Linear regression 02

Josep Fortiana 2019-10-08

1. Multicollinearity and condition number

Acetylene data

Download the files: `Acetylene.dat.txt` (data description), `Acetylene.txt` (actual data). We will use these data for multiple linear regression.

```
Acetylene<-read.table("Acetylene.txt",header=TRUE)
str(Acetylene)
```

```
## 'data.frame':    16 obs. of  4 variables:
##  $ X1: int  1300 1300 1300 1300 1300 1300 1200 1200 1200 1200 ...
##  $ X2: num  7.5 9 11 13.5 17 23 5.3 7.5 11 13.5 ...
##  $ X3: num  0.012 0.012 0.0115 0.013 0.0135 0.012 0.04 0.038 0.032 0.026 ...
##  $ Y : num  49 50.2 50.5 48.5 47.5 44.5 28 31.5 34.5 35 ...
```

A description of data

```
summary(Acetylene)
```

```
##        X1             X2              X3                 Y
##  Min.   :1100   Min.   : 5.30   Min.   :0.01150   Min.   :15.00
##  1st Qu.:1175   1st Qu.: 7.50   1st Qu.:0.01275   1st Qu.:29.12
##  Median :1200   Median :11.00   Median :0.03300   Median :36.50
##  Mean   :1212   Mean   :12.44   Mean   :0.04031   Mean   :36.11
##  3rd Qu.:1300   3rd Qu.:17.00   3rd Qu.:0.05175   3rd Qu.:47.75
##  Max.   :1300   Max.   :23.00   Max.   :0.09800   Max.   :50.50
```

```
round(cor(Acetylene),2)
```

```
##       X1    X2    X3     Y
## X1  1.00  0.22 -0.96  0.95
## X2  0.22  1.00 -0.24  0.37
## X3 -0.96 -0.24  1.00 -0.91
## Y   0.95  0.37 -0.91  1.00
```

Linear model with a linear influence of each of the three available predictors *(main effects)*

```
Acetylene.lm.1<-lm(Y~X1+X2+X3,data=Acetylene)
summary(Acetylene.lm.1)
```

```
##
## Call:
## lm(formula = Y ~ X1 + X2 + X3, data = Acetylene)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.920 -1.856  0.234  2.074  6.948
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -121.26962   55.43571  -2.188   0.0492 *
## X1             0.12685    0.04218   3.007   0.0109 *
```

```
## X2             0.34816    0.17702   1.967   0.0728 .
## X3           -19.02170  107.92824  -0.176   0.8630
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.767 on 12 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.8998
## F-statistic: 45.88 on 3 and 12 DF,  p-value: 7.522e-07
```

Syntax for prediction

Assume we have a (or several) new observation(s) of $X1$, $X2$, $X3$. What does the model `Acetylene.lm.1` predict for the corresponding value(s) of Y?

$X1 = 1200$, $X2 = 11.0$, $X3 = 0.033$

Here I have chosen the median values, just as an example. Any value within the predictor ranges will do. Prediction for values outside this range (extrapolation) is possible but one should be wary of large errors.

Procedure is to build a `data.frame` with the new values (including variable names) and use the predict function:

```
new.observation<-data.frame(X1=1200,X2=11.0,X3=0.033)
Acetylene.prediction.1<-predict(Acetylene.lm.1,newdata=new.observation,type="response")
Acetylene.prediction.2<-predict(Acetylene.lm.1,newdata=new.observation,type="response", interval="confi
Acetylene.prediction.3<-predict(Acetylene.lm.1,newdata=new.observation,type="response", interval="predi
```

```
#str(Acetylene.prediction.1)
Acetylene.prediction.1
```

```
##        1
## 34.15702
```

```
#str(Acetylene.prediction.2)
Acetylene.prediction.2
```

```
##        fit      lwr      upr
## 1 34.15702 30.58172 37.73232
```

```
#str(Acetylene.prediction.3)
Acetylene.prediction.3
```

```
##        fit      lwr      upr
## 1 34.15702 25.20437 43.10967
```

Investigate multicollinearity

Extract the *regression (or model) matrix,* i.e., the result of pasting a column of ones to the matrix containing the values of `X1`, `X2`, `X3`. The new column will correspond to the `(Intercept)` regression term.

```
X<-model.matrix(Acetylene.lm.1)
X
```

```
##    (Intercept)   X1   X2     X3
## 1            1 1300  7.5 0.0120
## 2            1 1300  9.0 0.0120
## 3            1 1300 11.0 0.0115
## 4            1 1300 13.5 0.0130
## 5            1 1300 17.0 0.0135
```

```
## 6             1 1300 23.0 0.0120
## 7             1 1200  5.3 0.0400
## 8             1 1200  7.5 0.0380
## 9             1 1200 11.0 0.0320
## 10            1 1200 13.5 0.0260
## 11            1 1200 17.0 0.0340
## 12            1 1200 23.0 0.0410
## 13            1 1100  5.3 0.0840
## 14            1 1100  7.5 0.0980
## 15            1 1100 11.0 0.0920
## 16            1 1100 17.0 0.0860
## attr(,"assign")
## [1] 0 1 2 3
```

```r
round(cor(X[,-1]),3)
```

```
##        X1     X2     X3
## X1  1.000  0.224 -0.958
## X2  0.224  1.000 -0.240
## X3 -0.958 -0.240  1.000
```

Matrix $Q = X' \cdot X$ of cross-products, its determinant and its inverse:

```r
Q<- t(X) %*% X
round(Q,3)
```

```
##             (Intercept)         X1         X2      X3
## (Intercept)      16.000    19400.0    199.100   0.645
## X1            19400.000 23620000.0 242940.000 745.400
## X2              199.100   242940.0   2958.430   7.381
## X3                0.645      745.4      7.381   0.041
```

```r
round(det(Q),3)
```

```
## [1] 868196.3
```

```r
Q1<-solve(Q)
round(Q1,3)
```

```
##             (Intercept)     X1     X2       X3
## (Intercept)     216.556 -0.165 -0.048 -406.513
## X1               -0.165  0.000  0.000    0.307
## X2               -0.048  0.000  0.002    0.125
## X3             -406.513  0.307  0.125  820.847
```

Regression coefficients estimates $\hat{\beta}$:

$$\hat{\beta} = (X' \cdot X)^{-1} \cdot X' \cdot y.$$

y is the column of responses.

```r
y<-as.matrix(Acetylene$Y)
Q1%*%t(X) %*% y
```

```
##                     [,1]
## (Intercept) -121.2696214
## X1             0.1268539
## X2             0.3481576
## X3           -19.0216969
```

Compute V, variances and covariances matrix of the regression coefficients estimators $\hat{\beta}$, in two ways:

(1) Directly from the above formula, using the fact that $\mathrm{Var}(y) = \sigma^2 I$, substituting the ML estimate $\hat{\sigma}^2$.

(2) With the `vcov()` function in R.

```
sigma.hat<-summary(Acetylene.lm.1)$sigma
V<-sigma.hat^2*Q1
round(V,3)
```

```
##             (Intercept)     X1     X2        X3
## (Intercept)    3073.118 -2.335 -0.676 -5768.763
## X1               -2.335  0.002  0.000     4.352
## X2               -0.676  0.000  0.031     1.778
## X3            -5768.763  4.352  1.778 11648.504
```

```
vcov(Acetylene.lm.1)
```

```
##               (Intercept)            X1            X2           X3
## (Intercept)  3073.1176892 -2.3350689880 -0.6755487518 -5768.762894
## X1             -2.3350690  0.0017793193  0.0001764518     4.352212
## X2             -0.6755488  0.0001764518  0.0313350447     1.777998
## X3          -5768.7628939  4.3522124727  1.7779975238 11648.503915
```

Diagonal entries in V are the variances of the coefficient estimators. We observe two of them are quite large, indicating model instability.

