Homework 05 - Statistical Methods for Data Science

Letícia Negrão Pinto

DSSC - 2019

Core Statistics (CS)

Exercise 1.7

Let Y_1, Y_2 and Y_3 be independent $N(\mu, \sigma^2)$ r.v.s. Somehow using the matrix

$$\begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 2/3 & -1/3 & -1/3 \\ -1/3 & 2/3 & -1/3 \end{pmatrix}$$

show that $\bar{Y} = \sum_{i=1}^{3} \frac{Y_i}{3}$ and $\sum_{i=1}^{3} (Y_i - \bar{Y})^2$ are independent random variables.

For this exercise, let's consider the given a matrix of random Y_1, Y_2 and Y_3. In this case, it is possible to compute $\bar{Y} = \sum_{i=1}^{3} \frac{Y_i}{3}$ and $\sum_{i=1}^{3} (Y_i - \bar{Y})^2$, and the *covariance* between them:

```
# Let's call the matrix B:
B <- matrix(nrow=3, ncol=3, c((1/3), (1/3), (1/3),
                              (2/3),(-1/3), (-1/3),
                              (-1/3), (2/3),(-1/3)), byrow=TRUE)


mean_B <- c( mean(B[1,]), mean(B[2,]), mean(B[3,]) )

s <- c(sum((B[1,]-mean_B[1])^2), sum((B[2,]-mean_B[2])^2),   sum((B[3,]-mean_B[3])^2)  )

# Computing the covariance:
round(cov(mean_B, s))
```

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

Since the *covariance* between the variables in question is *zero*, we can assume they are independent.

Exercise 4.3

Random variables X and Y have joint p.d.f. $f(x, y) = kx^{\alpha}y^{\beta}$, $0 \leq x \leq 1, 0 \leq y \leq 1$. Assume that you have n independent pairs of observations (x_i, y_i).

(a) Evaluate k in terms of the α and β.

Since $0 \leq x \leq 1$ and $0 \leq y \leq 1$, and we are dealing with a p.d.f., we know that the integral from 0 to 1 in both variables must be equal 1.

$$\int_0^1 \int_0^1 kx^{\alpha}y^{\beta} dxdy = k \int_0^1 \int_0^1 x^{\alpha}y^{\beta} dxdy = \frac{k}{(\alpha+1)(\beta+1)} = 1$$

In this scenario, we have that k is the normalization factor given in terms of the α and β:

$$k = (\alpha+1)(\beta+1)$$

(b) Find the maximum likelihood estimators of α and β.

Assuming that each data point is independent, the likelihood of all of our data is the product of the likelihood of each data point. Mathematically, the likelihood of our data give parameters θ is:

$$L(\theta) = \prod_{i=1}^{n} f(X_i|\theta)$$

In our case, we have:

$$L(\alpha, \beta) = \prod_{i=1}^{n} (\alpha+1)(\beta+1)x_i^{\alpha} y_i^{\beta} = (\alpha+1)^n (\beta+1)^n \prod_{i=1}^{n} x_i^{\alpha} y_i^{\beta}$$

In maximum likelihood estimation (MLE) our goal is to chose values of our parameters (θ) that maximizes the likelihood function. In this case, $\hat{\theta}$ represent the best choice of values for our parameters:

$$\hat{\theta} = argmax_{\theta} L(\theta)$$

The *argmax* of a function is the value of the domain at which the function is maximized. An important property of *argmax* is that since log is a monotone function, the *argmax* of a function is the same as the *argmax* of the log of the function:

$$LL(\theta) = logL(\theta) = log \prod_{i=1}^{n} f(X_i|\theta) = \sum_{i=1}^{n} logf(X_i|\theta)$$

Here we have that $LL(\theta)$ is the log-likelihood function.

Going back to the case in study, we have that:

$$LL(\alpha, \beta) = logL(\alpha, \beta) = log\left[(\alpha+1)^n (\beta+1)^n \prod_{i=1}^{n} x_i^{\alpha} y_i^{\beta}\right]$$

That is:

$$LL(\alpha, \beta) = nlog(\alpha+1) + nlog(\beta+1) + \sum_{i=1}^{n} log(x_i^{\alpha}) + \sum_{i=1}^{n} log(y_i^{\beta})$$

$$= nlog(\alpha+1) + nlog(\beta+1) + \alpha \sum_{i=1}^{n} log(x_i) + \beta \sum_{i=1}^{n} log(y_i)$$

In order to find a maximum likelihood estimator, first we write the log-likelihood of the data given our parameters and choose the value of parameters that maximize the log-likelihood function.

