <- predict(best.fit,Hitters[folds==j,],id=i)
    cv.errors[j,i] <- mean( (Hitters$Salary[folds==j]-pred)^2 )
  }
}
cv.errors
```

```
##              1        2        3        4        5        6        7
##  [1,]  98623.24 115600.61 120884.31 113831.63 120728.51 122922.93 155507.25
##  [2,] 155320.11 100425.87 168838.35 159729.47 145895.71 123555.25 119983.35
##  [3,] 124151.77  68833.50  69392.29  77221.37  83802.82  70125.41  68997.77
##  [4,] 232191.41 279001.29 294568.10 288765.81 276972.83 260121.22 276413.09
##  [5,] 115397.35  96807.44 108421.66 104933.55  99561.69  86103.05  89345.61
##  [6,] 103839.30  75652.50  69962.31  58291.91  65893.45  64215.56  65800.88
##  [7,]  85793.95  78506.34  80541.35  84213.50  87140.78  80669.98  86247.85
##  [8,] 273084.54 235423.66 230706.43 205624.81 223867.35 205559.00 206556.77
##  [9,] 178316.69 163857.60 142998.75 120697.54 115261.58 108791.71 102320.25
## [10,] 131492.54  95111.58 104956.38  96978.66  91377.54  73322.28  71687.88
##              8        9       10       11       12       13       14
##  [1,] 137753.36 149198.01 153332.89 155702.91 155842.88 158755.87 156037.17
##  [2,]  96609.16  99057.32  80375.78  91290.74  92292.69 100498.84 101562.45
##  [3,]  64143.70  65813.14  65120.27  68160.94  70263.77  69765.81  68987.54
##  [4,] 259923.88 270151.18 263492.31 259154.53 269017.80 265468.90 269666.65
##  [5,]  87693.15  91631.88  88763.37  89801.07  91070.44  92429.43  92821.15
##  [6,]  61413.45  60200.70  59599.54  59831.90  60081.48  59662.51  60618.91
##  [7,]  89643.01  92081.37  89057.16  88102.28  90885.98  95025.99  99172.32
##  [8,] 182678.23 179783.18 179916.36 173790.82 180508.48 185424.38 183257.89
##  [9,]  89418.23  84366.23  80188.72  80665.49  82509.05  85078.50  89243.38
## [10,]  66524.07  63281.82  62320.54  66011.39  65086.73  66097.41  73444.48
##             15       16       17       18       19
##  [1,] 157739.46 155548.96 156688.01 156860.92 156976.98
##  [2,] 104621.38 100922.27 102198.69 105318.26 106064.89
##  [3,]  69471.32  69294.21  69199.91  68866.84  69195.74
##  [4,] 265518.87 267240.44 267771.74 267670.66 267717.80
##  [5,]  95849.81  96513.70  95209.20  94952.21  94951.70
##  [6,]  62540.03  62776.81  62717.77  62354.97  62268.97
##  [7,]  99314.04 100192.99 100302.79  99772.60 100659.75
##  [8,] 183331.01 185159.62 183643.63 182587.35 183436.78
##  [9,]  90478.82  91782.20  91723.08  91140.55  91344.82
## [10,]  72351.39  71311.99  71393.23  71333.19  71417.75
```

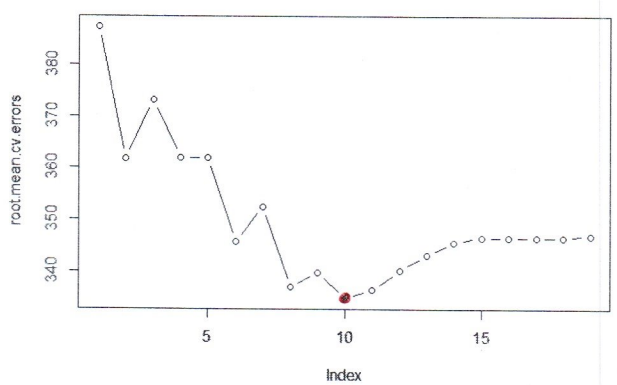
```r
root.mean.cv.errors <- sqrt(apply(cv.errors,2,mean)) # average over the columns
root.mean.cv.errors
```

```
##        1        2        3        4        5        6        7        8
## 387.0673 361.8315 372.9973 361.9790 362.0086 345.7436 352.5423 337.0164
##        9       10       11       12       13       14       15       16
## 339.9360 334.9876 336.5282 340.2292 343.2503 345.6605 346.5856 346.5174
##       17       18       19
## 346.5325 346.5339 346.9921
```

```r
x11()
plot(root.mean.cv.errors,type='b')

which.min(root.mean.cv.errors)
```

```
## 10
## 10
```

```r
points(which.min(root.mean.cv.errors),root.mean.cv.errors[which.min(root.mean.cv.errors)], col='red',pch=19)
```



Once we have selected the number of variables we go back and find out the coefficients:

we go back to the FULL DATASET!