This behaviour can be more precisely detected by using the *Variance Inflation Factors (VIF),* which we compute with the `vif()` function from the `car` package. Intuitively, VIF's quantify *multicollinearity* in the data, that is, linear dependences in the set of predictors. The VIF of a predictor is the proportion of its actual variance relative to the one it would have if it were linearly independent from the others:

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

```
## Loading required package: car
```

```
## Loading required package: carData
```

```
#?vif
```

```
vif(Acetylene.lm.1)
```

```
##        X1        X2        X3
## 12.225045  1.061838 12.324964
```

The commonly accepted rule of thumb is that a VIF larger than 10 is too large, indicating multicollinearity.

Another quantity used in multicollinearity detection is the *condition number* of the model matrix X. In general, the condition number $\kappa(A)$ of a matrix A measures the numerical inaccuracy introduced by solving the equation $A \cdot x = b$ for a given column vector b. Technically it is the ratio between the maximum and minimum singular values of A. When $\kappa \approx 1$ there is no significant precision loss. When $\kappa = 10^k$ the precision loss of k significant decimal digits:

The `kappa()` function in R gives the condition number of a matrix. We check the condition number of the model matrix X of this regression:

```
kappa(X)
```

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

We see that these data have bad condition for regression. The practical consequence of this fact is that predictions $\hat{y}$ obtained from this model are unreliable.

The predicted $\hat{y}$ for a new x can be computed as usual:

```
newy<-predict(Acetylene.lm.1,newdata=data.frame(X1=1100,X2=8,X3=0.02), type="response")
round(newy,3)
```

```
##      1
## 20.674
```

However, due to the bad condition of the model it would be unadvisable to rely upon this value, which possibly will have a large error (and we do not know how large).

```
round(predict(Acetylene.lm.1,newdata=data.frame(X1=1100,X2=8,X3=0.02), type="response",interval="confid
```

```
##      fit   lwr   upr
## 1 20.674 5.389 35.96
```

```
round(predict(Acetylene.lm.1,newdata=data.frame(X1=1100,X2=8,X3=0.02), type="response",interval="predict
```

```
##      fit   lwr    upr
## 1 20.674 3.324 38.024
```

We should restate the model, either replacing the original predictors with suitable linear combinations, designed to diminish multicollinearity, such as *PCR* and *PLS,* or by means of *regularization.*

GPA (Grade Point Average) dataset

The GPA dataset has 30 observacions (for 30 students) of 5 variables:

`gpa`: "graduate grade point average"

`greq`: "GRE exam quantitative score"

`grev`: "GRE exam verbal score"

`mat`: "Miller Analogies Test score"

`ar`: "rating by professors"

Read the data file into R:

```
gpa<-read.table("gpa.txt",header=TRUE)
str(gpa)
```

```
## 'data.frame':    30 obs. of  5 variables:
##  $ gpa : num  3.2 4.1 3 2.6 3.7 4 4.3 2.7 3.6 4.1 ...
##  $ greq: int  625 575 520 545 520 655 630 500 605 555 ...
##  $ grev: int  540 680 480 520 490 535 720 500 575 690 ...
##  $ mat : int  65 75 65 55 75 65 75 75 65 75 ...
##  $ ar  : num  2.7 4.5 2.5 3.1 3.6 4.3 4.6 3 4.7 3.4 ...
```

Fit a linear model to predict the response gpa from the remaining four variables:

```
gpa.lm.1<-lm(gpa~.,data=gpa)
summary(gpa.lm.1)
```

```
##
## Call:
## lm(formula = gpa ~ ., data = gpa)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7876 -0.2297  0.0069  0.2673  0.5260
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.738107   0.950740  -1.828   0.0795 .
## greq         0.003998   0.001831   2.184   0.0385 *
## grev         0.001524   0.001050   1.451   0.1593
## mat          0.020896   0.009549   2.188   0.0382 *
## ar           0.144234   0.113001   1.276   0.2135
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3874 on 25 degrees of freedom
## Multiple R-squared:  0.6405, Adjusted R-squared:  0.5829
## F-statistic: 11.13 on 4 and 25 DF,  p-value: 2.519e-05
```

Compute VIF and condition number

```r
vif(gpa.lm.1)
```

```
##     greq     grev      mat       ar
## 1.530841 1.469468 1.506900 1.734779
```

```r
X.gpa<-model.matrix(gpa.lm.1)
kappa(X.gpa)
```

```
## [1] 5506.107
```

The condition number κ is not small, but the VIF's are acceptable.

Exploring the effect of adding a new, redundant, variable

A *"new"* artificial predictor variable adding no new information. A linear combination plus a random noise:

```r
new<-gpa$greq+gpa$grev+gpa$mat+rnorm(30,0,0.5)
gpanew<-data.frame(gpa,new)
```

New linear model using the new, enlarged, predictor set

```r
gpa.lm.2<-lm(gpa~.,data=gpanew)
summary(gpa.lm.2)
```

```
##
## Call:
## lm(formula = gpa ~ ., data = gpanew)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84841 -0.23556  0.02753  0.26769  0.59109
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.7973     0.9580  -1.876   0.0728 .
## greq          0.1386     0.1554   0.892   0.3812
## grev          0.1365     0.1557   0.876   0.3896
```

```
## mat            0.1548     0.1548   1.000   0.3274
## ar             0.1646     0.1160   1.419   0.1687
## new           -0.1347     0.1555  -0.866   0.3948
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3893 on 24 degrees of freedom
## Multiple R-squared:  0.6514, Adjusted R-squared:  0.5787
## F-statistic: 8.969 on 5 and 24 DF,  p-value: 6.531e-05
```

VIF's and condition number of the new model

```r
vif(gpa.lm.2)
```

```
##         greq         grev          mat           ar          new
## 10917.674441 31995.252547   392.127453     1.809055 64982.505664
```

```r
X.gpa.2<-model.matrix(gpa.lm.2)
kappa(X.gpa.2)
```

```
## [1] 10980.34
```

Longley dataset

This is a classic infamous dataset, designed to reveal computational weaknesses in statistical software (of its time, fifty years ago). The source is: Longley, James W. (1967), *"An Appraisal of Least Squares Programs for the Electronic Computer from the Point of View of the User"*, Journal of the American Statistical Association, Vol. 62, No. 319, pp. 819-841.

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

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

```r
# ?longley
```

Exercise

Prepare a linear model to predict `Employed` from the remaining variables.

Obtain VIF's and condition number. Repeat after standardizing the predictors. What happens with VIF and `kappa`?

```r
#
# Insert here your code
#
```

