Lecture 17: Analysing and Visualizing Data II

In [1]:
```
%load_ext rmagic
%R rm()
```

In [2]:
```
%%R

NL = function() cat('\n\n')
```

This notebook topics:

In the last notebook, we focused on **visualizing population clusters and variable relationships with scatterplots**. We also described the `hclust` function that unravells **a hierchical cluster structure** in the data, and we explain how to **visualize this hierarchical structure** in terms of a **dendrogram plot**.

Today's we will focus on **linear regression techniques** to analyse **(linear) variable relationships**.

- **Simulating and visualizing clusters and relations** (reminder)
- **Computing and visualizing a regression**
- **Interpreting and predicting with a regression**

Simulating and visualizing clusters and relations (reminder)

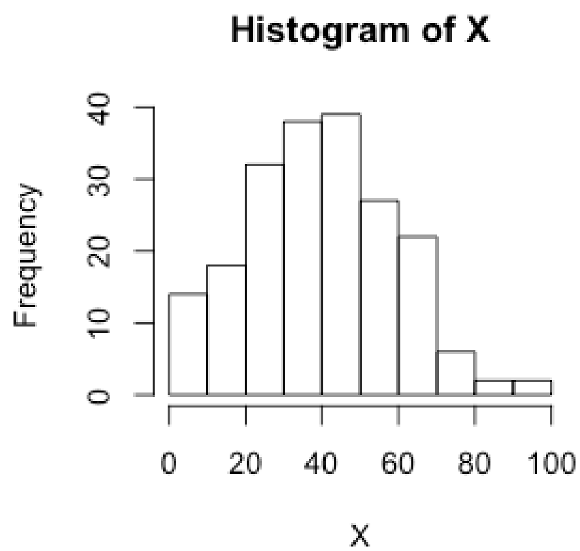
Simulating explanatory variables: X

In [3]:
```
%%R

clean.range = function(X, var.range){
        X[X < var.range[1]] = var.range[1]
        X[X > var.range[2]] = var.range[2]
        return(X)
    }
```

In [4]:
```
%%R

simulate.X = function(var.size=1, var.mean=0, var.sd=1, var.range=NULL)
{
    X = rnorm(var.size, mean=var.mean, sd=var.sd)
    if (!is.null(var.range)) X = clean.range(X, var.range)
    return(X)
}
```

In [4]:

```
%%R -r 86 -w 300 -h 300
X = simulate.X(var.size=200, var.mean=40, var.sd=20, var.range=c(0,100))
hist(X)
```



Simulating target variables: Y

In [5]:

```
%%R

simulate.Y = function(X.variables,       #
                      alpha,             #
                      beta,              #
                      residual.sd=1,     #
                      residual.mean=0,   #
                      var.range=NULL     #
                     )
    {
        X = as.matrix(X.variables)
        residuals = rnorm(dim(X)[1], mean=residual.mean, sd=residual.sd)
        e = matrix(residuals, ncol=1)
        v = matrix(beta, ncol=1)

        Y = alpha + X %*% v + e

        if(!is.null(var.range)) Y = clean.range(Y, var.range)
        return(as.numeric(Y))
    }
```

In [8]:

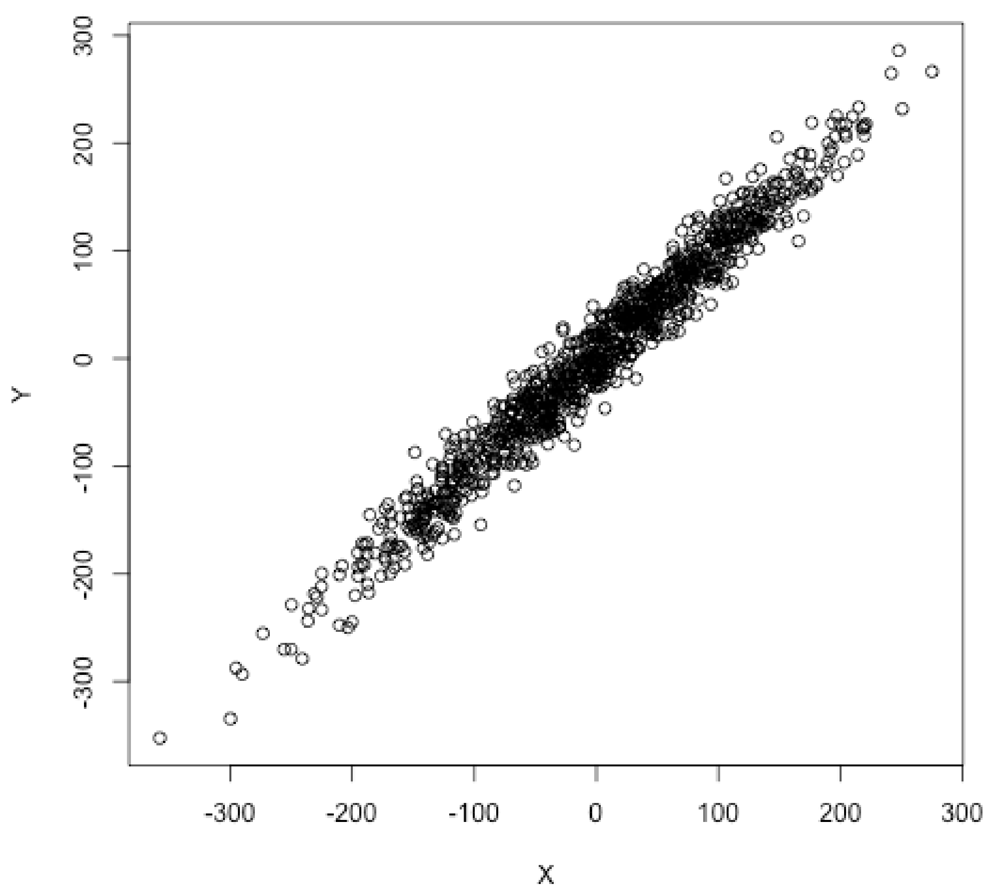
In [8]:

In [9]:

```
%%R -r 86 -w 300 -h 300

Xsize=1000; Xmean = 0; Xsd = 100;
```

```
alpha = 2
beta  = 1
error = 20

X = simulate.X(Xsize, Xmean, Xsd)
Y = simulate.Y(X, alpha, beta, error)
```

In [10]:
```
%%R

plot(X,Y)
```



Simulating clusters

In [11]:
```
%%R
#cluster informations
sizes.default   = c(30, 40, 50)
means.default   = c(-50, 100, 50)
sd.default      = c(10, 10, 10)
alphas.default  = c(20, 10, 30)
betas.default   = c(0.9, 1.2, -0.9)
errors.default  = c(10, 0, 20)
```

In [12]:
```
%%R
```

```
clusters.info = function(cl.sizes=sizes.default,      #
                         cl.means=means.default,      #
                         cl.sd=sd.default,            #
                         cl.alphas=alphas.default,    #
                         cl.betas=betas.default,      #
                         cl.errors=errors.default     #
                        )
    {
        cl.list     = list()
        cl.number   = length(cl.sizes)
        sample.size = sum(cl.sizes)

        for (i in 1:cl.number){
            cl.name = paste('cluster', i, sep='')
            cl.list[[cl.name]] = c(size=cl.sizes[i],     #
                                   mean=cl.means[i],     #
                                   sd=cl.sd[i],          #
                                   alpha=cl.alphas[i],   #
                                   beta=cl.betas[i],     #
                                   error=cl.errors[i])   #
        }
        return(t(as.data.frame(cl.list)))
    }
```

In [13]:

```
%%R

clusters.default = clusters.info()
print(clusters.default)
```

```
         size mean sd alpha beta error
cluster1   30  -50 10    20  0.9    10
cluster2   40  100 10    10  1.2     0
cluster3   50   50 10    30 -0.9    20
```

In [14]:

```
%%R

simulate.bivariateCluster = function(cl.info){
    X = simulate.X(cl.info['size'], cl.info['mean'], cl.info['sd'])
    Y = simulate.Y(X, cl.info['alpha'], cl.info['beta'], cl.info['error'])
    return(data.frame(X=X,Y=Y))
}
```

In [15]:

```
%%R

simulate.bivariateData = function(clusters.info=clusters.default)
{
    m = nrow(clusters.info)
    data = data.frame()
    for(i in 1:m){
        df = simulate.bivariateCluster(clusters.info[i,])
        df$Cluster = factor(rep(i, nrow(df)))
        data = rbind(data, df)
    }
```

```
    shuffle = sample(nrow(data))
    data = data[shuffle, ]
    return(data)
}
```

In [16]:

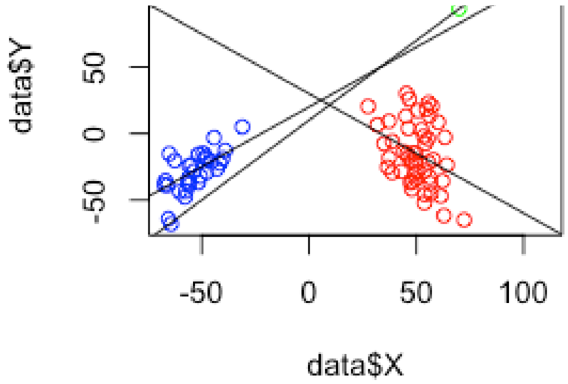
```
%%R -r 86 -w 600 -h 600

data = simulate.bivariateData(clusters.default)

print(clusters.default)
NL()
print(head(data))
```

```
         size mean sd alpha beta error
cluster1   30  -50 10    20  0.9    10
cluster2   40  100 10    10  1.2     0
cluster3   50   50 10    30 -0.9    20


            X         Y Cluster
66  95.08817 124.10580       2
73  45.72554 -15.08392       3
61 104.67994 135.61593       2
97  46.67724  26.02832       3
12 -42.88180 -26.60745       1
24 -66.89210 -35.49591       1
```