```r
# estimation on the full dataset
reg.best <- regsubsets(Salary~.,data=Hitters, nvmax=19)
coef(reg.best,10)
```

```
##  (Intercept)        AtBat         Hits        Walks       CAtBat        CRuns
##  162.5354420   -2.1686501    6.9180175    5.7732246   -0.1300798    1.4082490
##         CRBI       CWalks    DivisionW      PutOuts      Assists
##    0.7743122   -0.8308264 -112.3800575    0.2973726    0.2831680
```

WRONG WAY!

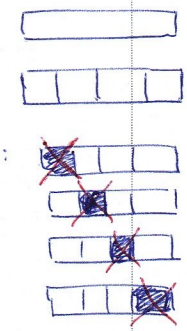
```
### ---------------------------------------------------------------------
### k-fold cross validation after model selection (  WRONG WAY!!  )
### ---------------------------------------------------------------------
best.fit <- regsubsets(Salary~.,data=Hitters,nvmax=19)
summary(best.fit)
```

the variable selection should not be performed BEFORE the cross validation loop

```
## Subset selection object
## Call: regsubsets.formula(Salary ~ ., data = Hitters, nvmax = 19)
## 19 Variables  (and intercept)
##            Forced in Forced out
## AtBat          FALSE      FALSE
## Hits           FALSE      FALSE
## HmRun          FALSE      FALSE
## Runs           FALSE      FALSE
## RBI            FALSE      FALSE
## Walks          FALSE      FALSE
## Years          FALSE      FALSE
## CAtBat         FALSE      FALSE
## CHits          FALSE      FALSE
## CHmRun         FALSE      FALSE
## CRuns          FALSE      FALSE
## CRBI           FALSE      FALSE
## CWalks         FALSE      FALSE
## LeagueN        FALSE      FALSE
## DivisionW      FALSE      FALSE
## PutOuts        FALSE      FALSE
## Assists        FALSE      FALSE
## Errors         FALSE      FALSE
## NewLeagueN     FALSE      FALSE
## 1 subsets of each size up to 19
## Selection Algorithm: exhaustive
##           AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI
## 1  ( 1 )  " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 2  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 3  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 4  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 5  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 6  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"
## 7  ( 1 )  " "   "*"  " "   " "  " " " "   " "   "*"    "*"   "*"    " "   " "
## 8  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   " "    " "   "*"    "*"   "*"
## 9  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   "*"    " "   " "    "*"   "*"
## 10  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   "*"    " "   " "    "*"   "*"
## 11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    " "   " "    "*"   "*"
## 15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"    "*"   " "    "*"   "*"
## 16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"    "*"   " "    "*"   "*"
## 17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   " "    "*"   "*"
## 18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
## 19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"    "*"   "*"    "*"   "*"
##           CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
## 1  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 2  ( 1 )  " "    " "     " "       " "     " "     " "    " "
## 3  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 4  ( 1 )  " "    " "     " "       "*"     " "     " "    " "
## 5  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 6  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 7  ( 1 )  " "    " "     "*"       "*"     " "     " "    " "
## 8  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 9  ( 1 )  "*"    " "     "*"       "*"     " "     " "    " "
## 10  ( 1 ) "*"    " "     "*"       "*"     "*"     " "    " "
## 11  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 12  ( 1 ) "*"    "*"     "*"       "*"     "*"     " "    " "
## 13  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 14  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 15  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 16  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    " "
## 17  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 18  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
## 19  ( 1 ) "*"    "*"     "*"       "*"     "*"     "*"    "*"
```

The process is:

1. full data:
2. division in k folds:
3. variable selection in the data -j:



Whenever we perform cross validation, a piece of the data is taken out at THE BEGINNING of WHATEVER ANALYSIS

!! Remember that with cross validation we're testing the method, not if the covariates selectioned at the begin are good for the data \ the j-th fold!

```
cv.errors_wrong <- matrix(NA,k,19, dimnames=list(NULL, paste(1:19)))
for(j in 1:k)
  for(i in 1:19){
    covariate <- which(summary(best.fit)$which[i,-c(1,19)])
    mod <- lm(Salary~.,data=Hitters[folds!=j,c(covariate,19)])
    pred <- predict(mod,Hitters)[folds==j]
    cv.errors_wrong[j,i] <- mean( (Hitters$Salary[folds==j]-pred)^2)
  }