2. Polynomial regression

The cars dataset

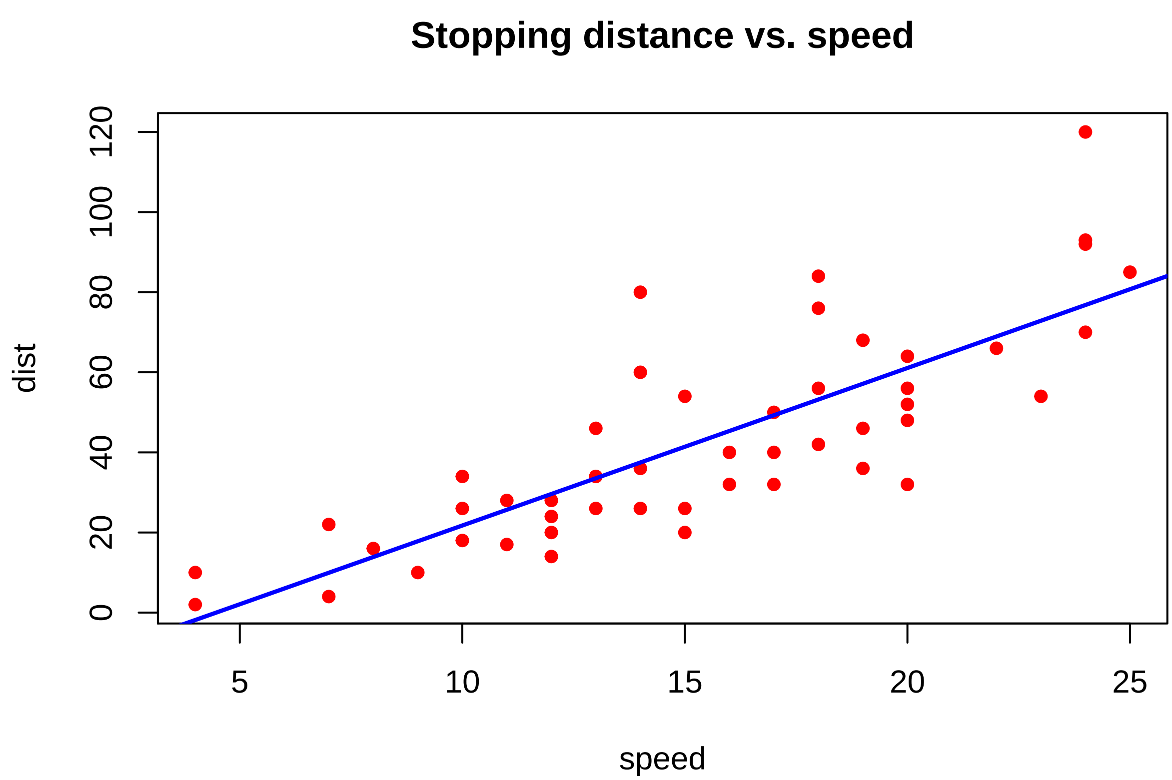
We take another classic dataset: `cars`, stopping distance of several vehicles as a function of the initial speed. These data were registered in the 1920's. From Physics, stopping distance should be a quadratic function of speed. Firstly we fit a simple linear regression and then we try a linear regression with a quadratic term.

Simple linear model

```
data(cars)
# ?cars
str(cars)
```

```
## 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

```
plot(dist~speed,data=cars,pch=19,col="red",cex=0.8,main="Stopping distance vs. speed")
cars.lm.1<-lm(dist~speed,data=cars)
abline(cars.lm.1,lwd=2.1,col="blue")
```




```
summary(cars.lm.1)
```

```
##
## Call:
## lm(formula = dist ~ speed, data = cars)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -29.069  -9.525  -2.272   9.215  43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.38 on 48 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6438
## F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

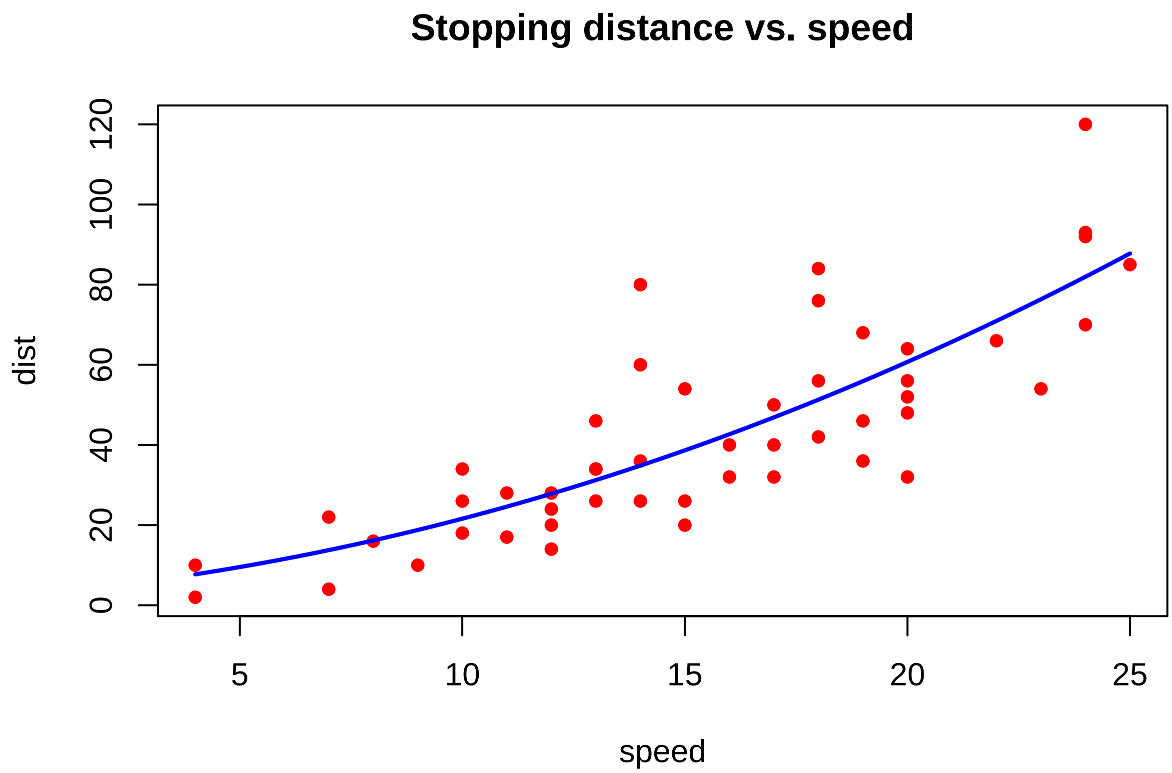
Linear regression with a quadratic term

Note the model syntax!

```r
cars.lm.2<-lm(dist~speed+I(speed^2),data=cars)
summary(cars.lm.2)
```

```
##
## Call:
## lm(formula = dist ~ speed + I(speed^2), data = cars)
##
## 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
## speed        0.91329    2.03422   0.449    0.656
## I(speed^2)   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
```

```r
plot(dist~speed,data=cars,pch=19,col="red",cex=0.8,main="Stopping distance vs. speed")
s<-seq(min(cars$speed),max(cars$speed),length=400)
d<-predict(cars.lm.2,newdata=data.frame(speed=s))
lines(s,d,lwd=2.1,col="blue")
```



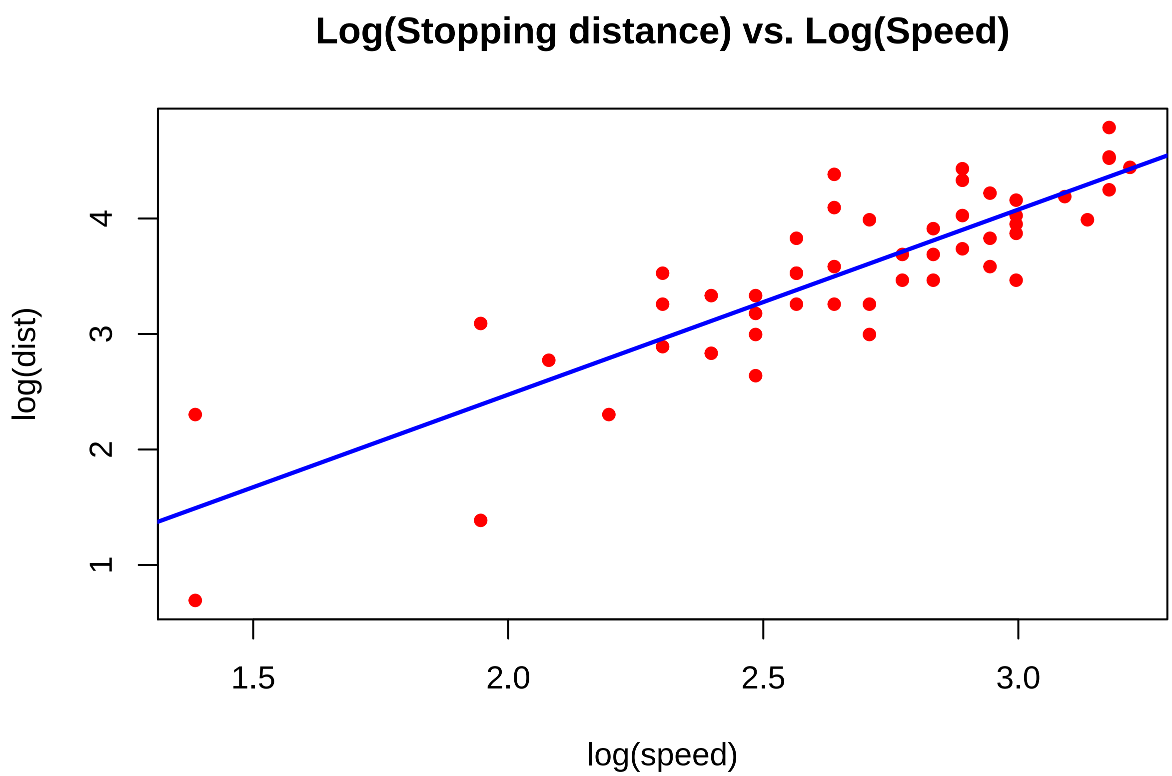
Linear model with the logarithms of both predictor and response