True clusters and true relations

In [17]:

```
%%R -r 86 -w 300 -h 300

mycolors = c('Blue', 'Green', 'Red')
colors   = mycolors[data$Cluster]

plot(data$X, data$Y, col=colors)

plot.trueLines = function()
{
    for (cluster in 1:3){
        intercept   = clusters.default[cluster, 'alpha']
        coefficient = clusters.default[cluster, 'beta' ]
        color       = mycolors[cluster]
        abline(intercept, coefficient, color)
    }
}

plot.trueLines()
```



In [17]:

```
```

Computing and visualizing a regression

In [57]:

```
%%R

model = lm(data$Y ~ data$X)
print(model)
```

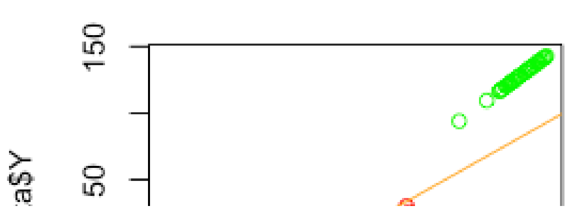
```
Call:
lm(formula = data$Y ~ data$X)

Coefficients:
(Intercept)       data$X
    -7.5572       0.9078
```

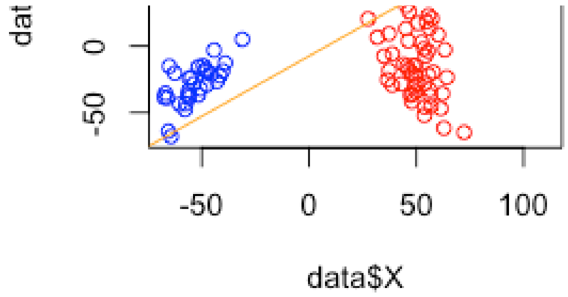
Extracting information from the model

```
abline(model)
fitted(model)
resid(model)
```

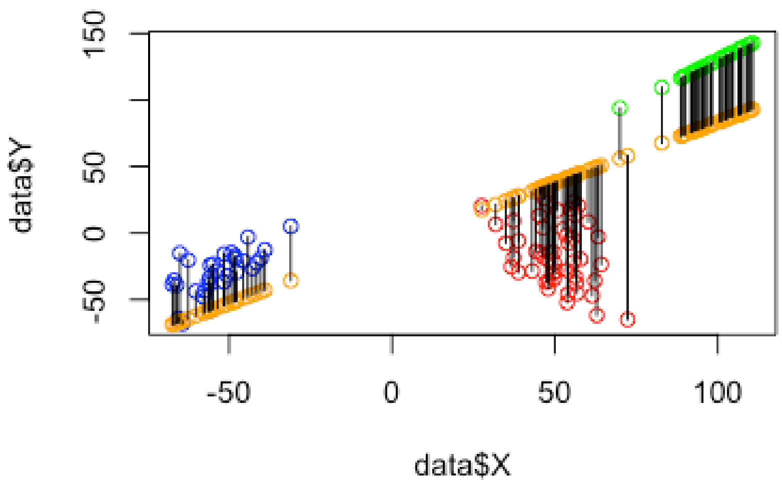
In [58]:

```
%%R -r 86 -w 300 -h 300

plot(data$X, data$Y, col=colors)

abline(model, col='Orange')
```





In [59]:
```
%%R -r 86 -w 400 -h 300

plot(data$X, data$Y, col=colors)

points(data$X, fitted(model), col='Orange')
segments(data$X, fitted(model), data$X, data$Y)
```

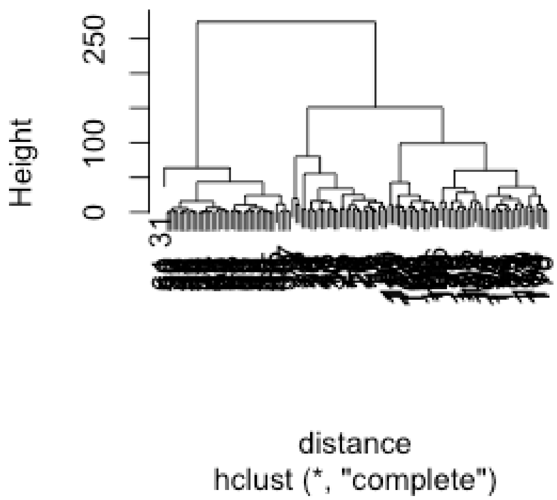


In [60]:
```
%%R

error = sum(resid(model)^2)
print(error)
```

```
[1] 292275.4
```