cv.errors_wrong
```

```
##              1         2         3         4         5         6         7
##  [1,]  98623.24 100025.08 108594.60 113831.63 109928.48 122922.93 133223.96
##  [2,] 126554.96 100425.87  90243.27  85000.05  79143.84  75732.01 107282.84
##  [3,] 110257.03  68833.50  53791.82  55087.62  52797.37  59367.05  60602.13
##  [4,] 232191.41 279001.29 294568.10 272134.61 270315.96 260121.22 248007.23
##  [5,] 115397.35  96807.44 108421.66 104933.55  99561.69  86103.05  85638.42
##  [6,] 103839.30  75652.50  62963.06  58291.91  57685.35  64215.56  61562.62
##  [7,]  85793.95  78506.34  80541.35  84213.50  87140.78  80669.98  74185.42
##  [8,] 273084.54 215931.04 194759.90 191579.63 175494.24 170802.69 199833.42
##  [9,] 154315.15 120983.23 109399.88 102000.51  96120.50  91172.85  88804.98
## [10,] 131492.54  95111.58  79491.06  70383.63  76217.30  73322.28  66180.97
##               8         9        10        11        12        13        14
##  [1,] 131554.97 126071.10 131818.11 128937.50 130158.61 130158.61 129333.46
##  [2,]  71251.97  74656.56  80375.78  81379.62  81788.48  81788.48  84078.46
##  [3,]  64143.70  65813.14  65120.27  68160.94  70263.77  70263.77  69619.78
##  [4,] 252181.50 252222.15 245373.30 243108.69 258276.16 258276.16 257687.96
##  [5,]  82807.54  81658.21  88763.37  89801.07  89072.77  89072.77  89338.77
##  [6,]  58787.34  60200.70  59599.54  59831.90  59339.64  59339.64  60199.38
##  [7,]  84872.47  84106.87  89057.16  88102.28  87260.53  87260.53  90250.98
##  [8,] 182678.23 179783.18 172950.31 173790.82 172480.25 172480.25 172602.36
##  [9,]  88048.51  84366.23  80188.72  80665.49  82209.73  82209.73  82106.64
## [10,]  62257.89  63281.82  62320.54  61619.06  65086.73  65086.73  64647.76
##              15        16        17        18        19
##  [1,] 134743.16 134745.29 140117.06 141918.84 156803.71
##  [2,]  98090.98  99941.11 100768.60 102719.83 103028.27
##  [3,]  69720.79  69408.24  68622.08  68690.01  68863.06
##  [4,] 258337.65 258871.49 259389.89 261059.13 261023.89
##  [5,]  90512.85  93016.76  94603.75  95209.20  95199.86
##  [6,]  60557.16  60207.20  60975.37  61103.98  62372.10
##  [7,]  90045.24  89667.03  93765.08  98890.97 100069.89
##  [8,] 172777.92 175270.52 174182.90 179569.37 182587.35
##  [9,]  82636.23  82639.90  85552.16  85420.29  91140.55
## [10,]  65668.34  69228.92  68102.60  68237.91  71532.13
```

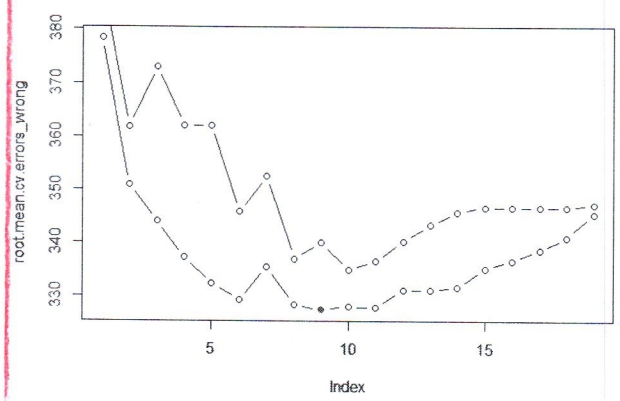
```
root.mean.cv.errors_wrong <- sqrt(apply(cv.errors_wrong,2,mean)) # average over the columns
root.mean.cv.errors_wrong
```

```
##        1        2        3        4        5        6        7        8
## 378.3582 350.8957 343.9149 337.2620 332.3260 329.3068 335.4582 328.4180
##        9       10       11       12       13       14       15       16
## 327.4385 327.9584 327.9325 331.0493 331.0493 331.6422 335.1254 336.6001
##       17       18       19
## 338.5380 341.0014 345.3434
```

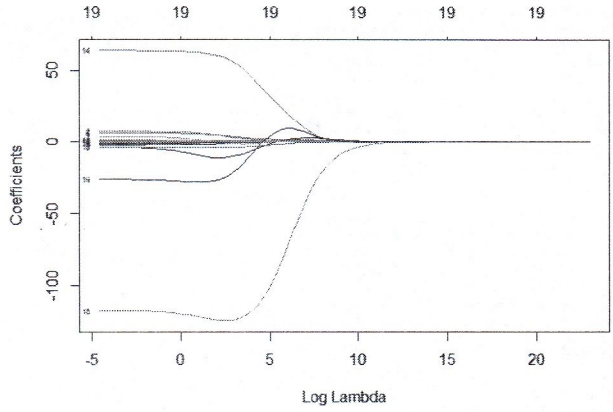
```
x11()
plot(root.mean.cv.errors_wrong,type='b')

which.min(root.mean.cv.errors_wrong)
```

```
## 9
## 9
```

```
points(which.min(root.mean.cv.errors_wrong),root.mean.cv.errors_wrong[which.min(root.mean.cv.errors_wrong)], col='red',pch=1
9)

points(root.mean.cv.errors,type='b',col='blue')
```




```
### ----------------------------------------------------------------------------
### Ridge regression (with glmnet)
### ----------------------------------------------------------------------------
x <- model.matrix(Salary~.,Hitters)[,-1] # predictor matrix
y <- Hitters$Salary # response
grid <- 10^seq(10,-2,length=100) # grid of lambda