This is an expedient device when one expects a power function relationship.

```
cars.lm.3<- lm(log(dist) ~ log(speed), data = cars)
summary(cars.lm.3)
```

```
##
## Call:
## lm(formula = log(dist) ~ log(speed), data = cars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00215 -0.24578 -0.02898  0.20717  0.88289
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.7297     0.3758  -1.941   0.0581 .
## log(speed)    1.6024     0.1395  11.484 2.26e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4053 on 48 degrees of freedom
## Multiple R-squared:  0.7331, Adjusted R-squared:  0.7276
## F-statistic: 131.9 on 1 and 48 DF,  p-value: 2.259e-15
```

```r
plot(log(dist)~log(speed),data=cars,pch=19,col="red",cex=0.8,main="Log(Stopping distance) vs. Log(Speed
abline(cars.lm.3,lwd=2.1,col="blue")
```



3. Comparing regression models

`hills` dataset

Package DAAG

From the book: Maindonald, J., Braun, J. (2003), *Data Analysis and Graphics Using R.* Cambridge University Press.

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

```
## Loading required package: DAAG

## Loading required package: lattice

##
## Attaching package: 'DAAG'

## The following object is masked from 'package:car':
##
##     vif
```

Load the `hills` dataset and read its help:

```r
data(hills)
str(hills)
```

```
## 'data.frame':    35 obs. of  3 variables:
##  $ dist : num  2.4 6 6 7.5 8 8 16 6 5 6 ...
##  $ climb: num  650 2500 900 800 3070 ...
##  $ time : num  0.268 0.806 0.561 0.76 1.038 ...
```

```
#?hills
```

Intuition about the physics underlying this problem suggests a power function relationship which, in turn, motivates us to apply a logarithmic transformation.

We prepare a new `data.frame` with the logarithms.

```
loghills<-log(hills)
names(loghills)<-c("logdist","logclimb","logtime")
```

Does the transformation improve the linear regression? Compare both settings, scatterplots, correlation matrices.

```
#
# Insert here your code
#
```

Comparing regressions: relevant quantities

To this end we have the following available quantities:

1. ResSS, the Residual Sum of Squares
2. The coefficient of determination R^2
3. The F statistic.
4. The AIC statistic (Akaike Information Criterion).

ResSS is a goodness-of-fit measure: smaller values indicate a better fit. However, when a model L_1 is obtained by adding predictors to an initial one L_0, inevitably ResSS decreases ($\mathrm{ResSS}_1 \leq \mathrm{ResSS}_0$) by geometry, since in the L_1 we are projecting on a subspace larger than in L_0. Similarly $R_1^2 \geq R_0^2$.

From a statistical perspective we need to decide whether an observed decrease in ResSS, or increase in R^2 is significant. This is the purpose of F and AIC.

When we get a new model L_1 by adding a single predictor to a model L_0, the comparison F statistic is defined as the relative decrease in ResSS, that is, the ratio:

```
# F01<-(ResSS0-ResSS1)/(ResSS1/df1)
```

`ResSS0` and `ResSS` are, respectively, the residual sum of squares of L_0 and L_1, and `df1` is the number of degrees of freedom of L_1. `F01` is a test statistic to decide between the null hypothesis:

$$H_0 = \text{“The new predictor does not improve the model"}.$$

versus the alternative that it does. When the models are Gauss-Markov normal, `F01` follows a Fisher-Snedecor F distribution with 1 and `df1` degrees of freedom.

AIC (Akaike Information Criterion) is an answer to a different problem: adding a predictor to a prediction model means we are adding a parameter to the prediction function (a linear function in this case) and, in general, increasing the number of parameters in a prediction function learned from a given training dataset tends to fit better this particular dataset at the cost of deteriorating the quality of predictions for other samples *from the same statistical population* from which the training sample was extracted.

This is another instance of the bias/variance tradeoff.

AIC is a *penalized (minus)log-likelihood,* the difference:

$$-\log \mathcal{L} + \lambda\,\mathcal{P}, \qquad \lambda > \prime,$$

where an increase in log-likelihood gained by adding a new parameter can be cancelled by an increase in the penalization term $\mathcal{P}$. In this way we can tell the better from two *nested* models (a pair of models where the set of predictors in one of them is a subset of that in the other).

The better model is the one with a smaller AIC (tending to a larger likelihood). In R the AIC of a model L can be obtained with:

```
# extractAIC(L)
```

Which uses the definition:

```
# AIC = - 2*log L + k * edf,
```

In particular, for a linear model, (DAAG pag. 153):

```
# AIC= n*log(ResSS/n)+2*Regdf
```

For the `hills` dataset:

```
loghills.l0<-lm(logtime~logdist,data=loghills)
loghills.l1<-lm(logtime~logdist+logclimb,data=loghills)
```

The following function extracts a list of relevant quantities from an object `L` of the `lm` class (the ones from last session, plus AIC).

```
Regression.Quantities<-function(L){
    y<-L$model[[1]]
    y0<-y-mean(y)
    TotalSS0<-sum(y0^2)
    n<-length(y)
    Totaldf<-n
    Totaldf0<-Totaldf-1
#
    yhat<-as.numeric(L$fitted.values)
    yhat0<-yhat-mean(yhat)
    RegSS0<-sum(yhat0^2)
    Regdf<-L$rank
    Regdf0<-Regdf-1
#
    ytilde<-as.numeric(L$residuals)
    ResSS<-sum(ytilde^2)
    Resdf<-Totaldf0-Regdf0
#
    TotalMeanS0<-TotalSS0/Totaldf0
    RegMeanS0<-RegSS0/Regdf0
    ResMeanS<-ResSS/Resdf
#
    R2<-RegSS0/TotalSS0
    RegF<-RegMeanS0/ResMeanS
    AIC=n*log(ResSS/n)+2*Regdf
#
    Q<-list(
        y=y,y0=y0,n=n,TotalSS0=TotalSS0,Totaldf=Totaldf,Totaldf0=Totaldf0,
        yhat=yhat,yhat0=yhat0,RegSS0=RegSS0,Regdf=Regdf,Regdf0=Regdf0,
        ytilde=ytilde,ResSS=ResSS,Resdf=Resdf,
```

```
        TotalMeanS0=TotalMeanS0,RegMeanS0=RegMeanS0,ResMeanS=ResMeanS,
        R2=R2,RegF=RegF,AIC=AIC)
    return(Q)
    }
```