Retrieving the clusters without knowing them

In [22]:
```
%%R -r 86 -w 300 -h 300


distance  = dist(data[c(1,2)])
htree = hclust(distance)
plot(htree)
```



In [23]:
```
%%R

clusters = cutree(htree, 3)
print(clusters)
```

```
 66  73  61  97  12  24  89  21 113  28  39  59  60  47  82 118  75  92  2
6  41
  1   2   1   2   3   3   2   3   2   3   1   1   1   1   2   2   2   2
3   1
 96  78  90  55  14   5 112  34   2  46 117  54  42  79 103  16 109   4  8
7 106
  2   2   2   1   3   3   2   1   3   1   2   1   1   2   2   3   2   3
2   2
 72  76 110   3 115  29  74   9 101  98  23  44  38 119  20  91  56  52  6
2  93
  2   2   2   3   2   3   2   3   2   2   3   1   1   2   3   2   1   1
1   2
116  83 114  51  64  58 120  33 111  11  94   6  63  70  18   7  77  84  5
3  85
  2   2   2   1   1   1   2   1   2   3   2   3   1   1   3   3   2   2
1   2
102  19 100  40  10  22  17   1  95  36  65  15  25  48  81  69   8  43  3
2  45
  2   3   2   1   3   3   3   3   2   1   1   3   3   1   2   1   3   1
1   1
 35  86  88 105  71 108  37 107  27  50  57  49  31  99 104  13  30  80  6
8  67
  1   2   2   2   2   2   1   2   3   1   1   1   1   2   2   3   3   2
1   1
```

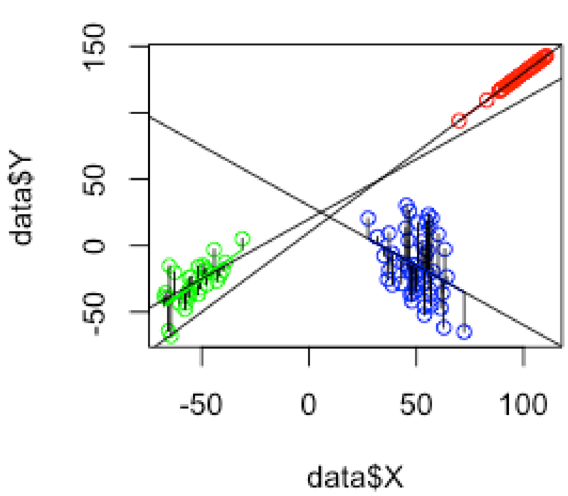
In [24]:
```
%%R

A = clusters[clusters == 1]
B = clusters[clusters == 2]
C = clusters[clusters == 3]

print(A)
print(names(A))
```

```
66 61 39 59 60 47 41 55 34 46 54 42 44 38 56 52 62 51 64 58 33 63 70 53 40
```

```
 36 
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 
  1 
65 48 69 43 32 45 35 37 50 57 49 31 68 67 
 1  1  1  1  1  1  1  1  1  1  1  1  1  1 
 [1] "66" "61" "39" "59" "60" "47" "41" "55" "34" "46" "54" "42" "44" "38"
 "56"
[16] "52" "62" "51" "64" "58" "33" "63" "70" "53" "40" "36" "65" "48" "69"
 "43"
[31] "32" "45" "35" "37" "50" "57" "49" "31" "68" "67"
```

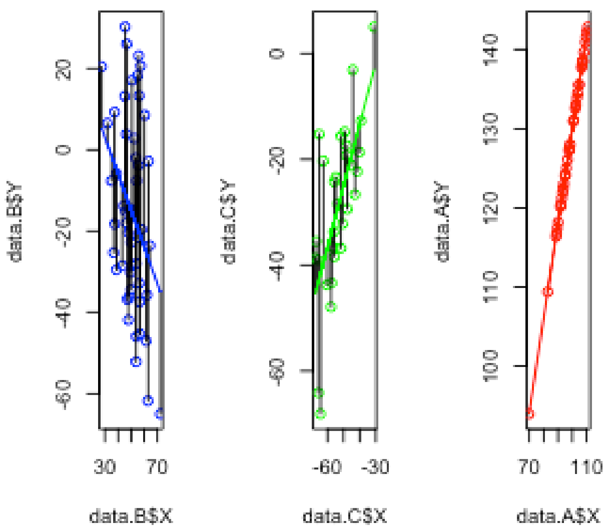
In [25]:
```
%%R

data.A = data[names(A),]
data.B = data[names(B),]
data.C = data[names(C),]

model.A = lm(data.A$Y ~ data.A$X)
model.B = lm(data.B$Y ~ data.B$X)
model.C = lm(data.C$Y ~ data.C$X)
```