# Ridge regression

ridge.mod <- glmnet(x,y,alpha=0,lambda=grid)

x11()
plot(ridge.mod,xvar='lambda',label=TRUE)
```



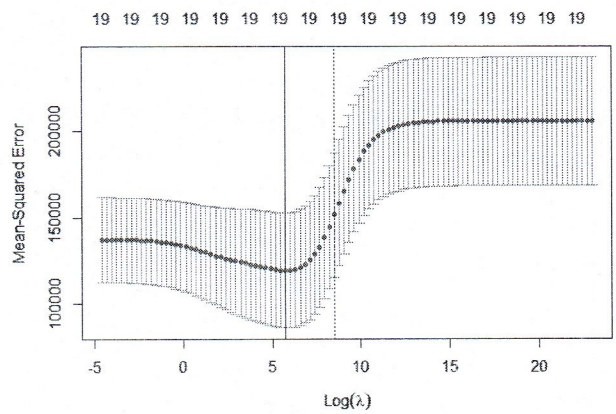

```
# choosing the parameter Lambda
set.seed(123)
cv.out <- cv.glmnet(x,y,alpha=0,nfold=3,lambda=grid)

x11()
plot(cv.out)

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

```
## [1] 305.3856
```

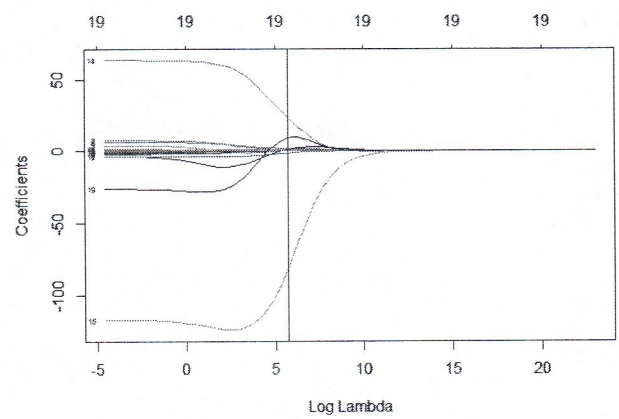
```
log(bestlam.ridge)
```

```
## [1] 5.721575
```

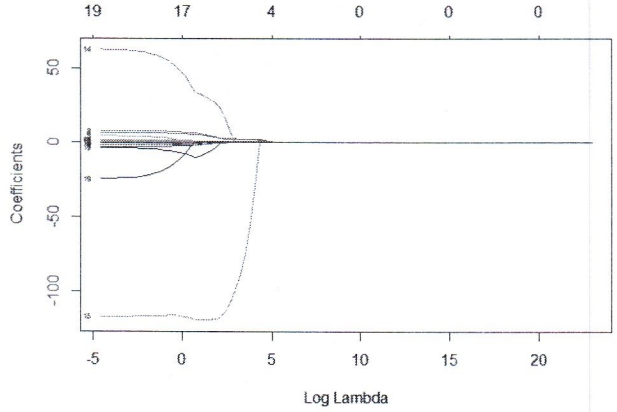
```
abline(v=log(bestlam.ridge))
```




```
x11()
plot(ridge.mod,xvar='lambda',label=TRUE)
abline(v=log(bestlam.ridge))
```




```
### ------------------------------------------------------------------------
### Lasso regression (with glmnet)
### ------------------------------------------------------------------------
lasso.mod <- glmnet(x,y,alpha=1,lambda=grid)

x11()
plot(lasso.mod,xvar='lambda',label=TRUE)
```



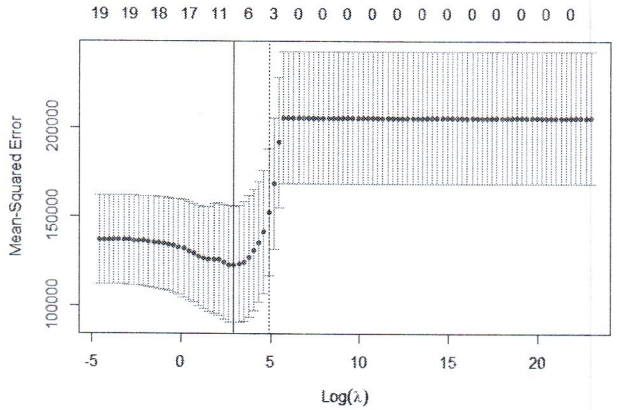

```
# choosing the parameter Lambda
set.seed(123)
cv.out <- cv.glmnet(x,y,alpha=1,nfold=3,lambda=grid)

x11()
plot(cv.out)

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