That said, in order to estimate α we do:

$$\frac{\partial LL(\alpha, \beta)}{\partial \alpha} = 0$$

$$\frac{\partial}{\partial \alpha}\left[nlog(\alpha+1) + nlog(\beta+1) + \alpha \sum_{i=1}^{n} log(x_i) + \beta \sum_{i=1}^{n} log(y_i)\right] = 0$$

$$n\frac{\partial}{\partial \alpha} log(\alpha+1) + \frac{\partial}{\partial \alpha}\left[\alpha \sum_{i=1}^{n} log(x_i)\right] = 0$$

$$n\frac{\partial}{\partial\alpha}log(\alpha+1)+\sum_{i=1}^{n}log(x_i)=\frac{n}{\alpha+1}+\sum_{i=1}^{n}log(x_i)=0$$

$$\frac{n}{\alpha+1}=-\sum_{i=1}^{n}log(x_i)$$

$$\alpha+1=-\frac{n}{\sum_{i=1}^{n}log(x_i)}$$

And finally we have the estimate for α:

$$\hat{\alpha}=-\frac{n}{\sum_{i=1}^{n}log(x_i)}-1$$

Similarly, in order to estimate β we do:

$$\frac{\partial LL(\alpha,\beta)}{\partial\beta}=0$$

$$\frac{\partial}{\partial\beta}\left[nlog(\alpha+1)+nlog(\beta+1)+\alpha\sum_{i=1}^{n}log(x_i)+\beta\sum_{i=1}^{n}log(y_i)\right]=0$$

$$n\frac{\partial}{\partial\beta}log(\beta+1)+\sum_{i=1}^{n}log(y_i)=\frac{n}{\beta+1}+\sum_{i=1}^{n}log(y_i)=0$$

And finally we have the estimate for β:

$$\hat{\beta}=-\frac{n}{\sum_{i=1}^{n}log(y_i)}-1$$

(c) Find approximate variances of $\hat{\alpha}$ and $\hat{\beta}$.

The variance of an Maximum Likelihood estimator is calculated by the inverse of the Information matrix:

$$Var(\hat{\theta})=[I(\theta)]^{-1}$$

The "Information", $I(\theta)$, can be defined as:

$$I(\theta)=-E\left(\frac{\partial^2 LL(\theta)}{\partial^2\theta}\right)=-\int\frac{\partial^2 LL(\theta)}{\partial^2\theta}f(X|\theta)d\theta$$

In this context, in order to find the variance of $\hat{\alpha}$, we first compute the second derivative of the log-likelihood. In previous sections we had that:

$$\frac{\partial LL(\alpha,\beta)}{\partial\alpha}=\frac{n}{\alpha+1}+\sum_{i=1}^{n}log(x_i)$$

So the second derivative can be computed as:

$$\frac{\partial^2 LL(\alpha,\beta)}{\partial^2\alpha}=\frac{\partial}{\partial\alpha}\left[\frac{n}{\alpha+1}+\sum_{i=1}^{n}log(x_i)\right]=-\frac{n}{(\alpha+1)^2}$$

For α we have:

$$E\left(\frac{\partial^2 LL(\theta)}{\partial^2 \alpha}\right) = \int\int -\frac{n}{(\alpha+1)^2}(\alpha+1)(\beta+1)x^\alpha y^\beta dxdy$$

$$= -n\frac{(\beta+1)}{(\alpha+1)}\int\int x^\alpha y^\beta dxdy = -n\frac{(\beta+1)}{(\alpha+1)}\frac{1}{(\alpha+1)(\beta+1)} = -\frac{n}{(\alpha+1)^2}$$

So the Information is:

$$I(\alpha) = \frac{n}{(\alpha+1)^2}$$

Analogously, for β, the Information is:

$$I(\beta) = \frac{n}{(\beta+1)^2}$$

Now we can compute the variance for $\hat{\alpha}$:

$$Var(\hat{\alpha}) = \frac{1}{I(\alpha)} = \frac{1}{\frac{n}{(\alpha+1)^2}} = \frac{(\alpha+1)^2}{n}$$

Analogously, the variance for $\hat{\beta}$ is then given by:

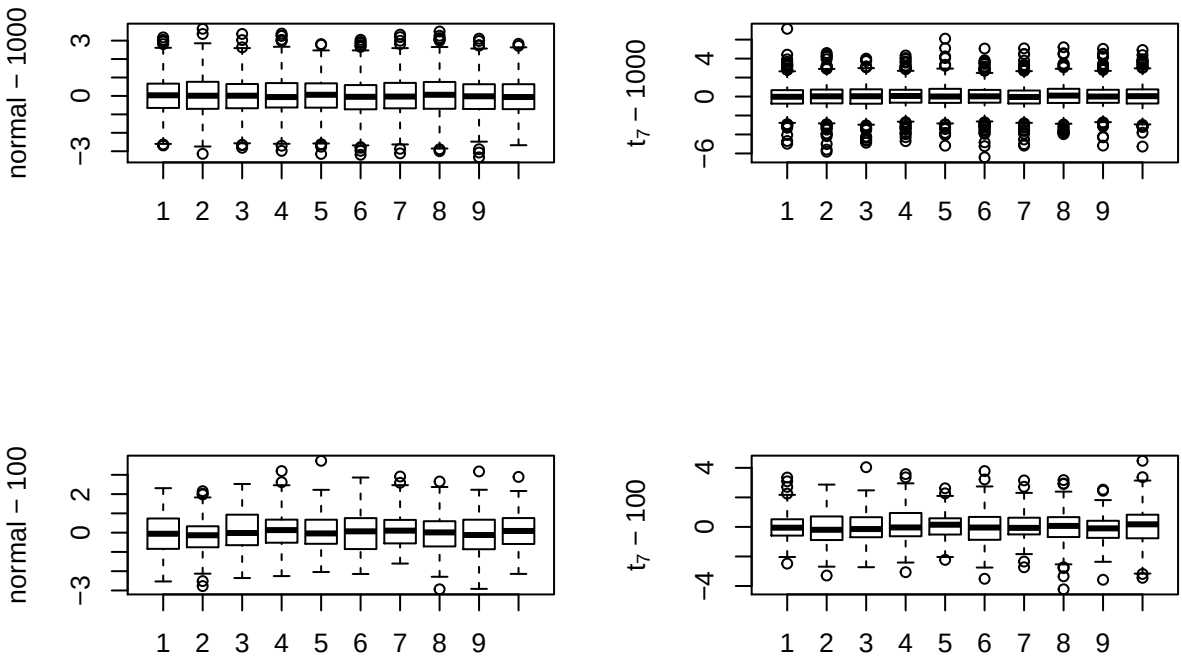
$$Var(\hat{\beta}) = \frac{(\beta+1)^2}{n}$$

Data Analysis and Graphics Using R (DAAG)

Exercise 4.5

The following code draws, in a 2 x 2 layout, 10 boxplots of random samples of 1000 from a normal distribution, 10 boxplots of random samples of 1000 from a t-distribution with 7 d.f., 10 boxplots of random samples of 200 from a normal distribution, and 10 boxplots of random samples of 200 from a t-distribution with 7 d.f.:

```
oldpar <- par(mfrow=c(2,2))
tenfold1000 <- rep(1:10, rep(1000,10))
boxplot(split(rnorm(1000*10), tenfold1000), ylab="normal - 1000")
boxplot(split(rt(1000*10, 7), tenfold1000),
ylab=expression(t[7]*" - 1000"))
tenfold100 <- rep(1:10, rep(100, 10))
boxplot(split(rnorm(100*10), tenfold100), ylab="normal - 100")
boxplot(split(rt(100*10, 7), tenfold100),
ylab=expression(t[7]*" - 100"))
```




```
par(oldpar)
```

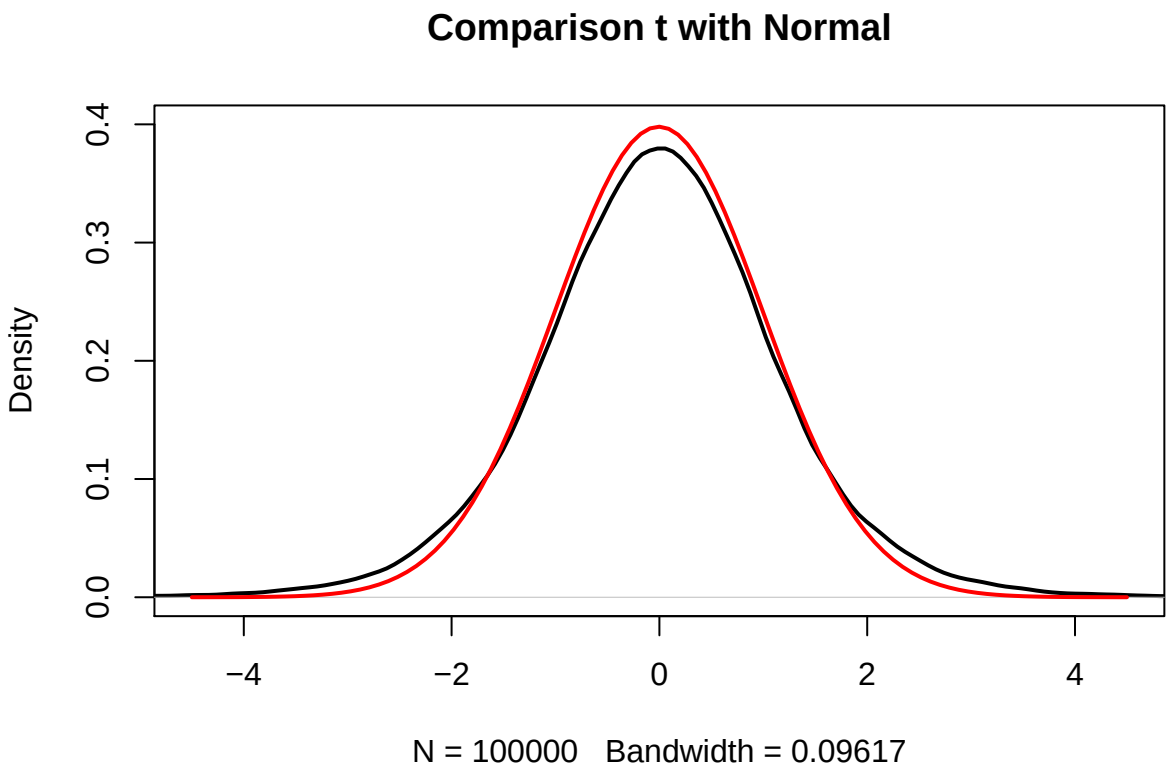
Refer back to the discussion of heavy-tailed distributions in Subsection 3.2.2, and comment on the different numbers and configurations of points that are flagged as possible outliers.

From the box plots, one can notice that not only the t distributions appeared to contain more outliers (showed as individual points in the graph), but also the number of outliers seems to increase with the number of samples considered. In order to investigate this behaviour and compare both t and *normal* distributions (with the same characteristics as shown previously), the following graph is plotted:

```
# Comparing the Normal and t distributions as were used previously:
set.seed(2019)
normal <- c(rnorm(100000))
t <- c(rt(100000, 7))

# t
plot(density(t), xlim=c(-4.5,4.5), ylim=c(0,0.4),
     main="Comparison t with Normal", lwd=2)

#Normal
sigma <- sd   ( normal )
mu    <- mean( normal )
curve( dnorm(x, mu, sigma), add = TRUE, col="red", lwd=2 )
```



As can be seen in the graph, the t distribution presents a heavier tail. According to the DAAG book, *If the distribution is symmetric, but "heavy-tailed", then a higher proportion of values are out beyond the boxplot whiskers on both sides.* This behaviour can be proved when computing the percentage of outliers for both distributions:

```
# Normal distribution - 1st and 3rd Quartiles
Q25 <- quantile(normal, 0.25)
Q75 <- quantile(normal, 0.75)
IQR <- Q75 - Q25
#lower:
l_normal <- Q25 - 1.5*IQR
#upper:
u_normal <- Q75 + 1.5*IQR

#outliers:
n_out = c(which(normal < l_normal | normal > u_normal))

# t distribution - 1st and 3rd Quartiles
Q25t <- quantile(t, 0.25)
Q75t <- quantile(t, 0.75)
IQRt <- Q75t - Q25t
#lower:
l_t <- Q25t - 1.5*IQRt
#upper:
u_t <- Q75t + 1.5*IQRt

#outliers:
```

```
t_out = c(which(t < l_t | t > u_t))

# Percentage of outliers - Normal distribution:
(length(n_out)/length(normal))*100
```

```
## [1] 0.72
```

```
# Percentage of outliers - t distribution:
(length(t_out)/length(t))*100
```

```
## [1] 2.419
```

As expected, t distribution presented a higher percentage of outliers (2.42%, while normal distribution presented 0.72%). The article called "*A generalized box plot for skewed and heavy-tailed distributions*"[1], by Bruffaerts et al., discusses this issue of the standard box plot and proposes a generalized approach, suitable for skewed and heavy-tailed distributions. In conclusion, one must be aware of this issue in order to correctly spot outliers.

Exercise 11.4

Copy down the email spam data set from the web site given at the start of Section 11.2. Carry out a tree-based regression using all 57 available explanatory variables.

First of all, the data set was downloaded from the website: http://archive.ics.uci.edu/ml/datasets/Spambase.

```
# We read the file and confirm which is the content of it
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
spam <- read.table("spambase.data", header = FALSE, sep = ",")
names_spam <- scan("names.txt", what = "")
names_spam[58] <- "spam_or_not" # spam or not
names(spam) <- names_spam # naming variables
table(spam[,58]) # Confirm that we only have 0s and 1s (sanity test)
```

```
##
##    0    1
## 2788 1813
```

```
glimpse(spam)
```

```
## Observations: 4,601
## Variables: 58
## $ make                      <dbl> 0.00, 0.21, 0.06, 0.00, 0.00, 0.00,...
## $ address                   <dbl> 0.64, 0.28, 0.00, 0.00, 0.00, 0.00,...
## $ all                       <dbl> 0.64, 0.50, 0.71, 0.00, 0.00, 0.00,...
## $ `3d`                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

[1] Bruffaerts, C. et al. "A generalized box plot for skewed and heavy-tailed distributions", Statistics and Probability Letters 95 (2014) 110–117.

```
## $ our                          <dbl> 0.32, 0.14, 1.23, 0.63, 0.63, 1.85,...
## $ over                         <dbl> 0.00, 0.28, 0.19, 0.00, 0.00, 0.00,...
## $ remove                       <dbl> 0.00, 0.21, 0.19, 0.31, 0.31, 0.00,...
## $ internet                     <dbl> 0.00, 0.07, 0.12, 0.63, 0.63, 1.85,...
## $ order                        <dbl> 0.00, 0.00, 0.64, 0.31, 0.31, 0.00,...
## $ mail                         <dbl> 0.00, 0.94, 0.25, 0.63, 0.63, 0.00,...
## $ receive                      <dbl> 0.00, 0.21, 0.38, 0.31, 0.31, 0.00,...
## $ will                         <dbl> 0.64, 0.79, 0.45, 0.31, 0.31, 0.00,...
## $ people                       <dbl> 0.00, 0.65, 0.12, 0.31, 0.31, 0.00,...
## $ report                       <dbl> 0.00, 0.21, 0.00, 0.00, 0.00, 0.00,...
## $ addresses                    <dbl> 0.00, 0.14, 1.75, 0.00, 0.00, 0.00,...
## $ free                         <dbl> 0.32, 0.14, 0.06, 0.31, 0.31, 0.00,...
## $ business                     <dbl> 0.00, 0.07, 0.06, 0.00, 0.00, 0.00,...
## $ email                        <dbl> 1.29, 0.28, 1.03, 0.00, 0.00, 0.00,...
## $ you                          <dbl> 1.93, 3.47, 1.36, 3.18, 3.18, 0.00,...
## $ credit                       <dbl> 0.00, 0.00, 0.32, 0.00, 0.00, 0.00,...
## $ your                         <dbl> 0.96, 1.59, 0.51, 0.31, 0.31, 0.00,...
## $ font                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ `000`                        <dbl> 0.00, 0.43, 1.16, 0.00, 0.00, 0.00,...
## $ money                        <dbl> 0.00, 0.43, 0.06, 0.00, 0.00, 0.00,...
## $ hp                           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ hpl                          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ george                       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ `650`                        <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...
## $ lab                          <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ labs                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ telnet                       <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ `857`                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ data                         <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...
## $ `415`                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ `85`                         <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ technology                   <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...
## $ `1999`                       <dbl> 0.00, 0.07, 0.00, 0.00, 0.00, 0.00,...
## $ parts                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ pm                           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ direct                       <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00,...
## $ cs                           <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ meeting                      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ original                     <dbl> 0.00, 0.00, 0.12, 0.00, 0.00, 0.00,...
## $ project                      <dbl> 0.00, 0.00, 0.00, 0.00, 0.00, 0.00,...
## $ re                           <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00,...
## $ edu                          <dbl> 0.00, 0.00, 0.06, 0.00, 0.00, 0.00,...
## $ table                        <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ conference                   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
## $ `;`                          <dbl> 0.000, 0.000, 0.010, 0.000, 0.000, ...
## $ `(`                          <dbl> 0.000, 0.132, 0.143, 0.137, 0.135, ...
## $ `[`                          <dbl> 0.000, 0.000, 0.000, 0.000, 0.000, ...
## $ `!`                          <dbl> 0.778, 0.372, 0.276, 0.137, 0.135, ...
## $ `$`                          <dbl> 0.000, 0.180, 0.184, 0.000, 0.000, ...
## $ `#`                          <dbl> 0.000, 0.048, 0.010, 0.000, 0.000, ...
## $ capital_run_length_average <dbl> 3.756, 5.114, 9.821, 3.537, 3.537, ...
## $ capital_run_length_longest <int> 61, 101, 485, 40, 40, 15, 4, 11, 44...
## $ capital_run_length_total   <int> 278, 1028, 2259, 191, 191, 54, 112,...
## $ spam_or_not                <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
```

From what can be seen above, it is possible to notice that we have, indeed, 57 explanatory variables and yet one more, the target variable (*spam_or_not*). The package "*rpart*" was chosen to analyse the data. The package may be used in recursive partitioning for classification, regression, and survival trees.

```
library(rpart)
set.seed(2019)
spam.rpart <- rpart(spam_or_not ~ ., data = spam, cp = 1e-05, method = "class")

# The printcp function is used to print a table of optimal prunings based on
# a complexity parameter.
printcp(spam.rpart)
```

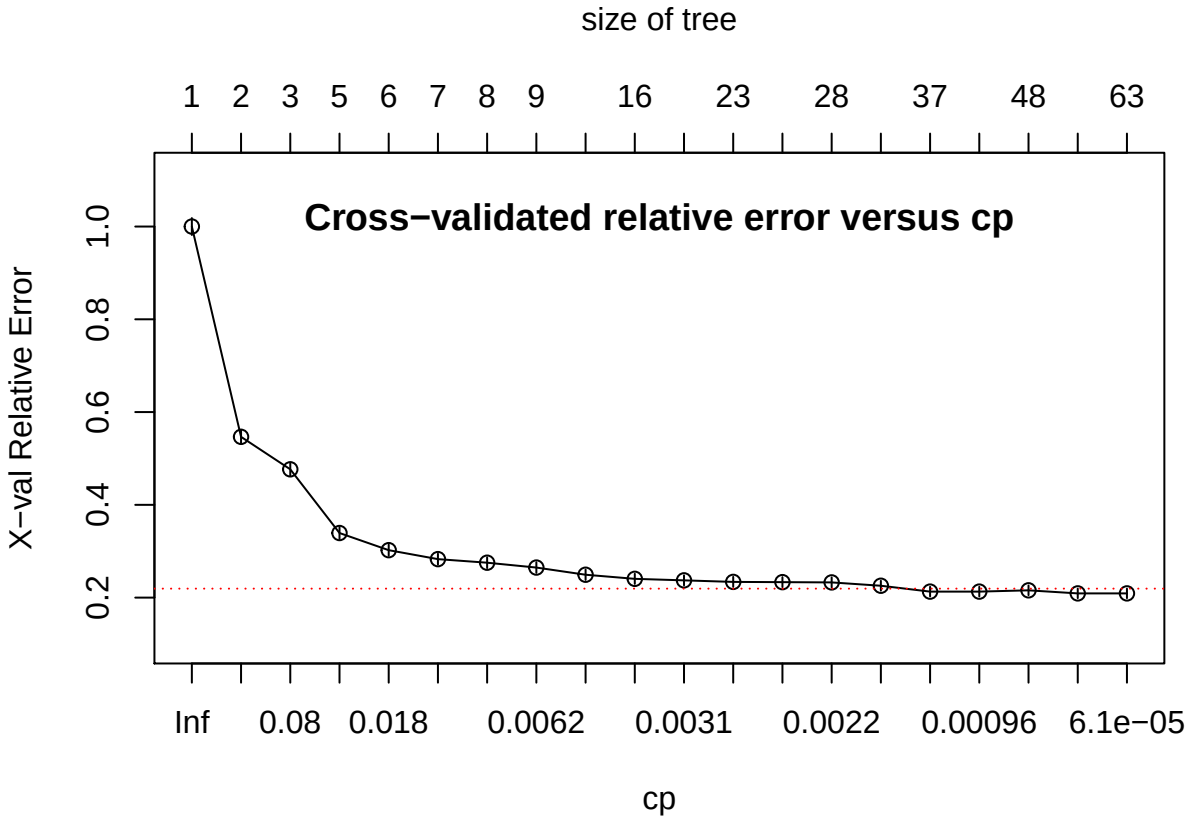
```
##
## Classification tree:
## rpart(formula = spam_or_not ~ ., data = spam, method = "class",
##     cp = 1e-05)
##
## Variables actually used in tree construction:
##  [1] !                          $
##  [3] (                          ;
##  [5] 1999                       650
##  [7] address                    capital_run_length_average
##  [9] capital_run_length_longest capital_run_length_total
## [11] data                       edu
## [13] email                      font
## [15] free                       george
## [17] hp                         internet
## [19] money                      our
## [21] over                       re
## [23] remove                     technology
## [25] will                       you
## [27] your
##
## Root node error: 1813/4601 = 0.39404
##
## n= 4601
##
##            CP nsplit rel error  xerror     xstd
## 1  0.47655819      0   1.00000 1.00000 0.018282
## 2  0.14892443      1   0.52344 0.54661 0.015380
## 3  0.04302261      2   0.37452 0.47656 0.014611
## 4  0.03088803      4   0.28847 0.33922 0.012732
## 5  0.01047987      5   0.25758 0.30226 0.012119
## 6  0.00827358      6   0.24710 0.28296 0.011776
## 7  0.00717044      7   0.23883 0.27523 0.011634
## 8  0.00529509      8   0.23166 0.26475 0.011437
## 9  0.00441258     14   0.19581 0.24931 0.011136
## 10 0.00358522     15   0.19140 0.24049 0.010958
## 11 0.00275786     19   0.17705 0.23718 0.010890
## 12 0.00257400     22   0.16878 0.23387 0.010822
## 13 0.00220629     25   0.16106 0.23331 0.010810
## 14 0.00211436     27   0.15665 0.23276 0.010799
## 15 0.00165472     33   0.14396 0.22559 0.010648
## 16 0.00110314     36   0.13900 0.21291 0.010372
## 17 0.00082736     43   0.13127 0.21291 0.010372
```

```
## 18 0.00055157      47   0.12796 0.21566 0.010433
## 19 0.00036771      53   0.12466 0.20905 0.010286
## 20 0.00001000      62   0.12135 0.20905 0.010286
```

Determine the change in the cross-validation estimate of predictive accuracy.

Since we are using the "*rpart*" library, we should simply use the function "*plotcp*" in order to have a visual representation of the cross-validation results in an rpart object:

```
plotcp(spam.rpart, minline = TRUE, col="red", ldw=6)
# minline is related to whether a horizontal line is drawn 1SE above the
# minimum of the curve
title("Cross-validated relative error versus cp", line = -2)
```



Analysing the graph above, it is possible to notice that the cross-validated relative error decreases and seems to stabilize with the increase of "*cp*". The red horizontal line is drawn 1SE above the minimum of the curve. Based on those results, we can poceed with the pruning.

```
# Pruning
# Here we can compare the results for two values of "cp"

# First we consider a non-optimal parameter:
spam.rpart1 <- prune(spam.rpart, cp = 0.01)
printcp(spam.rpart1)
```

```
##
## Classification tree:
## rpart(formula = spam_or_not ~ ., data = spam, method = "class",
##     cp = 1e-05)
```

```
##
## Variables actually used in tree construction:
## [1] !                          $
## [3] capital_run_length_total free
## [5] hp                         remove
##
## Root node error: 1813/4601 = 0.39404
##
## n= 4601
##
##         CP nsplit rel error  xerror     xstd
## 1 0.476558      0   1.00000 1.00000 0.018282
## 2 0.148924      1   0.52344 0.54661 0.015380
## 3 0.043023      2   0.37452 0.47656 0.014611
## 4 0.030888      4   0.28847 0.33922 0.012732
## 5 0.010480      5   0.25758 0.30226 0.012119
## 6 0.010000      6   0.24710 0.28296 0.011776
```

```
# And now using the best parameter:
spam.rpart2 <- prune(spam.rpart, cp = 0.001)
printcp(spam.rpart2)
```

```
##
## Classification tree:
## rpart(formula = spam_or_not ~ ., data = spam, method = "class",
##     cp = 1e-05)
##
## Variables actually used in tree construction:
##  [1] !                          $
##  [3] (                          ;
##  [5] 1999                       650
##  [7] capital_run_length_average capital_run_length_longest
##  [9] capital_run_length_total   edu
## [11] email                      font
## [13] free                       george
## [15] hp                         internet
## [17] money                      our
## [19] over                       re
## [21] remove                     technology
## [23] you                        your
##
## Root node error: 1813/4601 = 0.39404
##
## n= 4601
##
##           CP nsplit rel error  xerror     xstd
## 1  0.4765582      0   1.00000 1.00000 0.018282
## 2  0.1489244      1   0.52344 0.54661 0.015380
## 3  0.0430226      2   0.37452 0.47656 0.014611
## 4  0.0308880      4   0.28847 0.33922 0.012732
## 5  0.0104799      5   0.25758 0.30226 0.012119
## 6  0.0082736      6   0.24710 0.28296 0.011776
## 7  0.0071704      7   0.23883 0.27523 0.011634
## 8  0.0052951      8   0.23166 0.26475 0.011437
## 9  0.0044126     14   0.19581 0.24931 0.011136
```

```
## 10 0.0035852     15   0.19140 0.24049 0.010958
## 11 0.0027579     19   0.17705 0.23718 0.010890
## 12 0.0025740     22   0.16878 0.23387 0.010822
## 13 0.0022063     25   0.16106 0.23331 0.010810
## 14 0.0021144     27   0.15665 0.23276 0.010799
## 15 0.0016547     33   0.14396 0.22559 0.010648
## 16 0.0011031     36   0.13900 0.21291 0.010372
## 17 0.0010000     43   0.13127 0.21291 0.010372
```

Comparing both results we see that when choosing the best parameter we really have a gain in predictive accuracy.

Bayesian Computation (BC)

Exercise 2.5

Estimating a normal mean with a discrete prior

Suppose you are interested in estimating the average total snowfall per year μ (in inches) for a large city on the East Coast of the United States. Assume individual yearly snow totals $y_1, ..., y_n$ are collected from a population that is assumed to be normally distributed with mean μ and known standard deviation $\sigma = 10$ inches.

a) Before collecting data, suppose you believe that the mean snowfall μ can be the values 20, 30, 40, 50, 60, and 70 inches with the following probabilities:

μ 20 30 40 50 60 70

$g(\mu)$ .1 .15 .25 .25 .15 .1

Place the values of μ in the vector mu and the associated prior probabilities in the vector prior.

```
# Vector Mu:
mu <- c(20, 30, 40, 50, 60, 70)

# Vector Prior:
prior <- c(.1, .15, .25, .25, .15, .1)
```

b) Suppose you observe the yearly snowfall totals 38.6, 42.4, 57.5, 40.5, 51.7, 67.1, 33.4, 60.9, 64.1, 40.1, 40.7, and 6.4 inches. Enter these data into a vector y and compute the sample mean ybar.

```
# Vector y:
y <- c(38.6, 42.4, 57.5, 40.5, 51.7, 67.1, 33.4, 60.9, 64.1, 40.1, 40.7, 6.4)

# Sample mean:
ybar <- mean(y)
ybar
```

```
## [1] 45.28333
```

c) In this problem, the likelihood function is given by

$$L(\mu) \propto exp\left(-\frac{n}{2\sigma^2}(\mu - \bar{y})^2 \right)$$

where $\bar{y}$ is the sample mean. Compute the likelihood on the list of values in mu and place the likelihood values in the vector *like*.

```r
# Considering the values of mu:
n <- length(mu)
sigma <- 10 # Given by the exercise

# Likelihood Values:
like <- c()

for (i in 1 : n)
{
  like[i] <- exp(-(n/(2*sigma^2))*(mu[i]-ybar)^2)
}

like
```

```
## [1] 4.691993e-09 9.051514e-04 4.328308e-01 5.130365e-01 1.507341e-03
## [6] 1.097761e-08
```

d) One can compute the posterior probabilities for μ using the formula

$$post = prior * like/sum(prior * like)$$

Compute the posterior probabilities of μ for this example.

```r
# Posterior probabilities:
post <- prior*like/sum(prior*like)
post
```

```
## [1] 1.981176e-09 5.732950e-04 4.569028e-01 5.415692e-01 9.547031e-04
## [6] 4.635252e-09
```

e) Using the function *discint*, find an 80% probability interval for μ.

The function *discint* is a function of the library *LearnBayes* and computes a highest probability interval for a discrete probability distribution. [2]

```r
library(LearnBayes)

dist_prior <- cbind(mu, prior)
dist_prior
```

```
##      mu prior
## [1,] 20  0.10
## [2,] 30  0.15
## [3,] 40  0.25
## [4,] 50  0.25
## [5,] 60  0.15
## [6,] 70  0.10
```

```r
dist_posterior <- cbind(mu, post)
dist_posterior
```

```
##      mu         post
## [1,] 20 1.981176e-09
## [2,] 30 5.732950e-04
## [3,] 40 4.569028e-01
## [4,] 50 5.415692e-01
```

[2] R documentation

```
## [5,] 60 9.547031e-04
## [6,] 70 4.635252e-09
```

```
# We wish to summarize the discrete predictive distribution by an
# interval that covers at least 80% of the probability.
covprob=.8

# Considering Prior probabilities:
discint(dist_prior,covprob)
```

```
## $prob
## [1] 0.8
##
## $set
## [1] 30 40 50 60
```

```
# Considering Posterior probabilities:
discint(dist_posterior,covprob)
```

```
## $prob
## [1] 0.998472
##
## $set
## [1] 40 50
```

In the previous output, "*prob*" is the exact probability content of interval and "*set*" is the set of values of the probability interval.

From the results presented above, one can notice that considering the *prior* probabilities we had that the probability that mu falls in the interval {30, 60} is 80%.

On the other hand, when considering the *posterior* probabilities we have that the probability that mu falls in the interval {40, 50} is 99.84%