In [26]:
```
%%R -r 86 -w 300 -h 300

plot(data$X, data$Y, type='n')

plot.cluster = function(cluster, model, color)
{
    points(cluster$X, cluster$Y, col=color)
    segments(cluster$X, fitted(model), cluster$X, cluster$Y)
    lines(cluster$X, fitted(model), col=color)
}

plot.cluster(data.A, model.A, 'Red')
plot.cluster(data.B, model.B, 'Blue')
plot.cluster(data.C, model.C, 'Green')
plot.trueLines()
```



In [36]:

```
%%R -r 86 -w 300 -h 300

par(mfrow=c(1,3))

plot(data.B$X, data.B$Y, type='n')
plot.cluster(data.B, model.B, 'Blue')

plot(data.C$X, data.C$Y, type='n')
plot.cluster(data.C, model.C, 'Green')

plot(data.A$X, data.A$Y, type='n')
plot.cluster(data.A, model.A, 'Red')
```



Interpreting and predicting with a regression (multivariate)

```
summary(model)
plot(model)
predict(model)
```

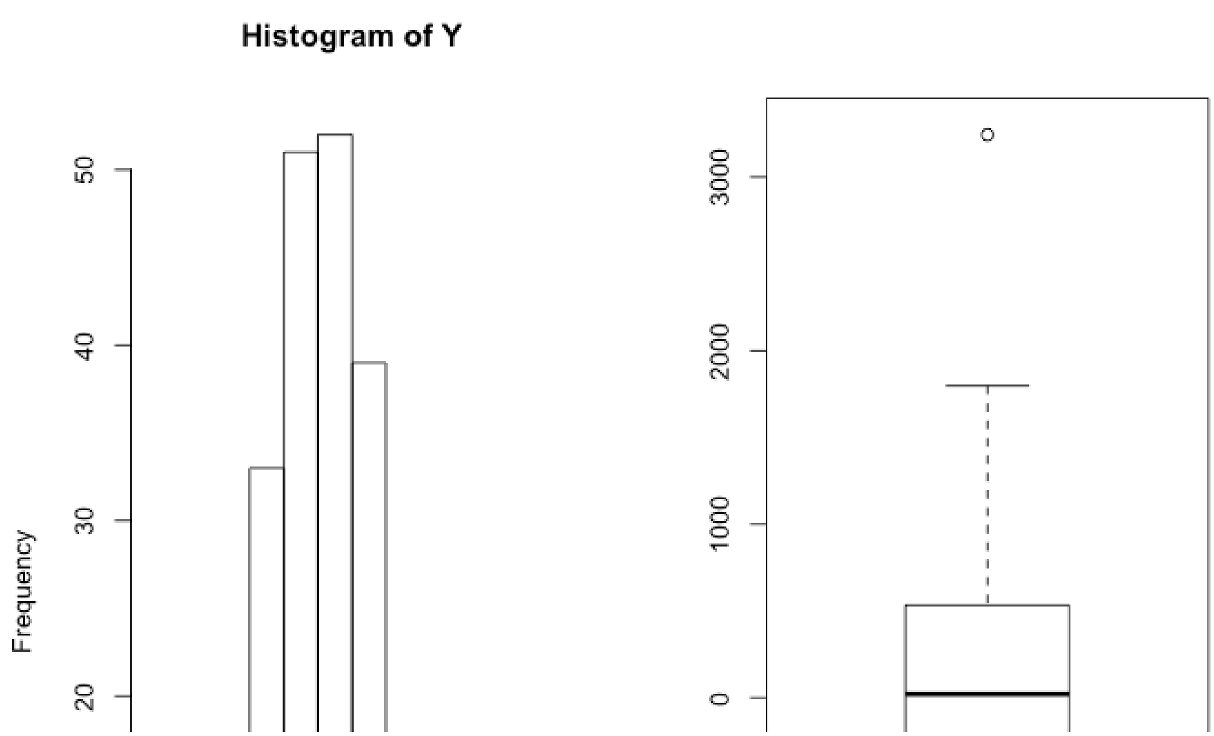
In [6]:

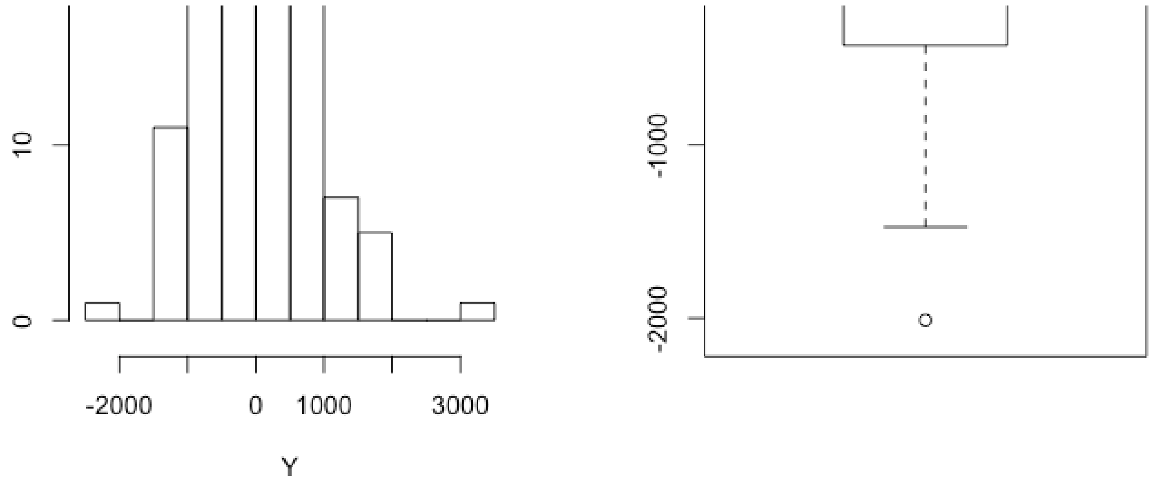
```
%%R -r 86 -w 300 -h 300

variable.number = 10
sample.size     = 200
variable.mean   = 100
variable.sd     = 200

X = replicate(variable.number, rnorm(sample.size, variable.mean, variable.
sd))

colnames(X) = paste('X', 1:variable.number, sep='')

print(head(X))
```

```
              X1         X2          X3          X4          X5         X6
   X7
[1,]  105.40420 285.02662  301.66928  408.69593   35.05395 186.95896 144.1
71848
[2,]   48.72788 -74.81719  156.10333  270.35009  287.49233 130.73525   6.1
13281
[3,]  164.08505 -98.62566   37.15868  119.49295 -228.03034  30.89589 -62.0
73405
[4,] -344.51869 140.54614  168.87509  -55.36539   19.74933 121.99664 279.5
96110
[5,]  -68.20329 195.18371  162.62079 -226.02108 -129.34261 679.01285 -31.1
87970
[6,] -105.68198 208.47142 -169.87913  -96.28966   19.38295 222.33534  69.9
09611
              X8         X9        X10
[1,] -229.68170   43.65169 -140.22419
[2,]  163.98883  252.95371  523.20702
[3,]  -55.77664  450.74193  493.41722
[4,]  145.30381  404.33338   99.73521
[5,]  398.36198  -46.24379 -198.82417
[6,]  202.53797 -125.22642  116.49260
```

In [9]:

```
%%R -r 86 -w 700 -h 700

alpha = 10
beta  = rnorm(variable.number)
error = rep(0, variable.number)#rnorm(variable.number)

Y = simulate.Y(X, alpha, beta, error)

par(mfrow=c(1,2))
hist(Y)
boxplot(Y)
```