```
## [1] 18.73817
```

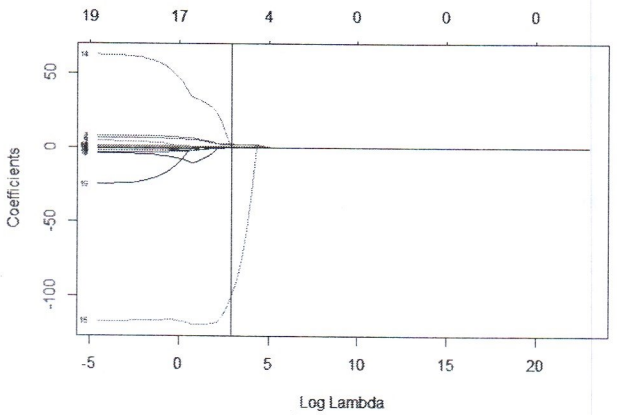
```
log(bestlam.lasso)
```

```
## [1] 2.930563
```

```
abline(v=log(bestlam.lasso))
```




```
x11()
plot(lasso.mod,xvar='lambda',label=TRUE)
abline(v=log(bestlam.lasso))
```



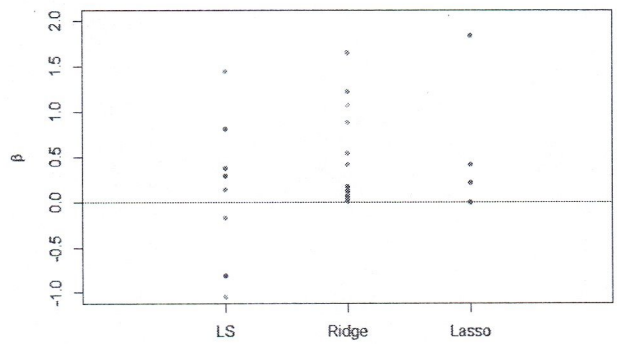

```
### ------------------------------------------------------------------------
# Compare coefficients estimates for LS, Ridge and Lasso

coef.ridge <- predict(ridge.mod, s=bestlam.ridge, type = 'coefficients')[1:20,]
coef.lasso <- predict(lasso.mod, s=bestlam.lasso, type = 'coefficients')[1:20,]
coef.ridge
```

```
##  (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##  13.69965470   0.07214526   0.88059238   0.54367980   1.07217873   0.87912033
##        Walks        Years       CAtBat        CHits       CHmRun        CRuns
##   1.64903804   1.22528239   0.01132160   0.05819360   0.41084586   0.11626187
##         CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##   0.12363705   0.05071118  22.84517646 -81.01744688   0.17009772   0.03096459
##       Errors   NewLeagueN
##  -1.42769526   8.96604057
```

```
coef.lasso
```

```
## (Intercept)        AtBat         Hits        HmRun         Runs          RBI
##  24.6392478   0.0000000   1.8499430   0.0000000   0.0000000   0.0000000
##       Walks        Years       CAtBat        CHits       CHmRun        CRuns
##   2.1961005   0.0000000   0.0000000   0.0000000   0.0000000   0.2058504
##        CRBI       CWalks      LeagueN    DivisionW      PutOuts      Assists
##   0.4095924   0.0000000   0.0000000 -99.9736534   0.2158902   0.0000000
##       Errors   NewLeagueN
##   0.0000000   0.0000000
```

```
x11()
plot(rep(0, dim(x)[2]), coef(lm(y~x))[-1], col=rainbow(dim(x)[2]), pch=20, xlim=c(-1,3), ylim=c(-1,2), xlab='', ylab=express
ion(beta),
     axes=F)
points(rep(1, dim(x)[2]), coef.ridge[-1], col=rainbow(dim(x)[2]), pch=20)
points(rep(2, dim(x)[2]), coef.lasso[-1], col=rainbow(dim(x)[2]), pch=20)
abline(h=0, col='grey41', lty=1)
box()
axis(2)
axis(1, at=c(0,1,2), labels = c('LS', 'Ridge', 'Lasso'))
```