Using `Regression.Quantities()`:

```
loghills.Q0<-Regression.Quantities(loghills.l0)
loghills.Q1<-Regression.Quantities(loghills.l1)
```

From these results, decide which is the better model, according to the above criteria.

```
#
# Insert your code here
#
```

The `stats` package has two functions, `add1()` and `drop1`, allowing us to perform this operation in a semi-automatic way. Either we add a predictor to the basic model:

```
add1(loghills.l0,"logclimb")
```

```
## Single term additions
##
## Model:
## logtime ~ logdist
##          Df Sum of Sq    RSS     AIC
## <none>                3.5055 -76.535
## logclimb  1    0.3355 3.1700 -78.056
```

```
add1(loghills.l0,"logclimb",test="F")
```

```
## Single term additions
##
## Model:
## logtime ~ logdist
##          Df Sum of Sq    RSS     AIC F value  Pr(>F)
## <none>                3.5055 -76.535
## logclimb  1    0.3355 3.1700 -78.056  3.3867 0.07501 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
add1(loghills.l0,"logclimb",test="Chisq")
```

```
## Single term additions
##
## Model:
## logtime ~ logdist
##          Df Sum of Sq    RSS     AIC Pr(>Chi)
## <none>                3.5055 -76.535
## logclimb  1    0.3355 3.1700 -78.056  0.06059 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Or drop a variable from a larger model:

```
drop1(loghills.l1,"logclimb")
```

```
## Single term deletions
##
```

```
## Model:
## logtime ~ logdist + logclimb
##          Df Sum of Sq    RSS     AIC
## <none>                3.1700 -78.056
## logclimb  1    0.3355 3.5055 -76.535
```

```r
drop1(loghills.l1,"logclimb",test="F")
```

```
## Single term deletions
##
## Model:
## logtime ~ logdist + logclimb
##          Df Sum of Sq    RSS     AIC F value  Pr(>F)
## <none>                3.1700 -78.056
## logclimb  1    0.3355 3.5055 -76.535  3.3867 0.07501 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
drop1(loghills.l1,"logdist")
```

```
## Single term deletions
##
## Model:
## logtime ~ logdist + logclimb
##         Df Sum of Sq    RSS     AIC
## <none>               3.1700 -78.056
## logdist  1    4.8786 8.0486 -47.445
```

```r
drop1(loghills.l1,"logdist",test="F")
```

```
## Single term deletions
##
## Model:
## logtime ~ logdist + logclimb
##         Df Sum of Sq    RSS     AIC F value    Pr(>F)
## <none>               3.1700 -78.056
## logdist  1    4.8786 8.0486 -47.445  49.247 5.918e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Steam dataset

Files: `Steam.dat.txt` (Data description), `Steam1.txt` (Actual data).

```r
Steam<-read.table("Steam1.txt",header=TRUE)
str(Steam)
```

```
## 'data.frame':    25 obs. of  10 variables:
##  $ y : num  10.98 11.13 12.51 8.4 9.27 ...
##  $ x1: num  5.2 5.12 6.19 3.89 6.28 5.76 3.45 6.57 5.69 6.14 ...
##  $ x2: num  0.61 0.64 0.78 0.49 0.84 0.74 0.42 0.87 0.75 0.76 ...
##  $ x3: num  7.4 8 7.4 7.5 5.5 8.9 4.1 4.1 4.1 4.5 ...
##  $ x4: num  31 29 31 30 31 30 31 31 30 31 ...
##  $ x5: num  20 20 23 20 21 22 11 23 21 20 ...
##  $ x6: num  22 25 17 22 0 0 0 0 0 0 ...
##  $ x7: num  35.3 29.7 30.8 58.8 61.4 71.3 74.4 76.7 70.7 57.5 ...
##  $ x8: num  54.8 64 54.8 56.3 30.3 79.2 16.8 16.8 16.8 20.3 ...
```

```
##  $ x9: num  4 5 4 4 5 4 2 5 4 5 ...
```

Exercise:

With the `Steam` dataset we try to find the best subset of predictors of `y` from `(x1,x2,x3,x4,x5,x6,x7,x8,x9)`:

With such data as `Steam`, with p predictors, there is a large number of possible regressions –precisely 2^p, the number of subsets of the set of all predictors- which, in principle, they can be tested by repeatedly applying the above functions, either starting with the base model with no predictors other than intercept:

```
ls0<-lm(y~1, data=Steam)
summary(ls0)
```

```
##
## Call:
## lm(formula = y ~ 1, data = Steam)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.064 -1.024 -0.284  1.516  3.086
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.4240     0.3261    28.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.631 on 24 degrees of freedom
```

and successively add predictors or, on the contrary, starting with the full model:

```
ls1<-lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9, data=Steam)
summary(ls1)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9,
##     data = Steam)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0726 -0.3670  0.0758  0.2942  1.0436
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.227769   7.833663  -0.029  0.97719
## x1           0.197252   0.635859   0.310  0.76067
## x2           5.763660   4.466510   1.290  0.21644
## x3           1.985020   0.861398   2.304  0.03592 *
## x4           0.132288   0.231817   0.571  0.57668
## x5          -0.046401   0.060859  -0.762  0.45762
## x6          -0.005743   0.027373  -0.210  0.83664
## x7          -0.069204   0.018456  -3.750  0.00193 **
## x8          -0.139526   0.060742  -2.297  0.03643 *
## x9          -0.318685   0.239128  -1.333  0.20252
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.644 on 15 degrees of freedom
## Multiple R-squared:  0.9025, Adjusted R-squared:  0.844
## F-statistic: 15.43 on 9 and 15 DF,  p-value: 4.667e-06
```

And then successively remove them.

For instance, we start with `ls0`:

```r
add1(ls0,"x1",test="F")
```

```
## Single term additions
##
## Model:
## y ~ 1
##        Df Sum of Sq    RSS    AIC F value  Pr(>F)
## <none>              63.816 25.428
## x1      1    9.3698 54.446 23.458  3.9581 0.05867 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# ...
add1(ls0,"x9",test="F")
```

```
## Single term additions
##
## Model:
## y ~ 1
##        Df Sum of Sq    RSS    AIC F value  Pr(>F)
## <none>              63.816 25.428
## x9      1    9.3183 54.497 23.482  3.9327 0.05942 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see that `x7` is the better addition.

We keep this variable and now we try to add a second one following the same procedure, and successively, until we find a satisfactory subset.

This is the *forward* variables selection method.

Symmetrically, the *backward* method starts with the full model `ls1` and sequentially removes variables.

It is worth noting that if, for instance, at a certain stage of a *forward* procedure the current subset of variables is A, we have not tested all subsets of A but only those in the path through which we arrived at A. To test them we should apply *backward* selection from A and so on.

The `stats` package has a `step()` function to automate this process:

```r
step(ls0,"~x1+x2+x3+x4+x5+x6+x7+x8+x9")
```

```
## Start:  AIC=25.43
## y ~ 1
##
##        Df Sum of Sq    RSS     AIC
## + x7    1    45.592 18.223 -3.9042
## + x6    1    26.192 37.624 14.2192
## + x3    1    14.357 49.459 21.0568
## + x5    1    10.739 53.077 22.8216
## + x8    1     9.934 53.882 23.1981
```

```
## + x1    1     9.370 54.446 23.4584
## + x9    1     9.318 54.497 23.4820
## + x2    1     5.958 57.858 24.9779
## <none>              63.816 25.4281
## + x4    1     1.193 62.623 26.9563
##
## Step:  AIC=-3.9
## y ~ x7
##
##        Df Sum of Sq    RSS      AIC
## + x1    1     9.292  8.931 -19.7327
## + x2    1     8.438  9.785 -17.4508
## + x5    1     3.140 15.084  -6.6317
## + x4    1     2.623 15.600  -5.7902
## + x9    1     2.236 15.988  -5.1763
## + x6    1     1.712 16.511  -4.3711
## <none>              18.223  -3.9042
## + x8    1     0.359 17.864  -2.4022
## + x3    1     0.224 17.999  -2.2134
## - x7    1    45.592 63.816  25.4281
##
## Step:  AIC=-19.73
## y ~ x7 + x1
##
##        Df Sum of Sq    RSS      AIC
## <none>               8.931 -19.7327
## + x9    1     0.319  8.612 -18.6433
## + x4    1     0.238  8.693 -18.4090
## + x6    1     0.032  8.899 -17.8228
## + x8    1     0.015  8.917 -17.7738
## + x2    1     0.004  8.927 -17.7449
## + x5    1     0.003  8.929 -17.7404
## + x3    1     0.000  8.931 -17.7338
## - x1    1     9.292 18.223  -3.9042
## - x7    1    45.515 54.446  23.4584
```

```
##
## Call:
## lm(formula = y ~ x7 + x1, data = Steam)
##
## Coefficients:
## (Intercept)           x7           x1
##     9.47422     -0.07976      0.76165
```

Also from the full model:

```r
step(ls1)
```

```
## Start:  AIC=-14.77
## y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS      AIC
## - x6    1    0.0183 6.2397 -16.6986
## - x1    1    0.0399 6.2614 -16.6120
## - x4    1    0.1351 6.3565 -16.2349
## - x5    1    0.2411 6.4625 -15.8213
```

```
## <none>               6.2214 -14.7719
## - x2    1    0.6907  6.9121 -14.1401
## - x9    1    0.7366  6.9581 -13.9743
## - x8    1    2.1884  8.4098  -9.2369
## - x3    1    2.2025  8.4240  -9.1949
## - x7    1    5.8314 12.0529  -0.2394
##
## Step:  AIC=-16.7
## y ~ x1 + x2 + x3 + x4 + x5 + x7 + x8 + x9
##
##        Df Sum of Sq     RSS      AIC
## - x1    1    0.0462  6.2859 -18.5142
## - x4    1    0.1455  6.3852 -18.1223
## - x5    1    0.2587  6.4984 -17.6829
## <none>               6.2397 -16.6986
## - x2    1    0.6959  6.9356 -16.0551
## - x9    1    0.7437  6.9834 -15.8837
## - x8    1    2.1822  8.4219 -11.2011
## - x3    1    2.1981  8.4378 -11.1539
## - x7    1   11.7123 17.9520   7.7207
##
## Step:  AIC=-18.51
## y ~ x2 + x3 + x4 + x5 + x7 + x8 + x9
##
##        Df Sum of Sq     RSS      AIC
## - x4    1    0.3000  6.5859 -19.3485
## - x5    1    0.3598  6.6457 -19.1228
## <none>               6.2859 -18.5142
## - x9    1    0.7015  6.9874 -17.8692
## - x8    1    2.5254  8.8113 -12.0710
## - x3    1    2.5653  8.8512 -11.9581
## - x2    1    4.7390 11.0249  -6.4679
## - x7    1   11.6798 17.9657   5.7397
##
## Step:  AIC=-19.35
## y ~ x2 + x3 + x5 + x7 + x8 + x9
##
##        Df Sum of Sq     RSS      AIC
## <none>               6.5859 -19.3485
## - x5    1    0.6317  7.2176 -19.0587
## - x9    1    1.1531  7.7391 -17.3149
## - x3    1    2.8405  9.4265 -12.3839
## - x8    1    2.9010  9.4869 -12.2240
## - x2    1    7.2384 13.8243  -2.8112
## - x7    1   12.5365 19.1225   5.2997

##
## Call:
## lm(formula = y ~ x2 + x3 + x5 + x7 + x8 + x9, data = Steam)
##
## Coefficients:
## (Intercept)           x2           x3           x5           x7
##     3.47721      7.88356      2.15116     -0.06817     -0.06783
##          x8           x9
```

```
##    -0.15287     -0.37100
```

Reasonably enough, this combined *forward* and *backward* procedure is called *stepwise.*

The MASS package has analogous functions, `addterm()`, `dropterm()`, and `stepAIC()`, with the same functionality as the above functions in the basic `stats` package.

Finally, there is the strategy of computing all the possible 2^p regressions, with all the subsets in the set of predictors, from $\emptyset$ to the total number p of predictors, and then comparing all the resulting AIC values, keeping the best one.

This is the *All Subsets Regression* method. Obviously, this computation is unfeasible by a brute force approach, except for models with a very small number of predictors. There are combinatorial *branch and bound* algorithms allowing us to limit the number of actual regressions we need to evaluate in full. The method depends on imposing a partial ordering on the set of all possible regressions, giving it a tree-like structure, in which we will need to evaluate only a small number of nodes. The `leaps` package implements this procedure (the name originates from the idiom *leaps and bounds).*

Homework 1 (part 1)

This is the first half. The second half will appear at the end of the next laboratory (Regression 03 - Regularization). The aim here is to apply linear regression on the `Home prices` dataset from Kaggle House Prices: Advanced Regression Techniques. As explained in the documentation there, this dataset is a test bed for many prediction methods, thus it is not to be expected that linear regression will be particularly succesful. Later on in the course we will try other methods.

As a warmup, take the similar but smaller `Boston home prices` dataset. This is a very classic example and it is easy to find its study with linear regression in many online resources. Googling "Boston home prices" will yield many implementations, both in R and in python.

You can learn from and even reproduce parts of these sources. This is acceptable provided that you:

1. Give credit where it is due, in particular, including full reference (URL) of any cited work
2. Do not copy/paste "in extenso" large chunks of code
3. Understand everything you write, explaining with sufficient details the steps you take and the results you obtain.

The Boston home prices data are intended for prediction of home values in suburbs of the namesake city. `medv` is the response variable and the other variables are predictors. See their description in the dataset help. Some of these predictors are intrinsic habitational characteristics, such as number of bedrooms; other predictors have a socioeconomic nature and, finally, other predictors are geographical or environmental.

In R you find this dataset in the MASS package, from the book by W. N. Venables and B. D. Ripley (2002), *Modern Applied Statistics with S:*

```r
require(MASS)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked _by_ '.GlobalEnv':
##
##     hills

## The following object is masked from 'package:DAAG':
##
##     hills
```

```r
data(Boston)
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Guidelines

Perform a statistical description of the data. In the first place individually, summarizing each variable, both graphically, e.g. with boxplot and histogram, and numerically. Do these variables have a normal appearance? Or, rather, do these variables show an asymmetric shape? Check correlations between pairs of predictors and between individual predictors and response. It will be useful to truncate to 2 or 1 decimal places, to avoid clutter:

In this way we can see at a glance which correlations are large or small. Is there a danger of multicollinearity?

Fit a linear regression model of `medv` on the remaining variables. From the model summary, can we state the model fits well? which variables appear to be better or worse predictors?

Prepare an optimal model with the better predictors. The `ResSS` of this model is much larger than the one from the full model? Note that this model still has a non significant predictor. Discard it.

Fit another linear regression model with response `log(medv)` on the remaining predictors or with their logarithms. Which one is better?

Back to the Kaggle ***House prices*** dataset

Redo the operations above with the new dataset, replacing the response `medv` with `SalePrice`.

```r
HP.train<-read.csv("House.prices.train.csv")
str(HP.train)
```

```
## 'data.frame':    1460 obs. of  81 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : Factor w/ 5 levels "C (all)","FV",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 4 1 1 1 1 4 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "AllPub","NoSeWa": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
```

```
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 8 levels "ClyTile","CompShg",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 2 3 2 3 2 3 4 4 3 3 ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 4 3 4 3 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 3 3 4 3 3 1 3 4 4 ...
##  $ BsmtCond     : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 2 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "Av","Gd","Mn",..: 4 2 3 4 1 4 1 3 4 4 ...
##  $ BsmtFinType1 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 3 1 3 1 3 3 3 1 6 3 ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 6 6 6 6 6 6 6 2 6 6 ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 1 1 1 3 1 1 1 1 3 1 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 5 levels "FuseA","FuseF",..: 5 5 5 5 5 5 5 5 2 5 ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 3 3 4 3 4 4 4 ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 3 7 ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : Factor w/ 5 levels "Ex","Fa","Gd",..: NA 5 5 3 5 NA 3 5 5 5 ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 6 2 2 2 2 6 2 ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : Factor w/ 3 levels "Fin","RFn","Unf": 2 2 2 3 2 3 2 2 3 2 ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 2 3 ...
##  $ GarageCond   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
```

```
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 3 levels "Ex","Fa","Gd": NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence        : Factor w/ 4 levels "GdPrv","GdWo",..: NA NA NA NA NA 3 NA NA NA NA ...
##  $ MiscFeature  : Factor w/ 4 levels "Gar2","Othr",..: NA NA NA NA NA 3 NA 3 NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

```r
HP.test<-read.csv("House.prices.test.csv")
str(HP.test)
```

```
## 'data.frame':    1459 obs. of  80 variables:
##  $ Id           : int  1461 1462 1463 1464 1465 1466 1467 1468 1469 1470 ...
##  $ MSSubClass   : int  20 20 60 60 120 60 20 60 20 20 ...
##  $ MSZoning     : Factor w/ 5 levels "C (all)","FV",..: 3 4 4 4 4 4 4 4 4 4 ...
##  $ LotFrontage  : int  80 81 74 78 43 75 NA 63 85 70 ...
##  $ LotArea      : int  11622 14267 13830 9978 5005 10000 7980 8402 10176 8400 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 1 1 1 1 1 1 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 2 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 1 level "AllPub": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 1 5 5 5 1 5 5 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 13 13 9 9 22 9 9 9 9 13 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 2 3 3 3 3 3 3 3 3 3 ...
##  $ Condition2   : Factor w/ 5 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 5 1 1 1 1 1 ...
##  $ HouseStyle   : Factor w/ 7 levels "1.5Fin","1.5Unf",..: 3 3 5 5 3 5 3 5 3 3 ...
##  $ OverallQual  : int  5 6 5 6 8 6 6 6 7 4 ...
##  $ OverallCond  : int  6 6 5 6 5 5 7 5 5 5 ...
##  $ YearBuilt    : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
##  $ YearRemodAdd : int  1961 1958 1998 1998 1992 1994 2007 1998 1990 1970 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 4 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 4 levels "CompShg","Tar&Grv",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Exterior1st  : Factor w/ 13 levels "AsbShng","AsphShn",..: 11 12 11 11 7 7 7 11 7 9 ...
##  $ Exterior2nd  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 14 13 13 7 7 7 13 7 10 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 3 2 3 2 3 3 3 3 3 3 ...
##  $ MasVnrArea   : int  0 108 0 20 0 0 0 0 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 4 4 4 3 4 4 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 3 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 2 2 3 3 3 3 3 3 3 2 ...
##  $ BsmtQual     : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 4 3 4 3 3 3 3 3 4 ...
##  $ BsmtCond     : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "Av","Gd","Mn",..: 4 4 4 4 4 4 4 4 2 4 ...
##  $ BsmtFinType1 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 5 1 3 3 1 6 1 6 3 1 ...
```

```
##  $ BsmtFinSF1   : int  468 923 791 602 263 0 935 0 637 804 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 4 6 6 6 6 6 6 6 6 5 ...
##  $ BsmtFinSF2   : int  144 0 0 0 0 0 0 0 0 78 ...
##  $ BsmtUnfSF    : int  270 406 137 324 1017 763 233 789 663 0 ...
##  $ TotalBsmtSF  : int  882 1329 928 926 1280 763 1168 789 1300 882 ...
##  $ Heating      : Factor w/ 4 levels "GasA","GasW",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 3 1 1 3 1 3 3 5 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 4 levels "FuseA","FuseF",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ X1stFlrSF    : int  896 1329 928 926 1280 763 1187 789 1341 882 ...
##  $ X2ndFlrSF    : int  0 0 701 678 0 892 0 676 0 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  896 1329 1629 1604 1280 1655 1187 1465 1341 882 ...
##  $ BsmtFullBath : int  0 0 0 0 0 0 1 0 1 1 ...
##  $ BsmtHalfBath : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  1 1 2 2 2 2 2 2 1 1 ...
##  $ HalfBath     : int  0 1 1 1 0 1 0 1 1 0 ...
##  $ BedroomAbvGr : int  2 3 3 3 2 3 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 3 4 3 3 4 4 4 3 4 ...
##  $ TotRmsAbvGrd : int  5 6 6 7 5 7 6 7 5 4 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ Fireplaces   : int  0 0 1 1 0 1 0 1 1 0 ...
##  $ FireplaceQu  : Factor w/ 5 levels "Ex","Fa","Gd",..: NA NA 5 3 NA 5 NA 3 4 NA ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ GarageYrBlt  : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
##  $ GarageFinish : Factor w/ 3 levels "Fin","RFn","Unf": 3 3 1 1 2 1 1 1 3 1 ...
##  $ GarageCars   : int  1 1 2 2 2 2 2 2 2 2 ...
##  $ GarageArea   : int  730 312 482 470 506 440 420 393 506 525 ...
##  $ GarageQual   : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ GarageCond   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  140 393 212 360 0 157 483 0 192 240 ...
##  $ OpenPorchSF  : int  0 36 34 36 82 84 21 75 0 0 ...
##  $ EnclosedPorch: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ScreenPorch  : int  120 0 0 0 144 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 2 levels "Ex","Gd": NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence        : Factor w/ 4 levels "GdPrv","GdWo",..: 3 NA 3 NA NA NA 1 NA NA 3 ...
##  $ MiscFeature  : Factor w/ 3 levels "Gar2","Othr",..: NA 1 NA NA NA NA 3 NA NA NA ...
##  $ MiscVal      : int  0 12500 0 0 0 0 500 0 0 0 ...
##  $ MoSold       : int  6 6 3 6 1 4 3 5 2 4 ...
##  $ YrSold       : int  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 5 5 5 5 5 5 5 ...
```

Remarks

1. This dataset is divided in two halves, `train` and `test`. The first one is to train the prediction algorithm (in the `lm` case, compute the regression coefficients) which then is assessed with the second one. This is a usual procedurein Machine Learning, to avoid the optimistic bias due to using the same data for both functions. Note that here in the testing half the response is missing. We are to predict it!
2. The first column in both train and test files is an `Id` case number. Make sure you remove it before any

computation. It may be kept aside for post-processing steps, e.g., visualization.

```
HP.train<-HP.train[,-1]
str(HP.train)
```

```
## 'data.frame':    1460 obs. of  80 variables:
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : Factor w/ 5 levels "C (all)","FV",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ LotFrontage  : int  65 80 68 60 84 85 75 NA 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 4 1 1 1 1 4 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "AllPub","NoSeWa": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 8 levels "ClyTile","CompShg",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 2 3 2 3 2 3 4 4 3 3 ...
##  $ MasVnrArea   : int  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 4 3 4 3 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 3 3 4 3 3 1 3 4 4 ...
##  $ BsmtCond     : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 2 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "Av","Gd","Mn",..: 4 2 3 4 1 4 1 3 4 4 ...
##  $ BsmtFinType1 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 3 1 3 1 3 3 3 1 6 3 ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 6 6 6 6 6 6 6 2 6 6 ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 1 1 1 3 1 1 1 1 3 1 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 5 levels "FuseA","FuseF",..: 5 5 5 5 5 5 5 5 2 5 ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
```

```
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 3 3 4 3 4 4 4 ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 3 7 ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : Factor w/ 5 levels "Ex","Fa","Gd",..: NA 5 5 3 5 NA 3 5 5 5 ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 6 2 2 2 2 6 2 ...
##  $ GarageYrBlt  : int  2003 1976 2001 1998 2000 1993 2004 1973 1931 1939 ...
##  $ GarageFinish : Factor w/ 3 levels "Fin","RFn","Unf": 2 2 2 3 2 3 2 2 3 2 ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 2 3 ...
##  $ GarageCond   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 3 levels "Ex","Fa","Gd": NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence        : Factor w/ 4 levels "GdPrv","GdWo",..: NA NA NA NA NA 3 NA NA NA NA ...
##  $ MiscFeature  : Factor w/ 4 levels "Gar2","Othr",..: NA NA NA NA NA 3 NA 3 NA NA ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
```

```
HP.test<-HP.test[,-1]
str(HP.test)
```

```
## 'data.frame':    1459 obs. of  79 variables:
##  $ MSSubClass   : int  20 20 60 60 120 60 20 60 20 20 ...
##  $ MSZoning     : Factor w/ 5 levels "C (all)","FV",..: 3 4 4 4 4 4 4 4 4 4 ...
##  $ LotFrontage  : int  80 81 74 78 43 75 NA 63 85 70 ...
##  $ LotArea      : int  11622 14267 13830 9978 5005 10000 7980 8402 10176 8400 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 2 levels "Grvl","Pave": NA NA NA NA NA NA NA NA NA NA ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 1 1 1 1 1 1 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 2 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 1 level "AllPub": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 1 5 5 5 1 5 5 5 1 ...
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 13 13 9 9 22 9 9 9 9 13 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 2 3 3 3 3 3 3 3 3 3 ...
##  $ Condition2   : Factor w/ 5 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 5 1 1 1 1 1 ...
##  $ HouseStyle   : Factor w/ 7 levels "1.5Fin","1.5Unf",..: 3 3 5 5 3 5 3 5 3 3 ...
##  $ OverallQual  : int  5 6 5 6 8 6 6 6 7 4 ...
##  $ OverallCond  : int  6 6 5 6 5 5 7 5 5 5 ...
##  $ YearBuilt    : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
```

```
##  $ YearRemodAdd : int  1961 1958 1998 1998 1992 1994 2007 1998 1990 1970 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 4 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 4 levels "CompShg","Tar&Grv",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Exterior1st  : Factor w/ 13 levels "AsbShng","AsphShn",..: 11 12 11 11 7 7 7 11 7 9 ...
##  $ Exterior2nd  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 14 13 13 7 7 7 13 7 10 ...
##  $ MasVnrType   : Factor w/ 4 levels "BrkCmn","BrkFace",..: 3 2 3 2 3 3 3 3 3 3 ...
##  $ MasVnrArea   : int  0 108 0 20 0 0 0 0 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 4 4 4 3 4 4 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 3 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 2 2 3 3 3 3 3 3 3 2 ...
##  $ BsmtQual     : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 4 3 4 3 3 3 3 3 4 ...
##  $ BsmtCond     : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ BsmtExposure : Factor w/ 4 levels "Av","Gd","Mn",..: 4 4 4 4 4 4 4 4 2 4 ...
##  $ BsmtFinType1 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 5 1 3 3 1 6 1 6 3 1 ...
##  $ BsmtFinSF1   : int  468 923 791 602 263 0 935 0 637 804 ...
##  $ BsmtFinType2 : Factor w/ 6 levels "ALQ","BLQ","GLQ",..: 4 6 6 6 6 6 6 6 6 5 ...
##  $ BsmtFinSF2   : int  144 0 0 0 0 0 0 0 0 78 ...
##  $ BsmtUnfSF    : int  270 406 137 324 1017 763 233 789 663 0 ...
##  $ TotalBsmtSF  : int  882 1329 928 926 1280 763 1168 789 1300 882 ...
##  $ Heating      : Factor w/ 4 levels "GasA","GasW",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 3 1 1 3 1 3 3 5 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 4 levels "FuseA","FuseF",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ X1stFlrSF    : int  896 1329 928 926 1280 763 1187 789 1341 882 ...
##  $ X2ndFlrSF    : int  0 0 701 678 0 892 0 676 0 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  896 1329 1629 1604 1280 1655 1187 1465 1341 882 ...
##  $ BsmtFullBath : int  0 0 0 0 0 0 1 0 1 1 ...
##  $ BsmtHalfBath : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  1 1 2 2 2 2 2 2 1 1 ...
##  $ HalfBath     : int  0 1 1 1 0 1 0 1 1 0 ...
##  $ BedroomAbvGr : int  2 3 3 3 2 3 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 4 3 4 3 3 4 4 4 3 4 ...
##  $ TotRmsAbvGrd : int  5 6 6 7 5 7 6 7 5 4 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 7 7 ...
##  $ Fireplaces   : int  0 0 1 1 0 1 0 1 1 0 ...
##  $ FireplaceQu  : Factor w/ 5 levels "Ex","Fa","Gd",..: NA NA 5 3 NA 5 NA 3 4 NA ...
##  $ GarageType   : Factor w/ 6 levels "2Types","Attchd",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ GarageYrBlt  : int  1961 1958 1997 1998 1992 1993 1992 1998 1990 1970 ...
##  $ GarageFinish : Factor w/ 3 levels "Fin","RFn","Unf": 3 3 1 1 2 1 1 1 3 1 ...
##  $ GarageCars   : int  1 1 2 2 2 2 2 2 2 2 ...
##  $ GarageArea   : int  730 312 482 470 506 440 420 393 506 525 ...
##  $ GarageQual   : Factor w/ 4 levels "Fa","Gd","Po",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ GarageCond   : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  140 393 212 360 0 157 483 0 192 240 ...
##  $ OpenPorchSF  : int  0 36 34 36 82 84 21 75 0 0 ...
##  $ EnclosedPorch: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ScreenPorch  : int  120 0 0 0 144 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 2 levels "Ex","Gd": NA NA NA NA NA NA NA NA NA NA ...
##  $ Fence        : Factor w/ 4 levels "GdPrv","GdWo",..: 3 NA 3 NA NA NA 1 NA NA 3 ...
```

```
##  $ MiscFeature  : Factor w/ 3 levels "Gar2","Othr",..: NA 1 NA NA NA NA 3 NA NA NA ...
##  $ MiscVal      : int  0 12500 0 0 0 0 500 0 0 0 ...
##  $ MoSold       : int  6 6 3 6 1 4 3 5 2 4 ...
##  $ YrSold       : int  2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 5 5 5 5 5 5 5 ...
```

3. Missing data are trouble in this dataset, something that, as a matter of fact, often happens with real data. As a first approach one could try to `na.omit()` (look up in help what this function does). Unfortunately it seems that each and every row contains some `NA` hence the result of `na.omit()` is an empty dataset. Better try to select, in a first analysis, those variables (columns) with no `NA`. The following piece of code will count the number of columns with no `NA`. I found 61 of them. Isolate these columns and perform the analysis with them.:

```
count.na<-function(x){sum(is.na(x))}
HP.train.m<-as.matrix(HP.train)
Rows.with.NA<-apply(HP.train.m,1,count.na)
table(Rows.with.NA)
```

```
## Rows.with.NA
##   1   2   3   4   5   6   7   8   9  10  11  14  15
##   2  16 132 635 501  58   5   5  32  60   7   3   4
```

```
Cols.with.NA<-apply(HP.train.m,2,count.na)
table(Cols.with.NA)
```

```
## Cols.with.NA
##    0    1    8   37   38   81  259  690 1179 1369 1406 1453
##   61    1    2    3    2    5    1    1    1    1    1    1
```