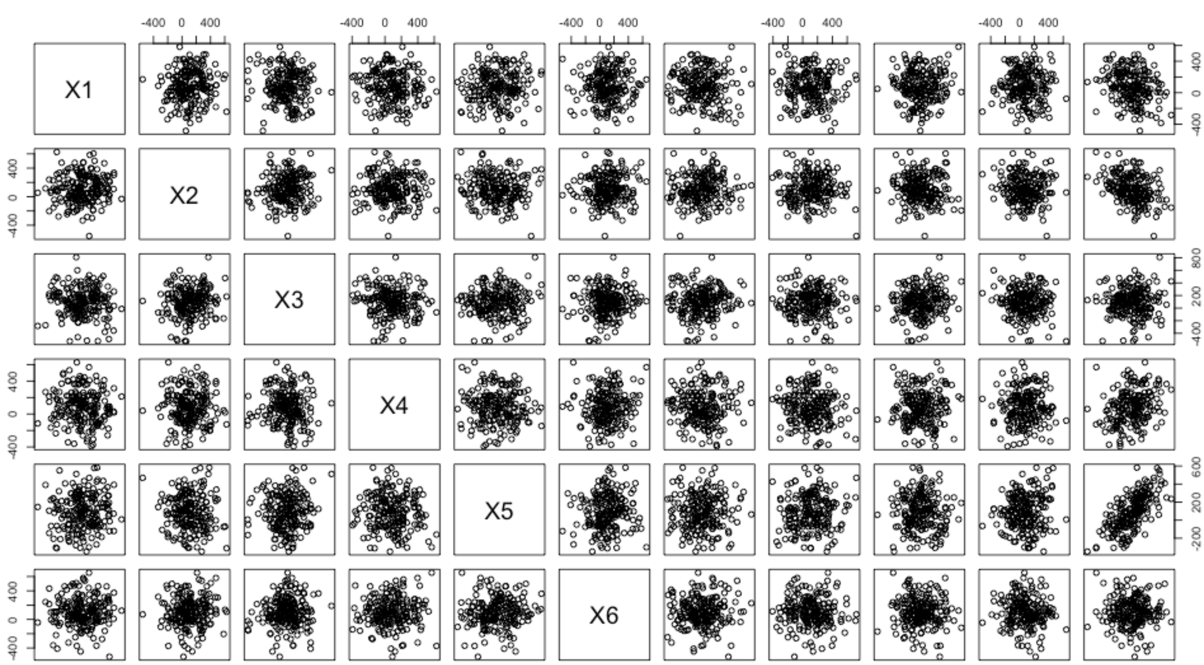


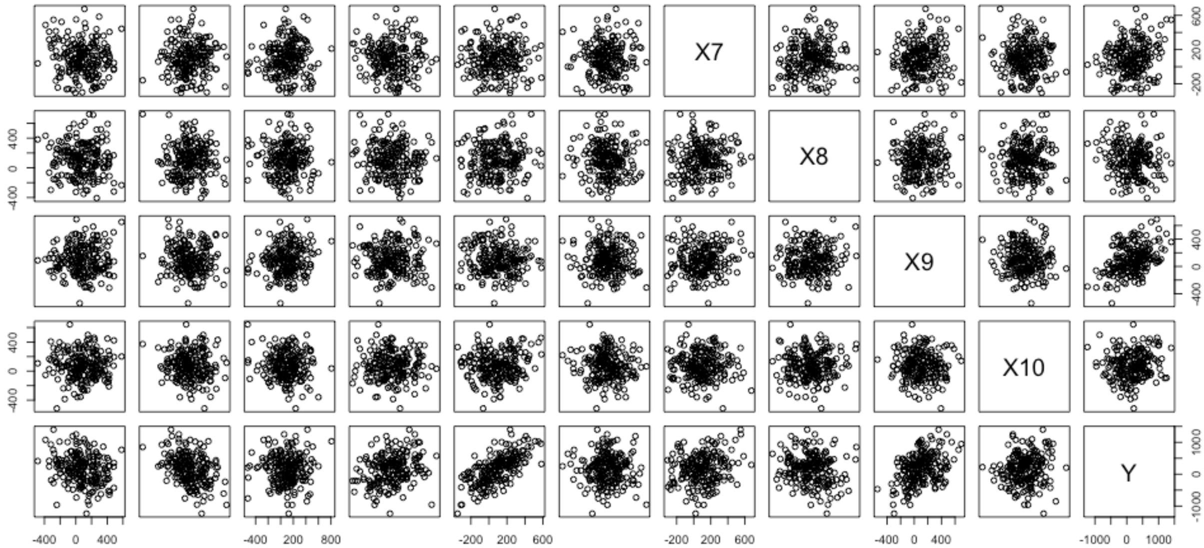
In [15]:
```
%%R

df = data.frame(X, Y=Y)
#print(head(df))

ind = df$Y > 2000

df.outliers = df[ind,]
print(df.outliers)
```

```
          X1       X2         X3          X4        X5        X6        X7       X8
177 321.0533 168.915 701.3128 -50.52474 5.549993 119.6978 88.54533 474.824
           X9       X10         Y
177 -321.7622 -124.5094 3243.655
```

In [15]:

In [14]:
```
%%R -r 86 -w 1000 -h 1000

plot(df)
```





In [49]:

We want to uncover a linear relation of the form:

$$Y = \alpha + \beta_1 X_1 + \cdots + \beta_n X_n$$

That is: We want to find

α and $\beta = (\beta_1, \ldots, \beta_n)$

such that the error committed on our sample (here `df`)

$$E_s = \|Y - (\alpha + X\beta)\|^2$$

is the smallest possible. The **residual** is the following vector computed on the sample:

$$\epsilon = Y - (\alpha + X\beta) = (\epsilon_1, \ldots, \epsilon_m),$$

where the i^{th} component indicates the error commited between the sample value of the i^{ith} observation and the value predicted by the model for this observation: i.e.,

$$\epsilon_i = y_i - (\alpha - \beta_1 x_{i1} + \cdots + \beta_n x_{in}).$$

An **an assumption of our model is that the residual are normally distributed around zero.** It can be interpreted as a **random error** or a **noise** in our data.

Using the linear model `lm`, we would like to see the strength of the correlation:

$$Y = \alpha + X\beta + \epsilon.$$

This is the generalization to several variables of what we just saw.