```
# L2 norm
sum((coef(lm(y~x))[-1])^2) # LS
```

```
## [1] 18337.3
```

```
sum((coef.ridge[-1])^2)    # ridge
```

```
## [1] 7175.61
```

```
# L1 norm
sum(abs(coef(lm(y~x))[-1])) # LS
```

```
## [1] 238.7295
```

```
sum(abs(coef.lasso[-1]))    # Lasso
```

```
## [1] 104.851
```

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Logistic Regression
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
help(glm)

### ------------------------------------------------------------------------
###  Pb2 of 23/07/2009
### ------------------------------------------------------------------------
# A store of an appliance chain sells two types of televisions: with 4:3
# screen and with 16:9 screen. The TV.txt file shows the number of
# televisions sold annually for both types of televisions from 1999
# to 2008. By introducing an appropriate logistic model and estimating
# the parameters with the maximum likelihood method:
# a) comment the residual deviance by comparing it with that of the model
#    without regressor.
# Assuming that the store is representative of the Italian situation:
# b) provide a pointwise estimate for the proportion of 16:9 televisions
#    sold in Italy in 2009;
# c) provide a pointwise estimate for the year in which sales of 16:9
#    televisions exceeded those of 4:3;
# d) provide a pointwise estimate for the year in which 16:9 televisions
#    will cover (or have covered) 99% of the Italian market.

TV <- read.table('TV.txt', header=T)
head(TV)
```

```
##       Anno         Tipo
## 1 2006.549   sedicinoni
## 2 2007.082   sedicinoni
## 3 2002.237   sedicinoni
## 4 2006.052   sedicinoni
## 5 2000.754 quattroterzi
## 6 2007.529   sedicinoni
```

```
dim(TV)
```

```
## [1] 198   2
```

```
#a)
fit <- glm(as.factor(Tipo) ~ Anno, data=TV, family='binomial')
summary(fit)
```

```
##
## Call:
## glm(formula = as.factor(Tipo) ~ Anno, family = "binomial", data = TV)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5109  -0.4890   0.1591   0.3787   2.0030
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1808.2102   264.4591  -6.837 8.06e-12 ***
## Anno            0.9035     0.1321   6.838 8.01e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 235.86  on 197  degrees of freedom
## Residual deviance: 126.57  on 196  degrees of freedom
## AIC: 130.57
##
## Number of Fisher Scoring iterations: 6
```

```
plot(TV$Anno, as.numeric(as.factor(TV$Tipo))-1)
lines(seq(1997, 2010, by=0.1),
      predict(fit, data.frame(Anno=seq(1997,2010,by=0.1)), type='response'))

# null model
Freq.tot <- table(TV$Tipo)[2]/ (table(TV$Tipo)[1] + table(TV$Tipo)[2])
abline(h = Freq.tot, col='blue', lty=2)

#b)
predict(fit, data.frame(Anno=2009), type='response')
```

```
##         1
## 0.9990309
```

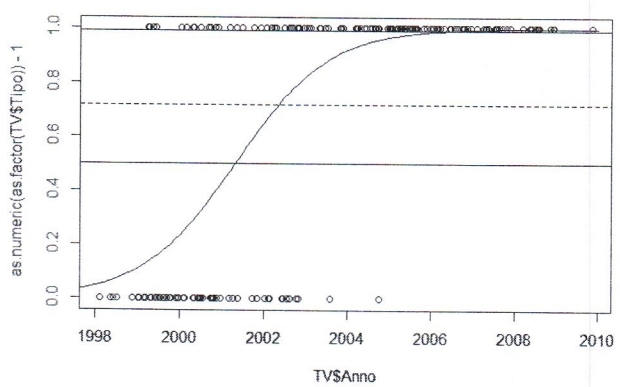
```
#c)
(log(0.5/0.5) - coefficients(fit)[1]) / coefficients(fit)[2]
```

```
## (Intercept)
##    2001.321
```

```
abline(h=0.5, col='red')

#d)
(log(0.99/0.01) - coefficients(fit)[1]) / coefficients(fit)[2]
```

```
## (Intercept)
##    2006.407
```

```
abline(h=0.99, col='red')
```



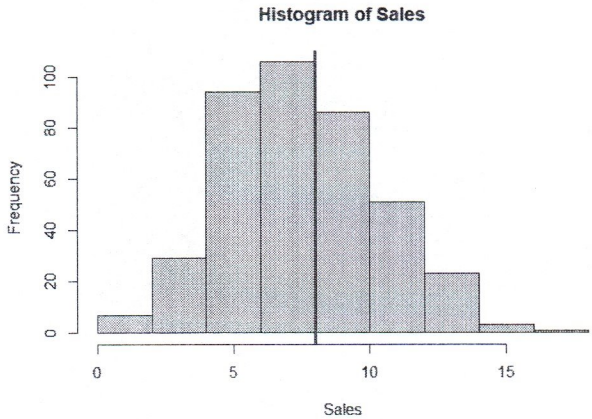

```
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Classification and regression trees
### ------------------------------------------------------------------------
### ------------------------------------------------------------------------
### Classification Trees: Carseats dataset
### ------------------------------------------------------------------------
attach(Carseats)

help(Carseats)
dim(Carseats)
```

```
## [1] 400  11
```

```
names(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

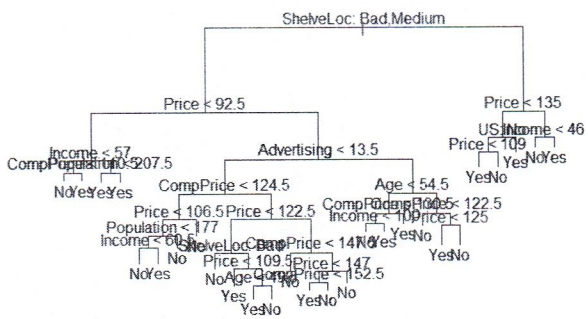
```
hist(Sales)
abline(v=8,lwd=3,col='red')
```




```
High <- ifelse(Sales<=8,"No","Yes")

Carseats <- data.frame(Carseats,High)

library(tree)
help(tree)

tree.carseats <- tree(as.factor(High)~.-Sales,Carseats)
summary(tree.carseats)
```

```
##
## Classification tree:
## tree(formula = as.factor(High) ~ . - Sales, data = Carseats)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
## [6] "Advertising" "Age"         "US"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

```
plot(tree.carseats)
text(tree.carseats,pretty=0)
```




```
tree.carseats
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
##   1) root 400 541.500 No ( 0.59000 0.41000 )
##     2) ShelveLoc: Bad,Medium 315 390.600 No ( 0.68889 0.31111 )
##       4) Price < 92.5 46  56.530 Yes ( 0.30435 0.69565 )
##         8) Income < 57 10  12.220 No ( 0.70000 0.30000 )
##          16) CompPrice < 110.5 5   0.000 No ( 1.00000 0.00000 ) *
##          17) CompPrice > 110.5 5   6.730 Yes ( 0.40000 0.60000 ) *
##         9) Income > 57 36  35.470 Yes ( 0.19444 0.80556 )
##          18) Population < 207.5 16  21.170 Yes ( 0.37500 0.62500 ) *
##          19) Population > 207.5 20   7.941 Yes ( 0.05000 0.95000 ) *
##       5) Price > 92.5 269 299.800 No ( 0.75465 0.24535 )
##        10) Advertising < 13.5 224 213.200 No ( 0.81696 0.18304 )
##          20) CompPrice < 124.5 96  44.890 No ( 0.93750 0.06250 )
##            40) Price < 106.5 38  33.150 No ( 0.84211 0.15789 )
##              80) Population < 177 12  16.300 No ( 0.58333 0.41667 )
##               160) Income < 60.5 6   0.000 No ( 1.00000 0.00000 ) *
##               161) Income > 60.5 6   5.407 Yes ( 0.16667 0.83333 ) *
##              81) Population > 177 26   8.477 No ( 0.96154 0.03846 ) *
##            41) Price > 106.5 58   0.000 No ( 1.00000 0.00000 ) *
##          21) CompPrice > 124.5 128 150.200 No ( 0.72656 0.27344 )
##            42) Price < 122.5 51  70.680 Yes ( 0.49020 0.50980 )
##              84) ShelveLoc: Bad 11   6.702 No ( 0.90909 0.09091 ) *
##              85) ShelveLoc: Medium 40  52.930 Yes ( 0.37500 0.62500 )
##               170) Price < 109.5 16   7.481 Yes ( 0.06250 0.93750 ) *
##               171) Price > 109.5 24  32.600 No ( 0.58333 0.41667 )
##                 342) Age < 49.5 13  16.050 Yes ( 0.30769 0.69231 ) *
##                 343) Age > 49.5 11   6.702 No ( 0.90909 0.09091 ) *
##            43) Price > 122.5 77  55.540 No ( 0.88312 0.11688 )
##              86) CompPrice < 147.5 58  17.400 No ( 0.96552 0.03448 ) *
##              87) CompPrice > 147.5 19  25.010 No ( 0.63158 0.36842 )
##               174) Price < 147 12  16.300 Yes ( 0.41667 0.58333 )
##                 348) CompPrice < 152.5 7   5.742 Yes ( 0.14286 0.85714 ) *
##                 349) CompPrice > 152.5 5   5.004 No ( 0.80000 0.20000 ) *
##               175) Price > 147 7   0.000 No ( 1.00000 0.00000 ) *
##        11) Advertising > 13.5 45  61.830 Yes ( 0.44444 0.55556 )
##          22) Age < 54.5 25  25.020 Yes ( 0.20000 0.80000 )
##            44) CompPrice < 130.5 14  18.250 Yes ( 0.35714 0.64286 )
##              88) Income < 100 9  12.370 No ( 0.55556 0.44444 ) *
##              89) Income > 100 5   0.000 Yes ( 0.00000 1.00000 ) *
##            45) CompPrice > 130.5 11   0.000 Yes ( 0.00000 1.00000 ) *
##          23) Age > 54.5 20  22.490 No ( 0.75000 0.25000 )
##            46) CompPrice < 122.5 10   0.000 No ( 1.00000 0.00000 ) *
##            47) CompPrice > 122.5 10  13.860 No ( 0.50000 0.50000 )
##              94) Price < 125 5   0.000 Yes ( 0.00000 1.00000 ) *
##              95) Price > 125 5   0.000 No ( 1.00000 0.00000 ) *
##     3) ShelveLoc: Good 85  90.330 Yes ( 0.22353 0.77647 )
##       6) Price < 135 68  49.260 Yes ( 0.11765 0.88235 )
##        12) US: No 17  22.070 Yes ( 0.35294 0.64706 )
##          24) Price < 109 8   0.000 Yes ( 0.00000 1.00000 ) *
##          25) Price > 109 9  11.460 No ( 0.66667 0.33333 ) *
##        13) US: Yes 51  16.880 Yes ( 0.03922 0.96078 ) *
##       7) Price > 135 17  22.070 No ( 0.64706 0.35294 )
##        14) Income < 46 6   0.000 No ( 1.00000 0.00000 ) *
##        15) Income > 46 11  15.160 Yes ( 0.45455 0.54545 ) *
```

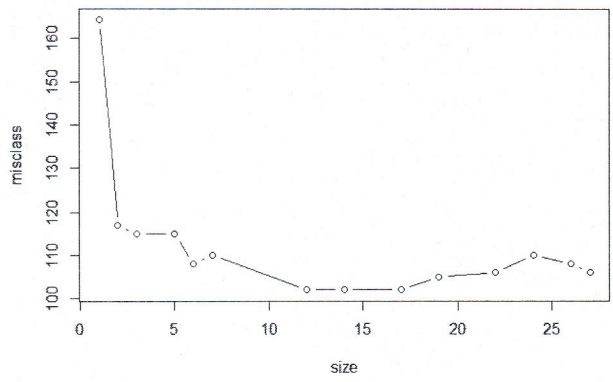
```
# use cross validation to prune the tree optimally
help(cv.tree)

set.seed(1)
cv.carseats <- cv.tree(tree.carseats,FUN=prune.misclass)

names(cv.carseats)
```

```
## [1] "size"   "dev"    "k"      "method"
```

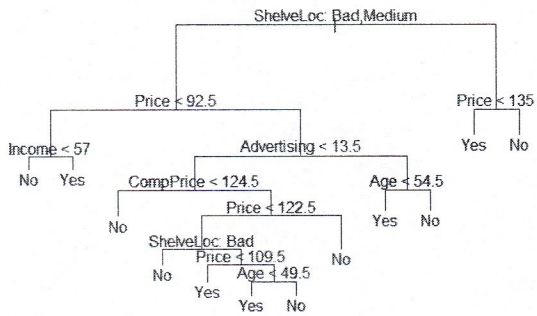
```
cv.carseats
```

```
## $size
##  [1] 27 26 24 22 19 17 14 12  7  6  5  3  2  1
##
## $dev
##  [1] 106 108 110 106 105 102 102 102 110 108 115 115 117 164
##
## $k
##  [1]      -Inf  0.000000  0.500000  1.000000  1.333333  1.500000  1.666667
##  [8]  2.500000  3.800000  4.000000  5.000000  7.500000 18.000000 47.000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.carseats$size,cv.carseats$dev,type="b",xlab="size",ylab="misclass")
```




```
help(prune.misclass)
prune.carseats <- prune.misclass(tree.carseats,best=12)

plot(prune.carseats)
text(prune.carseats,pretty=0)
```




```
detach(Carseats)

### ------------------------------------------------------------------------
### Regression Trees: Boston housing dataset
### ------------------------------------------------------------------------
help(Boston)
dim(Boston)
```

```
## [1] 506  14
```

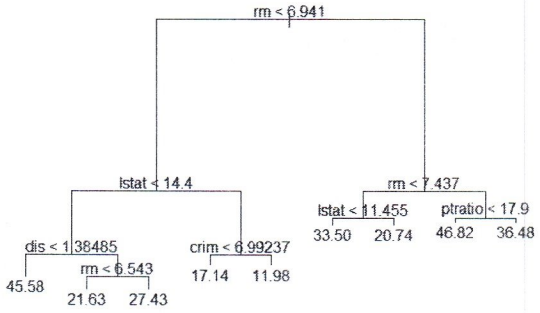
```
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

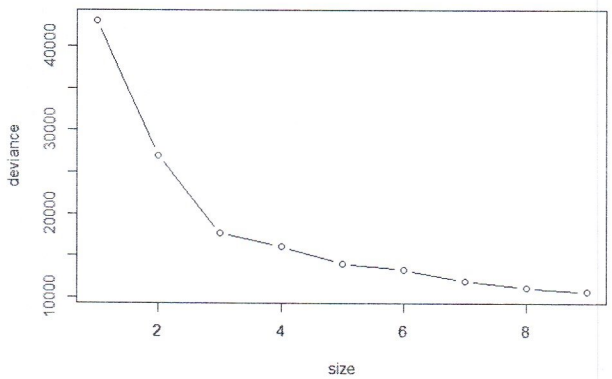
```
tree.boston <- tree(medv~.,Boston)
summary(tree.boston)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = Boston)
## Variables actually used in tree construction:
## [1] "rm"      "lstat"   "dis"     "crim"    "ptratio"
## Number of terminal nodes:  9
## Residual mean deviance:  13.55 = 6734 / 497
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -17.68000  -2.23000   0.07026   0.00000   2.22100  16.50000
```

```
plot(tree.boston)
text(tree.boston,pretty=0)
```




```
cv.boston <- cv.tree(tree.boston)

plot(cv.boston$size,cv.boston$dev,type='b',xlab='size',ylab='deviance')
```




```
prune.boston <- prune.tree(tree.boston,best=4)

plot(prune.boston)
text(prune.boston,pretty=0)
```