In [11]:

```
%%R

model = lm(Y ~ ., data=df)
stats = summary(model)
```

```
print(stats)
```

```
Call:
lm(formula = Y ~ ., data = df)

Residuals:
       Min         1Q     Median         3Q        Max 
-4.686e-12 -4.270e-14  4.240e-14  1.162e-13  8.004e-13 

Coefficients:
              Estimate Std. Error    t value Pr(>|t|)    
(Intercept)  1.000e+01  5.650e-14  1.770e+14   <2e-16 ***
X1          -6.079e-01  1.607e-16 -3.782e+15   <2e-16 ***
X2          -6.399e-01  1.692e-16 -3.782e+15   <2e-16 ***
X3          -1.302e-01  1.518e-16 -8.581e+14   <2e-16 ***
X4           6.121e-01  1.605e-16  3.814e+15   <2e-16 ***
X5           1.772e+00  1.695e-16  1.045e+16   <2e-16 ***
X6          -1.169e-01  1.603e-16 -7.295e+14   <2e-16 ***
X7           3.698e-01  1.699e-16  2.176e+15   <2e-16 ***
X8          -5.392e-01  1.496e-16 -3.605e+15   <2e-16 ***
X9           9.754e-01  1.548e-16  6.301e+15   <2e-16 ***
X10          1.575e-01  1.715e-16  9.182e+14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.486e-13 on 189 degrees of freedom
Multiple R-squared:      1,	Adjusted R-squared:      1 
F-statistic: 2.112e+31 on 10 and 189 DF,  p-value: < 2.2e-16
```

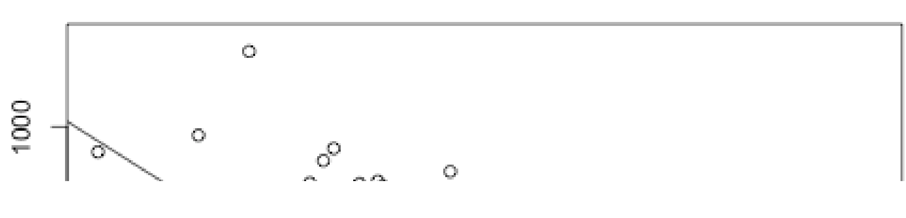
In [51]:

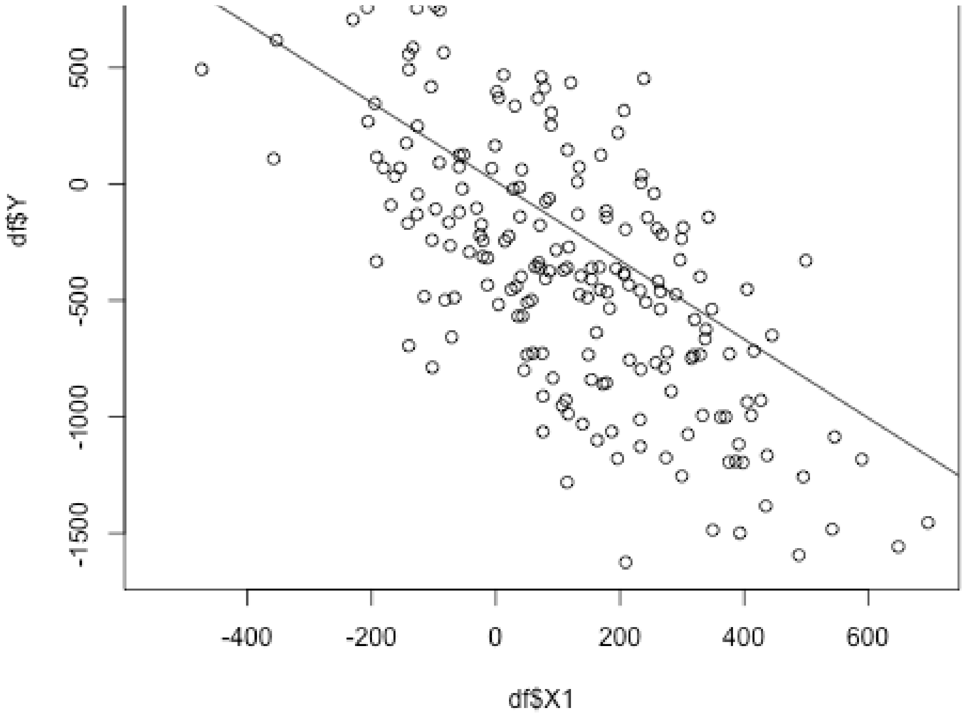
```
%%R



a = model$coefficients[1]
b = model$coefficients['X1']

print(c(a,b))
```

```
(Intercept)          X1 
  10.000000   -1.693349 
```

In [52]:

```
%%R

plot(df$X1, df$Y)
abline(a,b)
```





In [34]:

In [34]: