Introduction to R for Statistics & Data Science*

Peter Martey Addo
Data Scientist de DIVISION INFORMATION & INNOVATION
SNCF - DIRECTION DU MATERIEL

14 Février 2017

Contents

Introduction to R

Overview

R is a powerful **open-source statistical programming language** software program for advanced data techniques. It allows you to work on:

- statistical analysis

*Diffusion Limitée

Figure 1:

- machine learning & deep learning
- data visualization.

Commenting and R Environments

Adding Comments to R Code

Comments are meant to help describe the code, and should be used when the purpose of a line of code is unclear.In `R`, lines beginning with a pound character (`#`) are each a **comment**. Comments should be clear, concise, and helpful.

Here's an example of some commented R code, which will be explained below.

```
# Describe purpose of computation
amount.per.day <- 10 # amount in euros
result <- 10 * 7  # budget for the week
```

Interactive R Sessions

An interactive R session can be opened within terminal. To open, type the letter `R` into your terminal and hit `enter`, an interactive R session will appear :

Limitation: this environment doesn't provide much of an interface for writing R scripts.

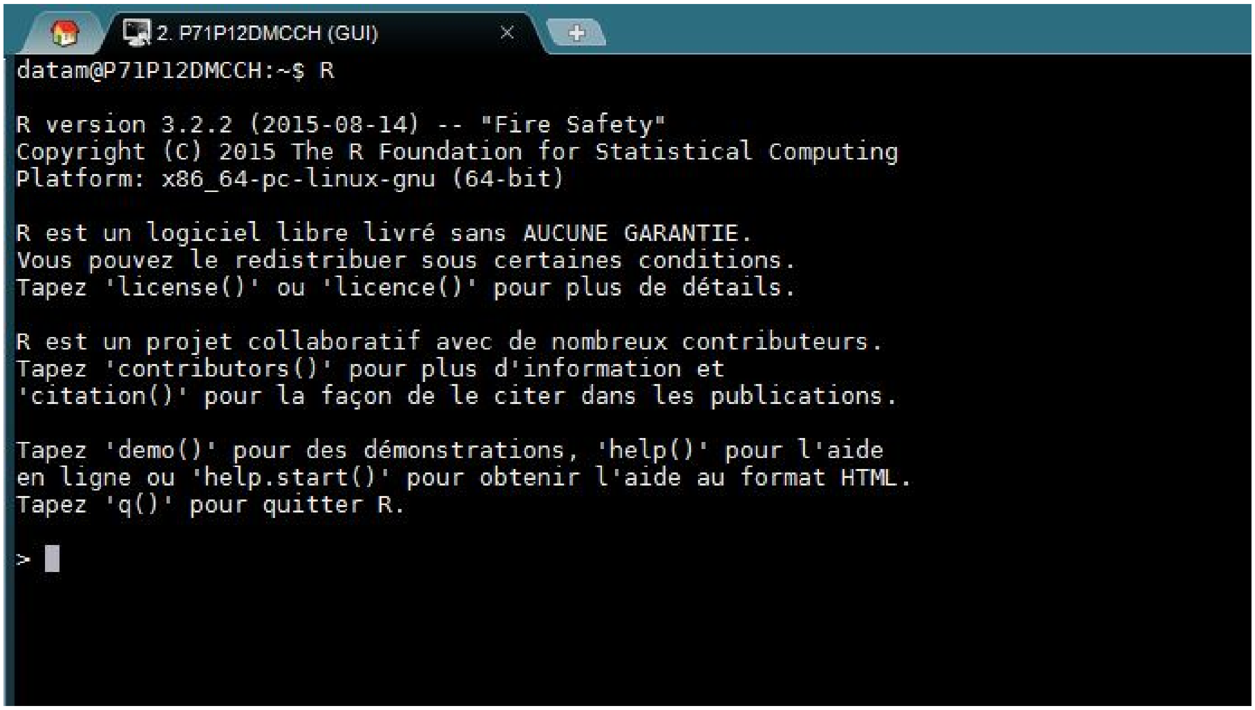


Figure 2: screenshot of interactive r session

RStudio

RStudio is an open-source **integraded development environment**(IDE) that provides an informative user-interface for interacting with the R software program (You can download the free version).

The RStudio Interface:

- **Script**: Text editor for writing your R code.
- **Console**: A console for entering R commands. This is very similar to your command-line. You can **use the up arrow** to easily access previously executed lines of code.
- **Environment**: displays information that you have stored inside of variables.
- **Files, Plots, Packages, Help etc.**: there are multiple tabs for accessing various information.

Remark: See **rstudio-IDE-cheatsheet** for detailed information on the IDE.

Creating Variables

In R, variable names can contain any combination of alphanumeric characters, as well as periods (`.`) or underscores (_) but *cannot* begin with a number, period, or underscore – they are also case sensitive.

When you are declaring a variable in R, you use the assignment operator `<-` to store information in a variable. For example:

```
# Stores the number 80 into a variable called monthly.transport.fee
monthly.transport.fee <- 80
```

Type the variable name into the R console and hit enter to see the information stored in a variable:

```
monthly.transport.fee
## [1] 80
```

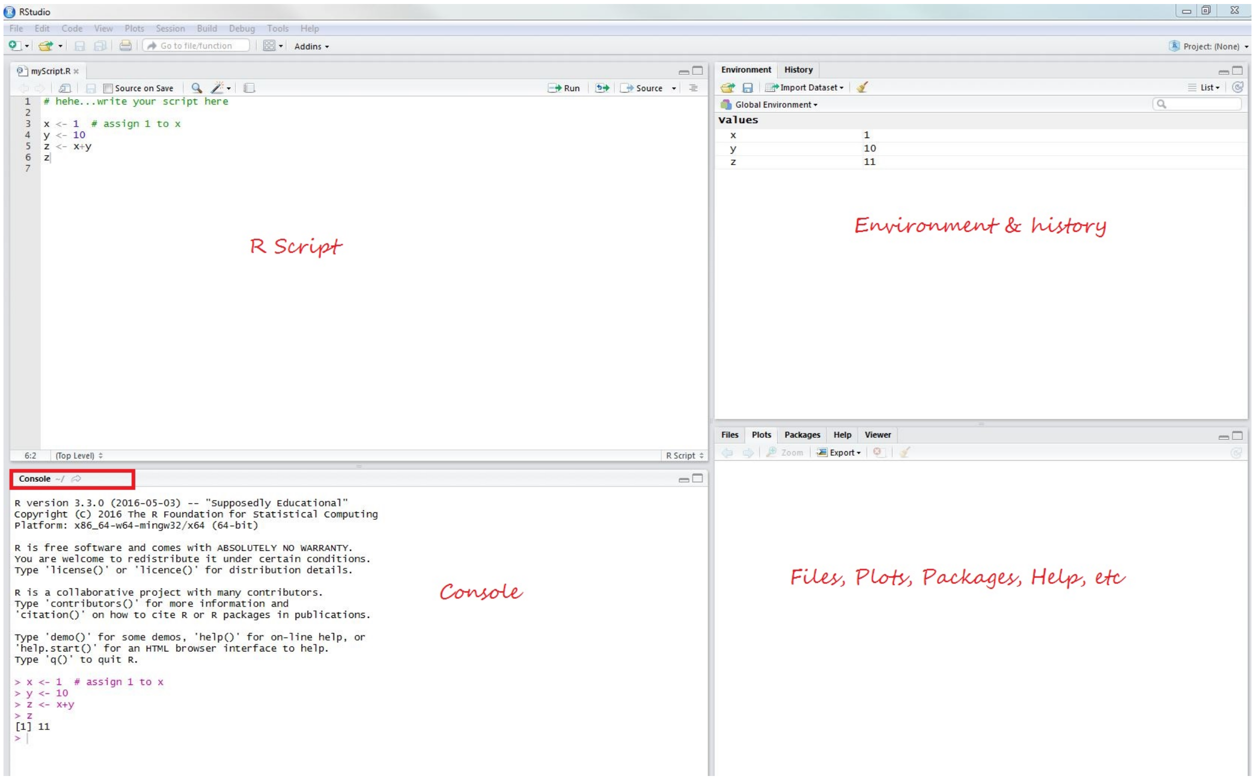


Figure 3: screenshot of labeled RStudio interface

You an also use print function

```
print(monthly.transport.fee)
    ## [1] 80
```

It's also quite easy to use **basic mathematical operators** (+, -, /, *) in your creation of variables. For example, you could create a variable that is the sum of two numbers as follows:

```
x <- 10 + 8
```

Other arithmetic operators include:

- Exponentiation: `^` : for example `3^2` equals 9.
- Modulo: `%%` : for example 5 modulo 3 or `5 %% 3` equals 2.

Workspace

- Use `ls()` to show contents in the workspace.
- Use `rm()` to remove contents in workspace.

```
# Clear the entire workspace
rm(list = ls())

# Create the variables milk, sugar and bread
milk <- 3
sugar <- 7
bread <- 1
```

```r
# Inspect the contents of the workspace again
ls()
```

```
## [1] "bread" "milk"  "sugar"
```

Basic Data Types

Basic data types in R:

- **Numeric**: Numeric data consists of the set of real numbers.

  ```r
  x <- 10.23 + 8
  ```

- **Character**: Character data stores strings of characters in a variable. Encapsulate the strings in either single (`'`) or double quotes (`"`):

  ```r
  # Create character variable `name.manager` with the value "Guillaume THERY"
  name.manager <- "Guillaume THERY"
  ```

- **Boolean**: Boolean (logical) data types can only take on two values: `TRUE` and `FALSE`.

```r
# Test if 3.15 is greater than pi, and store the results in a variable `x`
x <- 3.15 > pi # returns the boolean value TRUE since `pi=3.141593`
x
##[1] TRUE
```

- **Complex**: Complex numbers are created using the `i` syntax:

```r
complex.variable <- 4+5i # create a complex number with real=4 & imaginary=5
```

- **Integer**: Create an integer by placing a capital `L` after an integer value in variable assignment

```r
x.integer <- 4L
```

Package Management and Installation

Packages are used to extend the functionality of R

- Installing packages from CRAN:
  - One-by-one

    ```r
    install.packages("ggplot2")
    ```

  - Group

    ```r
    list.of.packages <- c("ggplot2","reshape2","plyr","data.table","dplyr","lubridate")
    new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
    if(length(new.packages)) install.packages(new.packages)
    ```

- Installing from Github:

  ```r
  require(devtools)
  install_github('ramnathv/rCharts')
  ```

- To load a package (need to load before using)

  ```r
  library("ggplot2")
  ```

- `all.packages <- available.packages()` to get the list of all the packages in CRAN

Data Imports, Preparation & Manipulation

Getting Started and Resource

```
?mean  # help on a function
help.search('kurtosis') # search help files for a word
help(package = 'ggplot2') # find help in package
search() # to list all of the functions now available to you
getwd() # to get the working directory
setwd() # to change the working directory
```

In getting started with R, RCheatSheet's will be provided as supporting documents to avoid repetitions in this training session. Participants can download such reference cards via RStudio website.

Remark: We will use the **Base R CheatSheet** to learn basic `R` syntax: Vectors, Matrices, Lists, Data Frames, Functions and etc.

How To Create A Simple Data Frame in R

```
Died.At <- c(22,40,72,41)
Writer.At <- c(16, 18, 36, 36)
First.Name <- I(c("Kevin", "Helen", "Walt", "Janet"))
Second.Name <- I(c("Doe", "Danse", "Whitman", "Austen"))
Sex <- c("MALE", "FEMALE", "MALE", "FEMALE")
Date.Of.Death <- as.Date(c("2015-05-10", "1949-10-07", "1992-03-26","1887-07-18"))
writers_df <- data.frame(Died.At, Writer.At, First.Name, Second.Name, Sex, Date.Of.Death)
head(writers_df) # to get the head in the data frame
```

```
##   Died.At Writer.At First.Name Second.Name    Sex Date.Of.Death
## 1      22        16      Kevin         Doe   MALE    2015-05-10
## 2      40        18      Helen       Danse FEMALE    1949-10-07
## 3      72        36       Walt     Whitman   MALE    1992-03-26
## 4      41        36      Janet      Austen FEMALE    1887-07-18
```

```
tail(writers_df) # to get the tail in the data frame
```

```
##   Died.At Writer.At First.Name Second.Name    Sex Date.Of.Death
## 1      22        16      Kevin         Doe   MALE    2015-05-10
## 2      40        18      Helen       Danse FEMALE    1949-10-07
## 3      72        36       Walt     Whitman   MALE    1992-03-26
## 4      41        36      Janet      Austen FEMALE    1887-07-18
```

Convert column `Date.Of.Death` as date format:

```
suppressMessages(library(lubridate))
```

```
## Warning: package 'lubridate' was built under R version 3.3.2
```

```
writers_df$Date.Of.Death <- ymd(writers_df$Date.Of.Death)
str(writers_df)
```

```
## 'data.frame':    4 obs. of  6 variables:
##  $ Died.At      : num  22 40 72 41
##  $ Writer.At    : num  16 18 36 36
##  $ First.Name   :Class 'AsIs'  chr [1:4] "Kevin" "Helen" "Walt" "Janet"
```

```
##  $ Second.Name  :Class 'AsIs'  chr [1:4] "Doe" "Danse" "Whitman" "Austen"
##  $ Sex          : Factor w/ 2 levels "FEMALE","MALE": 2 1 2 1
##  $ Date.Of.Death: Date, format: "2015-05-10" "1949-10-07" ...
```

What you need to know about Data Frames

We make use of the Airline Dataset. The data comes originally from RITA where it is described in detail.

```r
data <- read.csv("allyears2k.csv") # to read a CSV file into a data frame
names(data) # get names of variables / fields in the data frame
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay" "IsArrDelayed"
## [31] "IsDepDelayed"
```

```r
dim(data) # dimension of data frame
```

```
## [1] 43978    31
```

```r
nrow(data) # to get the number of rows in the data frame
```

```
## [1] 43978
```

```r
ncol(data) # to get the number of columns in the data frame
```

```
## [1] 31
```

```r
head(data,2) # to get the head of 2 rows in the data frame
```

```
##   Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
## 1 1987    10         14         3     741        730     912        849
## 2 1987    10         15         4     729        730     903        849
##   UniqueCarrier FlightNum TailNum ActualElapsedTime CRSElapsedTime AirTime
## 1            PS      1451    <NA>                91             79      NA
## 2            PS      1451    <NA>                94             79      NA
##   ArrDelay DepDelay Origin Dest Distance TaxiIn TaxiOut Cancelled
## 1       23       11    SAN  SFO      447     NA      NA         0
## 2       14       -1    SAN  SFO      447     NA      NA         0
##   CancellationCode Diverted CarrierDelay WeatherDelay NASDelay
## 1             <NA>        0           NA           NA       NA
## 2             <NA>        0           NA           NA       NA
##   SecurityDelay LateAircraftDelay IsArrDelayed IsDepDelayed
## 1            NA                NA          YES          YES
## 2            NA                NA          YES           NO
```

```r
tail(data,2) # to get the tail of 2 rows in the data frame
```

```
##       Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime
## 43977 2008     1          3         4    1652       1625    1750
## 43978 2008     1          3         4    1057       1030    1154
```

```
##       CRSArrTime UniqueCarrier FlightNum TailNum ActualElapsedTime
## 43977       1730            WN      1022  N655WN                58
## 43978       1140            WN      1041  N474WN                57
##       CRSElapsedTime AirTime ArrDelay DepDelay Origin Dest Distance TaxiIn
## 43977             65      44       20       27    SLC  BOI      291      4
## 43978             70      46       14       27    SLC  BOI      291      2
##       TaxiOut Cancelled CancellationCode Diverted CarrierDelay
## 43977      10         0                         0            0
## 43978       9         0                         0           NA
##       WeatherDelay NASDelay SecurityDelay LateAircraftDelay IsArrDelayed
## 43977            0        4             0                16          YES
## 43978           NA       NA            NA                NA          YES
##       IsDepDelayed
## 43977          YES
## 43978          YES
```

```r
str(data) # to get structure of the data frame
attributes(data) #  to see the data frame attributes
summary(data) # to see data summary
data$Year # to see the column Year...same as data[[1]]
data[1] # to create a new data frame with just column 1
data[1:4] # to create a data frame with first four columns
data[, 1] # to get column 1 listed
data$Year[20] # to get the 20th element in the Year column
tail(data,2) # to get the tail of 2 rows in the data frame
write.csv(data,"mydataname.csv", row.names = FALSE) # save a dataset in csv
```

- thus `data[,1][47]` is then the same as `data$Year[47]`
- `length(data$IsArrDelayed[is.na(data$IsArrDelayed)])` to find the number of NAs in a given column (IsArrDelayed) in a data frame

```r
# to get the mean of the valid numbers in the group
mean(data$Cancelled[!is.na(data$Cancelled)])
```

```
## [1] 0.02469417
```

```r
# to get the mean value of Departure Delay where Arrival Delay is > 10 and Day Of Week is Monday
mean(subset(data, !is.na(data$CarrierDelay) & data$ArrDelay > 10 & data$DayOfWeek == 1)$DepDelay)
```

```
## [1] 45.98084
```

Faster Data Manipulation in R using dplyr

Introduction

- Five basic verbs: `filter`, `select`, `arrange`, `mutate`, `summarise` (plus `group_by`)
- dplyr will mask a few base functions
- The data on Régularité mensuelle TGV depuis septembre 2011. The data can be download from the SNCF Open data platform. Click on download for data.

```r
# load packages
suppressMessages(library(dplyr))
```

```
## Warning: package 'dplyr' was built under R version 3.3.2
```

```r
suppressMessages(library(lubridate))

# import data from current working directory (eg data in the folder "opendatasncf")

regularite.mensuelle.tgv <- read.csv("opendatasncf/regularite-mensuelle-tgv.csv",
                                     encoding="UTF-8", sep=";")

# view data

head(regularite.mensuelle.tgv)
```

```
##      Date        Axe                     Départ             Arrivée
## 1 2014-08 Atlantique          ANGERS SAINT LAUD   PARIS MONTPARNASSE
## 2 2014-08 Atlantique                  ANGOULEME   PARIS MONTPARNASSE
## 3 2014-08    Sud-Est                     ANNECY           PARIS LYON
## 4 2014-08    Sud-Est                AVIGNON TGV           PARIS LYON
## 5 2014-08    Sud-Est BESANCON FRANCHE COMTE TGV           PARIS LYON
## 6 2014-08       Nord                      LILLE MARSEILLE ST CHARLES
##   Nombre.de.trains.programmés Nombre.de.trains.ayant.circulé
## 1                         440                            440
## 2                         327                            327
## 3                         124                            124
## 4                         524                            524
## 5                         214                            214
## 6                         124                            124
##   Nombre.de.trains.annulés Nombre.de.trains.en.retard.à.l.arrivée
## 1                        0                                     21
## 2                        0                                     31
## 3                        0                                      5
## 4                        0                                     89
## 5                        0                                      9
## 6                        0                                     13
##   Régularité
## 1       95.2
## 2       90.5
## 3       96.0
## 4       83.0
## 5       95.8
## 6       89.5
##
## 1
## 2
## 3
## 4 En août, la régularité des TGV entre Paris et la Méditerranée a été fortement impactée par 8 accide
## 5
## 6
```

- `tbl_df` creates a "local data frame"
- Local data frame is simply a wrapper for a data frame that prints nicely

```r
# convert to local data frame

regularite <- tbl_df(regularite.mensuelle.tgv)

# rename columns by removing french accents
```

```r
colnames(regularite)<-c("Date","Axe","Depart","Arrivee","Nombre.de.trains.programmes",
                        "Nombre.de.trains.ayant.circule","Nombre.de.trains.annules",
                        "Nombre.de.trains.en.retard.a.l.arrivee","Regularite","Commentaires")

# printing only shows 10 rows and as many columns as can fit on your screen
regularite
```

```
## # A tibble: 5,900 × 10
##       Date        Axe                     Depart              Arrivee
##     <fctr>     <fctr>                     <fctr>               <fctr>
## 1  2014-08 Atlantique          ANGERS SAINT LAUD   PARIS MONTPARNASSE
## 2  2014-08 Atlantique                  ANGOULEME   PARIS MONTPARNASSE
## 3  2014-08    Sud-Est                     ANNECY           PARIS LYON
## 4  2014-08    Sud-Est                AVIGNON TGV           PARIS LYON
## 5  2014-08    Sud-Est BESANCON FRANCHE COMTE TGV           PARIS LYON
## 6  2014-08       Nord                      LILLE MARSEILLE ST CHARLES
## 7  2014-08    Sud-Est             LYON PART DIEU          MONTPELLIER
## 8  2014-08    Sud-Est                MACON LOCHE           PARIS LYON
## 9  2014-08    Sud-Est             MULHOUSE VILLE           PARIS LYON
## 10 2014-08        Est                  PARIS EST           STRASBOURG
## # ... with 5,890 more rows, and 6 more variables:
## #   Nombre.de.trains.programmes <int>,
## #   Nombre.de.trains.ayant.circule <int>, Nombre.de.trains.annules <int>,
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>, Regularite <dbl>,
## #   Commentaires <fctr>
```

```r
# you can change Date in date format
regularite$Date1 <- paste0(regularite$Date,sep='-',01) # add a day part of date
regularite$Date2 <- ymd(regularite$Date1) # date column
regularite$Year <- year(regularite$Date2)
regularite$Month <- month(regularite$Date2)
regularite$Quarters <- quarters(regularite$Date2)

# you can specify that you want to see more rows
print(regularite, n=20)

# convert to a normal data frame to see all of the columns
data.frame(head(regularite))
```

filter: Keep rows matching criteria

- Base R approach to filtering forces you to repeat the data frame's name
- dplyr approach is simpler to write and read
- Command structure (for all dplyr verbs):
  - first argument is a data frame
  - return value is a data frame
  - nothing is modified in place

```r
# base R approach to view all regularite on Quarter 1 and January
regularite[regularite$Quarters=='Q1' & regularite$Month==1, ]
#another method
subset(regularite,regularite$Quarters=='Q1' & regularite$Month==1)
subset(regularite,Quarters=='Q1' & Month==1)
```

```r
# dplyr approach
# note: you can use comma or ampersand to represent AND condition
filter(regularite, Quarters=='Q1', Month==1)
```

```
## # A tibble: 500 × 15
##       Date     Axe               Depart                     Arrivee
##     <fctr>  <fctr>               <fctr>                      <fctr>
## 1  2015-01    Nord                DOUAI                  PARIS NORD
## 2  2015-01 Sud-Est          MACON LOCHE                  PARIS LYON
## 3  2015-01 Sud-Est MARSEILLE ST CHARLES              LYON PART DIEU
## 4  2015-01     Est                NANCY                   PARIS EST
## 5  2015-01 Sud-Est           NICE VILLE                  PARIS LYON
## 6  2015-01 Sud-Est           PARIS LYON BESANCON FRANCHE COMTE TGV
## 7  2012-01 Sud-Est  AIX EN PROVENCE TGV                  PARIS LYON
## 8  2012-01 Sud-Est          DIJON VILLE                  PARIS LYON
## 9  2012-01    Nord           PARIS NORD                       DOUAI
## 10 2012-01 Sud-Est           PARIS LYON                    GRENOBLE
## # ... with 490 more rows, and 11 more variables:
## #   Nombre.de.trains.programmes <int>,
## #   Nombre.de.trains.ayant.circule <int>, Nombre.de.trains.annules <int>,
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>, Regularite <dbl>,
## #   Commentaires <fctr>, Date1 <chr>, Date2 <date>, Year <dbl>,
## #   Month <dbl>, Quarters <chr>
```

```r
# use pipe for OR condition
filter(regularite, Depart=="LYON PART DIEU" | Depart=="PARIS NORD")
```

```
## # A tibble: 531 × 15
##       Date     Axe         Depart              Arrivee
##     <fctr>  <fctr>         <fctr>               <fctr>
## 1  2014-08 Sud-Est LYON PART DIEU          MONTPELLIER
## 2  2014-10 Sud-Est LYON PART DIEU MARSEILLE ST CHARLES
## 3  2014-10    Nord     PARIS NORD                DOUAI
## 4  2014-12    Nord     PARIS NORD                DOUAI
## 5  2015-06    Nord LYON PART DIEU                LILLE
## 6  2015-07    Nord     PARIS NORD                DOUAI
## 7  2015-07    Nord     PARIS NORD            DUNKERQUE
## 8  2015-08 Sud-Est LYON PART DIEU MARSEILLE ST CHARLES
## 9  2015-08    Nord     PARIS NORD                DOUAI
## 10 2015-09    Nord     PARIS NORD                DOUAI
## # ... with 521 more rows, and 11 more variables:
## #   Nombre.de.trains.programmes <int>,
## #   Nombre.de.trains.ayant.circule <int>, Nombre.de.trains.annules <int>,
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>, Regularite <dbl>,
## #   Commentaires <fctr>, Date1 <chr>, Date2 <date>, Year <dbl>,
## #   Month <dbl>, Quarters <chr>
```

```r
# you can also use %in% operator
filter(regularite, Depart %in% c("LYON PART DIEU", "PARIS NORD"))
```

select: Pick columns by name

- Base R approach is awkward to type and to read
- dplyr approach uses similar syntax to `filter`

- Like a SELECT in SQL

```r
# base R approach to select Depart, Arrivee, and Nombre.de.trains.programmes columns
regularite[, c("Depart", "Arrivee", "Nombre.de.trains.programmes")]
```

```r
# dplyr approach
select(regularite, Depart, Arrivee, Nombre.de.trains.programmes)
```

```
## # A tibble: 5,900 × 3
##                           Depart               Arrivee
##                           <fctr>                <fctr>
## 1            ANGERS SAINT LAUD   PARIS MONTPARNASSE
## 2                    ANGOULEME   PARIS MONTPARNASSE
## 3                       ANNECY           PARIS LYON
## 4                  AVIGNON TGV           PARIS LYON
## 5  BESANCON FRANCHE COMTE TGV           PARIS LYON
## 6                        LILLE MARSEILLE ST CHARLES
## 7               LYON PART DIEU          MONTPELLIER
## 8                  MACON LOCHE           PARIS LYON
## 9               MULHOUSE VILLE           PARIS LYON
## 10                   PARIS EST           STRASBOURG
## # ... with 5,890 more rows, and 1 more variables:
## #   Nombre.de.trains.programmes <int>
```

```r
# use colon to select multiple contiguous columns, and use `contains` to match columns by name
# note: `starts_with`, `ends_with`, and `matches` (for regular expressions)
# can also be used to match columns by name
select(regularite, Year:Month, contains("annules"), contains("retard"))
```

```
## # A tibble: 5,900 × 4
##     Year Month Nombre.de.trains.annules
##    <dbl> <dbl>                    <int>
## 1   2014     8                        0
## 2   2014     8                        0
## 3   2014     8                        0
## 4   2014     8                        0
## 5   2014     8                        0
## 6   2014     8                        0
## 7   2014     8                        0
## 8   2014     8                        0
## 9   2014     8                        0
## 10  2014     8                        0
## # ... with 5,890 more rows, and 1 more variables:
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>
```

Chaining Method (%>%: "then" operator)

- Usual way to perform multiple operations in one line is by nesting
- Chaining increases readability significantly when there are many commands

```r
# nesting method to select Depart and Nombre.de.trains.en.retard.a.l.arrivee columns
# and filter for delays over 60 minutes
filter(select(regularite, Depart, Nombre.de.trains.en.retard.a.l.arrivee),
       Nombre.de.trains.en.retard.a.l.arrivee > 60)
```

```r
# chaining method
regularite %>%
    select(Depart, Nombre.de.trains.en.retard.a.l.arrivee) %>%
    filter(Nombre.de.trains.en.retard.a.l.arrivee > 60)
```

```
## # A tibble: 662 × 2
##                   Depart Nombre.de.trains.en.retard.a.l.arrivee
##                   <fctr>                                  <int>
## 1            AVIGNON TGV                                     89
## 2         LYON PART DIEU                                     75
## 3  MARSEILLE ST CHARLES                                     105
## 4         LYON PART DIEU                                    147
## 5   AIX EN PROVENCE TGV                                      69
## 6            AVIGNON TGV                                     77
## 7  MARSEILLE ST CHARLES                                     155
## 8            MONTPELLIER                                     79
## 9   ST PIERRE DES CORPS                                      73
## 10 MARSEILLE ST CHARLES                                      79
## # ... with 652 more rows
```

- Operator is automatically imported from the magrittr package
- Can be used to replace nesting in R commands outside of dplyr

```r
# create two vectors and calculate Euclidian distance between them
a <- 3:7; b <- 1:5
sqrt(sum((a-b)^2))
```

```r
# chaining method
(a-b)^2 %>% sum() %>% sqrt()
```

```
## [1] 4.472136
```

arrange: Reorder rows

```r
# base R approach to select Depart and Nombre.de.trains.en.retard.a.l.arrivee columns and sort by
# Nombre.de.trains.en.retard.a.l.arrivee
regularite[order(regularite$Nombre.de.trains.en.retard.a.l.arrivee),
           c("Depart", "Nombre.de.trains.en.retard.a.l.arrivee")]
```

```r
# dplyr approach
regularite %>%
    select(Depart, Nombre.de.trains.en.retard.a.l.arrivee) %>%
    arrange(Nombre.de.trains.en.retard.a.l.arrivee)
```

```
## # A tibble: 5,900 × 2
##                         Depart Nombre.de.trains.en.retard.a.l.arrivee
##                         <fctr>                                  <int>
## 1                   PARIS LYON                                      0
## 2  SAINT ETIENNE CHATEAUCREUX                                       0
## 3           PARIS MONTPARNASSE                                      0
## 4  SAINT ETIENNE CHATEAUCREUX                                       0
## 5                   PARIS LYON                                      0
## 6  SAINT ETIENNE CHATEAUCREUX                                       0
## 7  SAINT ETIENNE CHATEAUCREUX                                       0
## 8                   PARIS LYON                                      0
```

```
## 9                  PARIS LYON                                        0
## 10                 PARIS LYON                                        0
## # ... with 5,890 more rows
```

```
# use `desc` for descending
regularite %>%
    select(Depart, Nombre.de.trains.en.retard.a.l.arrivee) %>%
    arrange(desc(Nombre.de.trains.en.retard.a.l.arrivee))
```

mutate: Add new variables

- Create new variables that are functions of existing variables

```
# base R approach to create a new variable Non.Regularite (in %)
regularite$Non.Regularite <- 100-regularite$Regularite
regularite[, c("Regularite", "Arrivee", "Non.Regularite")]
```

```
# dplyr approach (prints the new variable but does not store it)
regularite %>%
    select(Regularite, Arrivee) %>%
    mutate(Non.Regularite = 100-Regularite)
```

```
## # A tibble: 5,900 × 3
##    Regularite              Arrivee Non.Regularite
##         <dbl>               <fctr>          <dbl>
## 1        95.2   PARIS MONTPARNASSE            4.8
## 2        90.5   PARIS MONTPARNASSE            9.5
## 3        96.0           PARIS LYON            4.0
## 4        83.0           PARIS LYON           17.0
## 5        95.8           PARIS LYON            4.2
## 6        89.5 MARSEILLE ST CHARLES           10.5
## 7        81.1          MONTPELLIER           18.9
## 8        86.7           PARIS LYON           13.3
## 9        93.4           PARIS LYON            6.6
## 10       93.6           STRASBOURG            6.4
## # ... with 5,890 more rows
```

```
# store the new variable
regularite <- regularite %>% mutate(Non.Regularite = 100-Regularite)
```

summarise: Reduce variables to values

- Useful with data that has been grouped by one or more variables
- `group_by` creates the groups that will be operated on
- `summarise` uses the provided aggregation function to summarise each group

```
# base R approaches to calculate the average arrival delay to each destination
head(with(regularite, tapply(Nombre.de.trains.en.retard.a.l.arrivee, Arrivee, mean, na.rm=TRUE)))
head(aggregate(Nombre.de.trains.en.retard.a.l.arrivee ~ Arrivee, regularite, mean))
```

```
# dplyr approach: create a table grouped by Arrivee, and then summarise each group
# by taking the mean of Nombre.de.trains.en.retard.a.l.arrivee
regularite %>%
    group_by(Arrivee) %>%
    summarise(avg_delay = mean(Nombre.de.trains.en.retard.a.l.arrivee, na.rm=TRUE))
```

```
## # A tibble: 48 × 2
##                        Arrivee avg_delay
##                         <fctr>     <dbl>
## 1         AIX EN PROVENCE TGV  43.54237
## 2           ANGERS SAINT LAUD  31.03390
## 3                   ANGOULEME  23.44068
## 4                      ANNECY  11.88136
## 5                       ARRAS  31.11864
## 6                 AVIGNON TGV  40.94915
## 7            BELLEGARDE (AIN)  30.47458
## 8  BESANCON FRANCHE COMTE TGV  15.79661
## 9            BORDEAUX ST JEAN  50.30508
## 10                      BREST  11.52542
## # ... with 38 more rows
```

- `summarise_each` allows you to apply the same summary function to multiple columns at once
- Note: `mutate_each` is also available

```r
# for each Depart, calculate the mean of Nombre.de.trains.annules or
# Nombre.de.trains.en.retard.a.l.arrivee
regularite %>%
    group_by(Depart) %>%
    summarise_each(funs(mean), Nombre.de.trains.annules,
                   Nombre.de.trains.en.retard.a.l.arrivee)
```

```
## # A tibble: 48 × 3
##                         Depart Nombre.de.trains.annules
##                         <fctr>                    <dbl>
## 1         AIX EN PROVENCE TGV                 0.2711864
## 2           ANGERS SAINT LAUD                 2.0508475
## 3                   ANGOULEME                 2.1525424
## 4                      ANNECY                 1.0847458
## 5                       ARRAS                 1.9661017
## 6                 AVIGNON TGV                 1.9830508
## 7            BELLEGARDE (AIN)                 0.8135593
## 8  BESANCON FRANCHE COMTE TGV                 0.7457627
## 9            BORDEAUX ST JEAN                 3.3050847
## 10                      BREST                 0.4915254
## # ... with 38 more rows, and 1 more variables:
## #   Nombre.de.trains.en.retard.a.l.arrivee <dbl>
```

```r
# for each Depart, calculate the minimum and maximum arrival delays
regularite %>%
    group_by(Depart) %>%
    summarise_each(funs(min(., na.rm=TRUE), max(., na.rm=TRUE)),
                   matches("Nombre.de.trains.en.retard.a.l.arrivee"))
```

```
## # A tibble: 48 × 3
##                         Depart   min   max
##                         <fctr> <int> <int>
## 1         AIX EN PROVENCE TGV      0    95
## 2           ANGERS SAINT LAUD     16   115
## 3                   ANGOULEME     14   102
## 4                      ANNECY      5    24
## 5                       ARRAS     19   109
```

```
## 6                  AVIGNON TGV    24   140
## 7             BELLEGARDE (AIN)     6    84
## 8  BESANCON FRANCHE COMTE TGV     2    32
## 9             BORDEAUX ST JEAN    34   140
## 10                       BREST     2    29
## # ... with 38 more rows
```

- Helper function `n()` counts the number of rows in a group
- Helper function `n_distinct(vector)` counts the number of unique items in that vector

```
# for each month of the year, count the total number of rows and sort in descending order
regularite %>%
    group_by(Year, Month) %>%
    summarise(row_count = n()) %>%
    arrange(desc(row_count))
```

```
## Source: local data frame [59 x 3]
## Groups: Year [6]
##
##     Year Month row_count
##    <dbl> <dbl>     <int>
## 1   2011     9       100
## 2   2011    10       100
## 3   2011    11       100
## 4   2011    12       100
## 5   2012     1       100
## 6   2012     2       100
## 7   2012     3       100
## 8   2012     4       100
## 9   2012     5       100
## 10  2012     6       100
## # ... with 49 more rows
```

```
# rewrite more simply with the `tally` function
regularite %>%
    group_by(Year, Month) %>%
    tally(sort = TRUE)
```

```
## Source: local data frame [59 x 3]
## Groups: Year [6]
##
##     Year Month     n
##    <dbl> <dbl> <int>
## 1   2011     9   100
## 2   2011    10   100
## 3   2011    11   100
## 4   2011    12   100
## 5   2012     1   100
## 6   2012     2   100
## 7   2012     3   100
## 8   2012     4   100
## 9   2012     5   100
## 10  2012     6   100
## # ... with 49 more rows
```

```r
# for each Axe, count the total number of distinct destination
regularite %>%
    group_by(Axe) %>%
    summarise(Arrivee_count = n_distinct(Arrivee))
```

```
## # A tibble: 4 × 2
##          Axe Arrivee_count
##       <fctr>         <int>
## 1 Atlantique            18
## 2        Est             8
## 3       Nord             7
## 4    Sud-Est            21
```

Window Functions

- Aggregation function (like `mean`) takes n inputs and returns 1 value
- Window function takes n inputs and returns n values
  - Includes ranking and ordering functions (like `min_rank`), offset functions (`lead` and `lag`), and cumulative aggregates (like `cummean`).

```r
# for each Departure, calculate which three months of the year they had their
# longest departure delays
# note: smallest (not largest) value is ranked as 1, so you have to
# use `desc` to rank by largest value
regularite %>%
    group_by(Depart) %>%
    select(Quarters, Month, Nombre.de.trains.en.retard.a.l.arrivee) %>%
    filter(min_rank(desc(Nombre.de.trains.en.retard.a.l.arrivee)) <= 3) %>%
    arrange(Depart, desc(Nombre.de.trains.en.retard.a.l.arrivee))
```

```
## Adding missing grouping variables: `Depart`
```

```r
# rewrite more simply with the `top_n` function
regularite %>%
    group_by(Depart) %>%
    select(Quarters, Month, Nombre.de.trains.en.retard.a.l.arrivee) %>%
    top_n(3) %>%
    arrange(Depart, desc(Nombre.de.trains.en.retard.a.l.arrivee))
```

```
## Adding missing grouping variables: `Depart`
```

```
## Selecting by Nombre.de.trains.en.retard.a.l.arrivee
```

```
## Source: local data frame [151 x 4]
## Groups: Depart [48]
##
##                 Depart Quarters Month
##                 <fctr>    <chr> <dbl>
## 1  AIX EN PROVENCE TGV       Q3     7
## 2  AIX EN PROVENCE TGV       Q2     5
## 3  AIX EN PROVENCE TGV       Q4    11
## 4    ANGERS SAINT LAUD       Q4    11
## 5    ANGERS SAINT LAUD       Q3     7
## 6    ANGERS SAINT LAUD       Q4    10
## 7            ANGOULEME       Q1     2
## 8            ANGOULEME       Q4    11
```

```
## 9            ANGOULEME       Q3     7
## 10           ANGOULEME       Q4    10
## # ... with 141 more rows, and 1 more variables:
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>
```

```r
# for each Quarter, calculate the number of observations and the change from the previous Quarter
regularite %>%
    group_by(Quarters) %>%
    summarise(row_count = n()) %>%
    mutate(change = row_count - lag(row_count))
```

```
## # A tibble: 4 × 3
##   Quarters row_count change
##      <chr>     <int>  <int>
## 1       Q1      1500     NA
## 2       Q2      1500      0
## 3       Q3      1400   -100
## 4       Q4      1500    100
```

```r
# rewrite more simply with the `tally` function
regularite %>%
    group_by(Quarters) %>%
    tally() %>%
    mutate(change = n - lag(n))
```

```
## # A tibble: 4 × 3
##   Quarters     n change
##      <chr> <int>  <int>
## 1       Q1  1500     NA
## 2       Q2  1500      0
## 3       Q3  1400   -100
## 4       Q4  1500    100
```

Some Useful Functions

```r
# randomly sample a fixed number of rows, without replacement
regularite %>% sample_n(5)
```

```
## # A tibble: 5 × 16
##      Date     Axe                       Depart     Arrivee
##    <fctr>  <fctr>                       <fctr>      <fctr>
## 1 2012-11 Sud-Est CHAMBERY CHALLES LES EAUX  PARIS LYON
## 2 2014-02 Sud-Est                NICE VILLE  PARIS LYON
## 3 2012-10     Est                STRASBOURG      NANTES
## 4 2016-03     Est                 PARIS EST       NANCY
## 5 2012-07 Sud-Est                PARIS LYON AVIGNON TGV
## # ... with 12 more variables: Nombre.de.trains.programmes <int>,
## #   Nombre.de.trains.ayant.circule <int>, Nombre.de.trains.annules <int>,
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>, Regularite <dbl>,
## #   Commentaires <fctr>, Date1 <chr>, Date2 <date>, Year <dbl>,
## #   Month <dbl>, Quarters <chr>, Non.Regularite <dbl>
```

```r
# randomly sample a fraction of rows, with replacement
regularite %>% sample_frac(0.25, replace=TRUE)
```

```
## # A tibble: 1,475 × 16
##       Date       Axe                       Depart             Arrivee
##     <fctr>    <fctr>                       <fctr>              <fctr>
## 1  2015-08      Nord       MARSEILLE ST CHARLES               LILLE
## 2  2012-05   Sud-Est                  PERPIGNAN          PARIS LYON
## 3  2012-05      Nord       MARSEILLE ST CHARLES               LILLE
## 4  2012-02   Sud-Est SAINT ETIENNE CHATEAUCREUX          PARIS LYON
## 5  2014-10 Atlantique                 ANGOULEME  PARIS MONTPARNASSE
## 6  2012-02   Sud-Est                      NIMES          PARIS LYON
## 7  2014-01   Sud-Est                 PARIS LYON AIX EN PROVENCE TGV
## 8  2013-02       Est                  PARIS EST               NANCY
## 9  2013-06   Sud-Est       MARSEILLE ST CHARLES          PARIS LYON
## 10 2013-08   Sud-Est                     ANNECY          PARIS LYON
## # ... with 1,465 more rows, and 12 more variables:
## #   Nombre.de.trains.programmes <int>,
## #   Nombre.de.trains.ayant.circule <int>, Nombre.de.trains.annules <int>,
## #   Nombre.de.trains.en.retard.a.l.arrivee <int>, Regularite <dbl>,
## #   Commentaires <fctr>, Date1 <chr>, Date2 <date>, Year <dbl>,
## #   Month <dbl>, Quarters <chr>, Non.Regularite <dbl>
```

```r
# base R approach to view the structure of an object
str(regularite)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    5900 obs. of  16 variables:
##  $ Date                                 : Factor w/ 59 levels "2011-09","2011-10",..: 36 36 36 36 3
##  $ Axe                                  : Factor w/ 4 levels "Atlantique","Est",..: 1 1 4 4 4 3 4 4
##  $ Depart                               : Factor w/ 48 levels "AIX EN PROVENCE TGV",..: 2 3 4 6 8 2
##  $ Arrivee                              : Factor w/ 48 levels "AIX EN PROVENCE TGV",..: 33 33 32 32
##  $ Nombre.de.trains.programmes          : int  440 327 124 524 214 124 397 188 318 440 ...
##  $ Nombre.de.trains.ayant.circule       : int  440 327 124 524 214 124 397 188 318 440 ...
##  $ Nombre.de.trains.annules             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Nombre.de.trains.en.retard.a.l.arrivee: int  21 31 5 89 9 13 75 25 21 28 ...
##  $ Regularite                           : num  95.2 90.5 96 83 95.8 89.5 81.1 86.7 93.4 93.6 ...
##  $ Commentaires                         : Factor w/ 437 levels "","\"Les ralentissements pour trava
##  $ Date1                                : chr  "2014-08-1" "2014-08-1" "2014-08-1" "2014-08-1" ...
##  $ Date2                                : Date, format: "2014-08-01" "2014-08-01" ...
##  $ Year                                 : num  2014 2014 2014 2014 2014 ...
##  $ Month                                : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ Quarters                             : chr  "Q3" "Q3" "Q3" "Q3" ...
##  $ Non.Regularite                       : num  4.8 9.5 4 17 4.2 10.5 18.9 13.3 6.6 6.4 ...
```

```r
# dplyr approach: better formatting, and adapts to your screen width
glimpse(regularite)
```

```
## Observations: 5,900
## Variables: 16
## $ Date                                  <fctr> 2014-08, 2014-08, 2014...
## $ Axe                                   <fctr> Atlantique, Atlantique...
## $ Depart                                <fctr> ANGERS SAINT LAUD, ANG...
## $ Arrivee                               <fctr> PARIS MONTPARNASSE, PA...
## $ Nombre.de.trains.programmes           <int> 440, 327, 124, 524, 214...
## $ Nombre.de.trains.ayant.circule        <int> 440, 327, 124, 524, 214...
## $ Nombre.de.trains.annules              <int> 0, 0, 0, 0, 0, 0, 0, 0,...
## $ Nombre.de.trains.en.retard.a.l.arrivee <int> 21, 31, 5, 89, 9, 13, 7...
## $ Regularite                            <dbl> 95.2, 90.5, 96.0, 83.0,...
## $ Commentaires                          <fctr> , , , En août, la régu...
```

```
## $ Date1                                   <chr> "2014-08-1", "2014-08-1...
## $ Date2                                   <date> 2014-08-01, 2014-08-01...
## $ Year                                    <dbl> 2014, 2014, 2014, 2014,...
## $ Month                                   <dbl> 8, 8, 8, 8, 8, 8, 8, 8,...
## $ Quarters                                <chr> "Q3", "Q3", "Q3", "Q3",...
## $ Non.Regularite                          <dbl> 4.8, 9.5, 4.0, 17.0, 4....
```

Descriptive Statistics, EDA & Statistical testing

Descriptive Statistics

Let us play with the Airline Dataset previously loaded in R

```r
names(data) # Airline dataset
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay" "IsArrDelayed"
## [31] "IsDepDelayed"
```

Contingency table

```r
# Create new column by grouping Departure Delay
data$groupDepDelay<- with(data,cut(DepDelay, quantile(DepDelay,na.rm=T)))
# Create contingency table
my.table0<-with(data, table(groupDepDelay,DayOfWeek))
prop.table(my.table0,1)
```

```
##               DayOfWeek
## groupDepDelay          1          2          3          4          5
##      (-16,-2] 0.12530255 0.16198101 0.20350028 0.15686092 0.11320052
##      (-2,1]   0.14458502 0.17561868 0.14848413 0.15548659 0.12660142
##      (1,10]   0.13160541 0.16836108 0.14813987 0.18221428 0.13719138
##      (10,473] 0.12296547 0.15570609 0.13801863 0.24912974 0.13322043
##               DayOfWeek
## groupDepDelay          6          7
##      (-16,-2] 0.12297524 0.11617948
##      (-2,1]   0.12158829 0.12763587
##      (1,10]   0.10546308 0.12702491
##      (10,473] 0.08918995 0.11176969
```

```r
# Alternative Approach ... direct
# to build a contingency table of the counts at each combination of factor levels
with(data,table(DayOfWeek,IsArrDelayed)) # same as table(data$DayOfWeek,data$IsArrDelayed)
```

```
##          IsArrDelayed
```

```
## DayOfWeek   NO  YES
##         1 2559 3243
##         2 3421 3882
##         3 3285 3746
##         4 3188 4921
##         5 2250 3339
##         6 2366 2501
##         7 2468 2809
```

```r
# to get probabilities of Depture Delay on each Day of Week
my.table<-with(data, table(cut(DepDelay, quantile(DepDelay,na.rm=T)),DayOfWeek))
prop.table(my.table,1)
```

```
##           DayOfWeek
##                     1          2          3          4          5
##   (-16,-2] 0.12530255 0.16198101 0.20350028 0.15686092 0.11320052
##   (-2,1]   0.14458502 0.17561868 0.14848413 0.15548659 0.12660142
##   (1,10]   0.13160541 0.16836108 0.14813987 0.18221428 0.13719138
##   (10,473] 0.12296547 0.15570609 0.13801863 0.24912974 0.13322043
##           DayOfWeek
##                     6          7
##   (-16,-2] 0.12297524 0.11617948
##   (-2,1]   0.12158829 0.12763587
##   (1,10]   0.10546308 0.12702491
##   (10,473] 0.08918995 0.11176969
```

Balloon Plot

```r
suppressMessages(library(gplots))
```

```
## Warning: package 'gplots' was built under R version 3.3.1
```

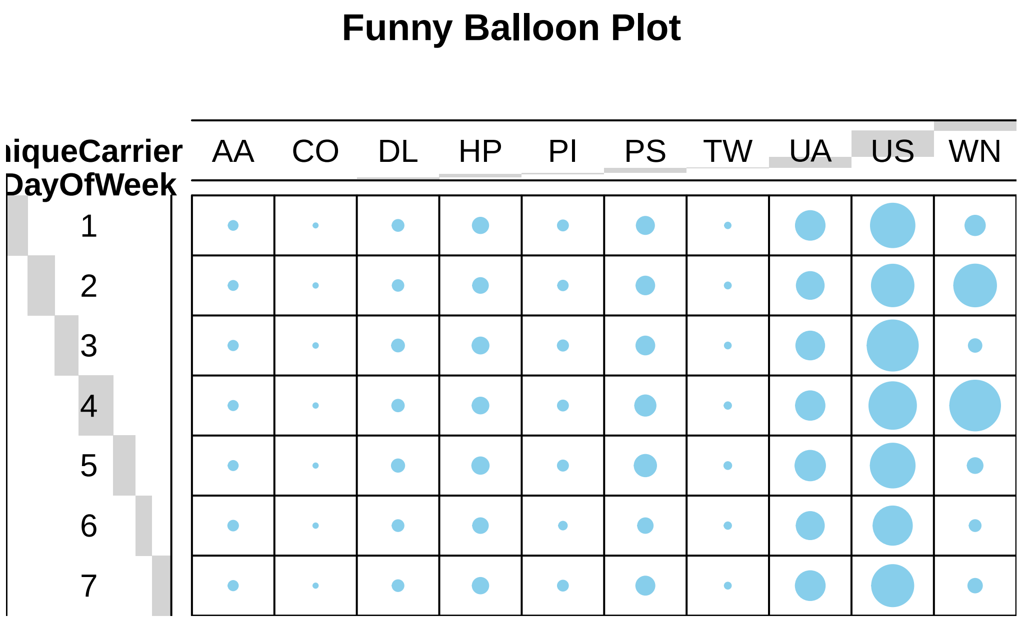
```r
suppressMessages(library(tidyr))
```

```
## Warning: package 'tidyr' was built under R version 3.3.2
```

```r
suppressMessages(library(dplyr))
## Checks on departure delays
data.filter4 <- data %>% filter(IsDepDelayed == 'YES')
data.select <- data.filter4 %>% select(DayOfWeek,UniqueCarrier)
tab<- (table(data.select))
tab
```

```
##          UniqueCarrier
## DayOfWeek   AA   CO   DL   HP   PI   PS   TW   UA   US   WN
##         1   58   12   88  175   74  219   22  617 1427  280
##         2   60   14   83  162   67  234   26  540 1303 1315
##         3   64   15  105  190   77  238   25  581 1897  117
##         4   62   14   98  187   74  307   30  607 1634 1870
##         5   61   13  112  200   75  341   34  653 1448  163
##         6   69   14   87  160   43  157   30  552 1103   88
##         7   63   13   86  178   73  244   26  625 1276  136
```

```r
# Plot a graphical matrix
balloonplot(t(tab), main ="Funny Balloon Plot", ylab ="DayOfWeek", xlab="UniqueCarrier",
         label = FALSE, show.margins = FALSE)
```

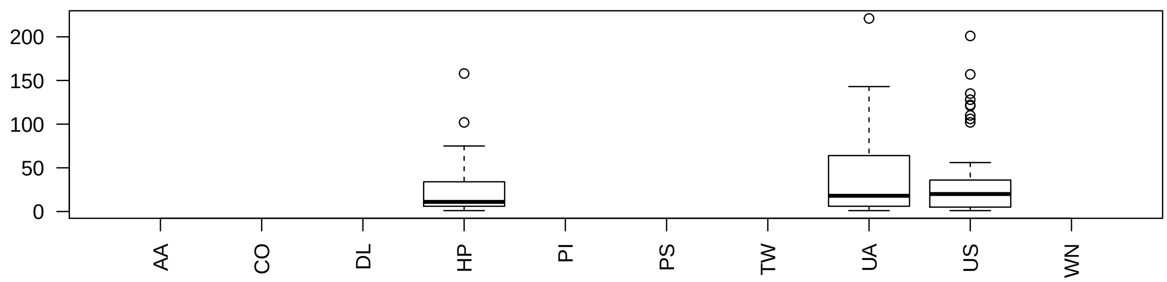


Boxplots

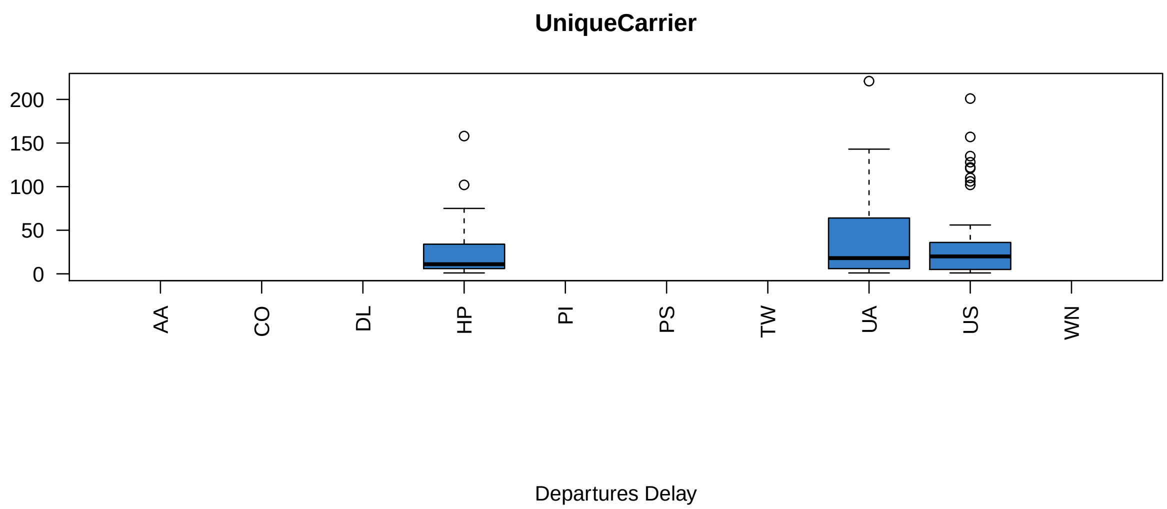
Checks on Departures Delayed from Boston between Year 2000 and 2006

```
# Librairy
library(ggplot2)
library(tidyr)
library(dplyr)
library(shiny)
library(ggvis)
library(corrplot)
library(data.table)
library(lubridate)

require(Matrix)
if (!require('vcd')) install.packages('vcd')
library(Ckmeans.1d.dp)


## Checks on Departures Delayed from Boston between Year 2000 and 2006

data.filter1 <- data %>% filter(IsDepDelayed == 'YES', Year >= 2000 & Year<2007,
                                Origin =='BOS')

par(las=2,                        # use perpendicular axis labels
```

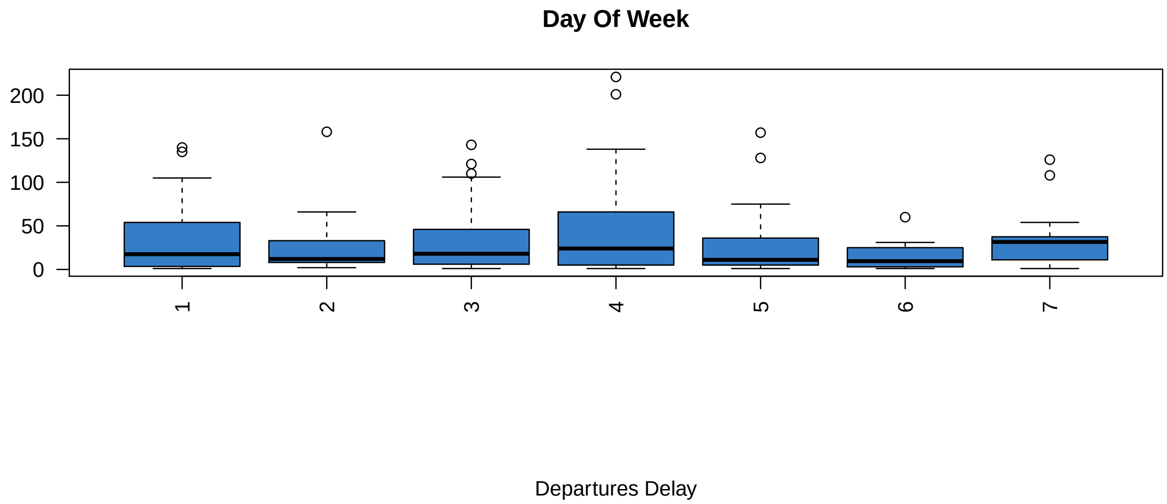
```
    mar=c(10.1,4.1,4.1,2.1),       # create enough space for long x labels
    mgp=c(8,1,0)                   # move x axis legend down to avoid overlap
)

# start with simple boxplot with options
boxplot(DepDelay~UniqueCarrier,data=data.filter1)
```



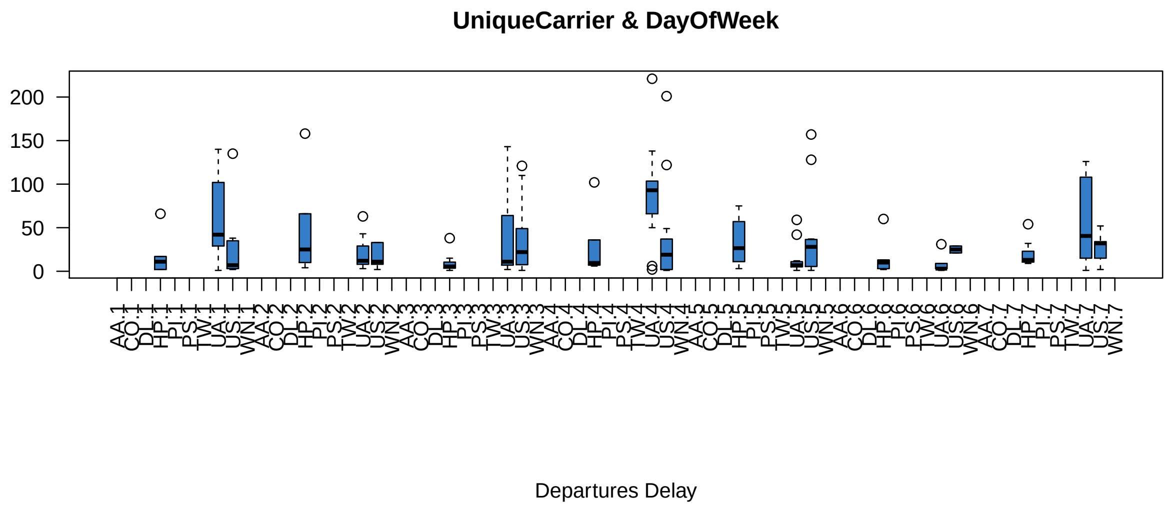

```
#par(mfrow=c(1,3))
boxplot(DepDelay~UniqueCarrier,data=data.filter1, main="UniqueCarrier", xlab="Departures Delay",
        ylab="Lifetime",col="#357EC7")
```




```
boxplot(DepDelay~DayOfWeek,data=data.filter1, main="Day Of Week", xlab="Departures Delay",
        ylab="",col="#357EC7") # Day Of Week = 1 is Monday
```

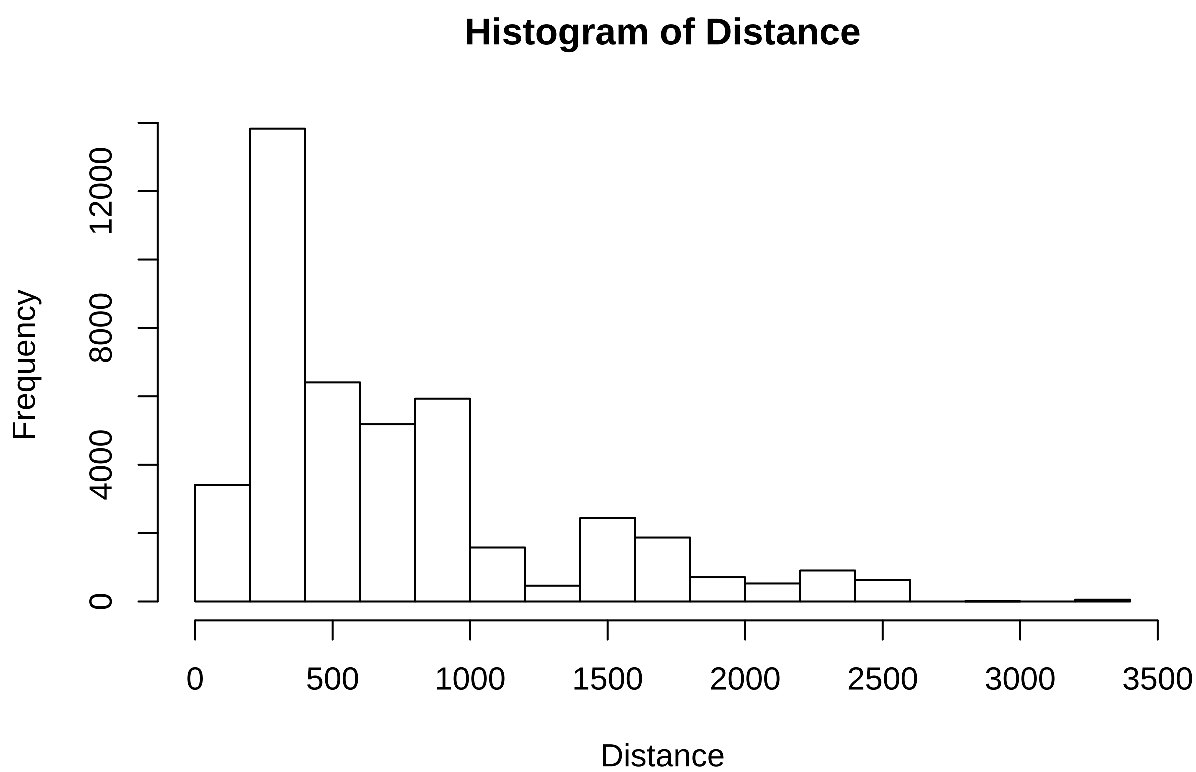



```r
boxplot(DepDelay~UniqueCarrier+DayOfWeek,data=data.filter1,
        main="UniqueCarrier & DayOfWeek", xlab="Departures Delay", ylab="Lifetime",col="#357EC7")
```



Histogram

```r
## Histogram of Distance on full data set
with(data,hist(Distance))
```

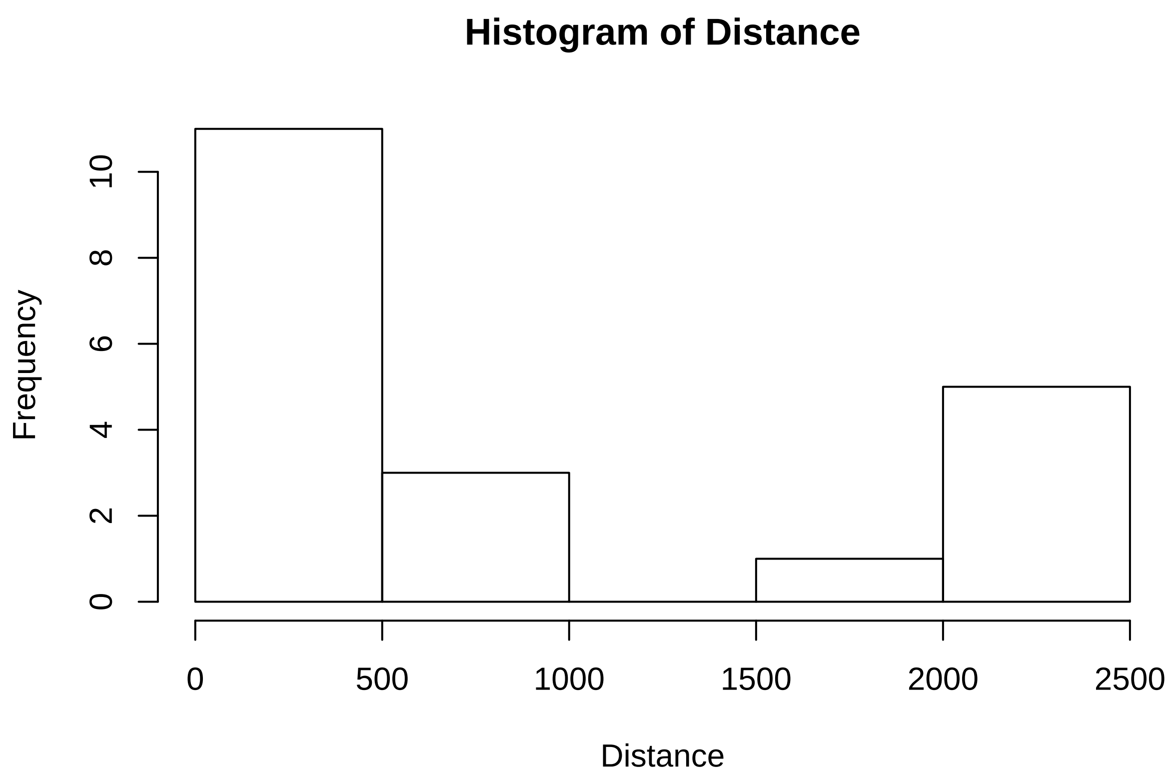



```
## Checks on early arrivals from Boston
data.filter2 <- data %>% filter(IsDepDelayed == 'NO', ArrDelay == 0,Origin=='BOS')
summary(data.filter2)
```

```
##       Year          Month      DayofMonth      DayOfWeek    DepTime
##  Min.   :1991   Min.   :1   Min.   : 4.00   Min.   :1   Min.   : 559.0
##  1st Qu.:1998   1st Qu.:1   1st Qu.: 9.75   1st Qu.:2   1st Qu.: 944.5
##  Median :2002   Median :1   Median :12.00   Median :3   Median :1328.0
##  Mean   :2001   Mean   :1   Mean   :13.90   Mean   :3   Mean   :1294.0
##  3rd Qu.:2004   3rd Qu.:1   3rd Qu.:15.50   3rd Qu.:4   3rd Qu.:1656.5
##  Max.   :2006   Max.   :1   Max.   :31.00   Max.   :6   Max.   :1959.0
##
##    CRSDepTime      ArrTime       CRSArrTime   UniqueCarrier
##  Min.   : 605   Min.   : 750   Min.   : 750   US     :11
##  1st Qu.: 945   1st Qu.:1294   1st Qu.:1294   HP     : 5
##  Median :1338   Median :1549   Median :1549   UA     : 4
##  Mean   :1310   Mean   :1527   Mean   :1527   AA     : 0
##  3rd Qu.:1661   3rd Qu.:1876   3rd Qu.:1876   CO     : 0
##  Max.   :2005   Max.   :2138   Max.   :2138   DL     : 0
##                                               (Other): 0
##    FlightNum         TailNum   ActualElapsedTime CRSElapsedTime
##  Min.   : 101.0   N815äâ : 2   Min.   : 70.0     Min.   : 67.0
##  1st Qu.: 425.5   N300Aä : 1   1st Qu.: 97.5     1st Qu.: 92.0
##  Median : 759.0   N327UA : 1   Median :116.0     Median :115.0
##  Mean   :1002.9   N355US : 1   Mean   :179.1     Mean   :175.0
##  3rd Qu.:1317.8   N371äâ : 1   3rd Qu.:295.2     3rd Qu.:286.2
```

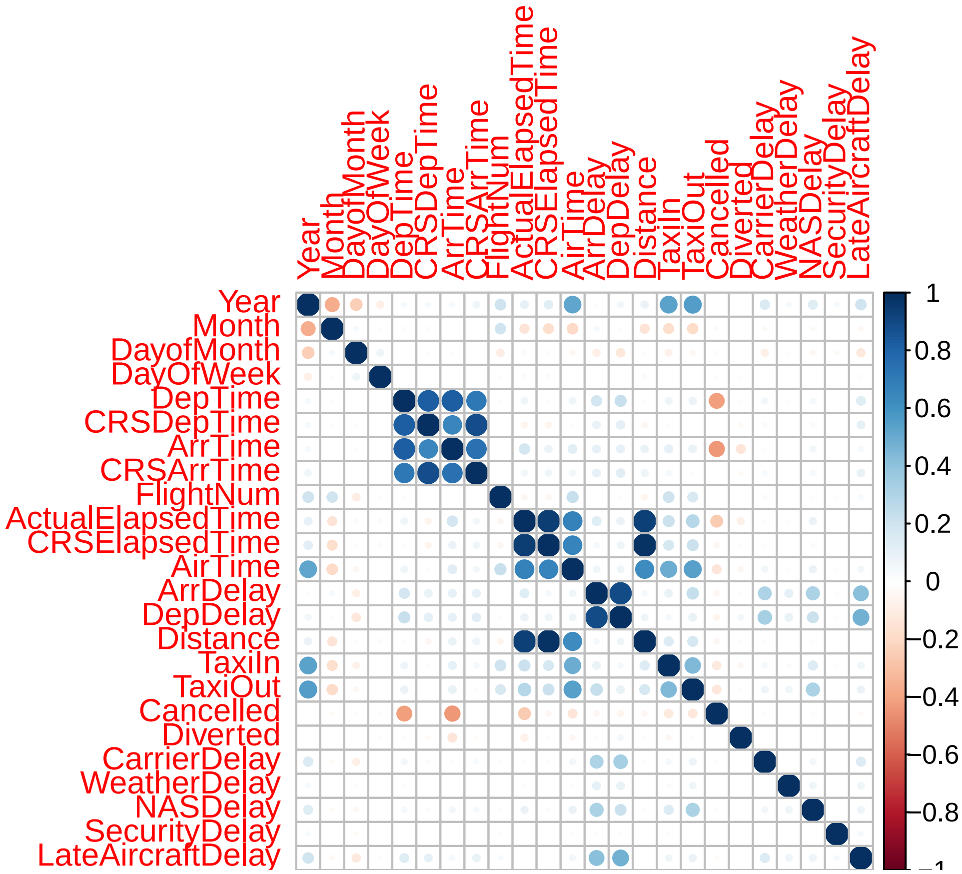
```
##  Max.   :2792.0   (Other):11   Max.   :351.0     Max.   :348.0
##                   NA's   : 3
##     AirTime         ArrDelay    DepDelay          Origin        Dest
##  Min.   : 44.0   Min.   :0   Min.   :-12.00   BOS    :20   PHX    :5
##  1st Qu.: 73.0   1st Qu.:0   1st Qu.: -6.00   ABE    : 0   PIT    :4
##  Median :104.0   Median :0   Median : -4.00   ABQ    : 0   PHL    :3
##  Mean   :165.8   Mean   :0   Mean   : -4.05   ACY    : 0   ORD    :2
##  3rd Qu.:311.0   3rd Qu.:0   3rd Qu.: -1.75   ALB    : 0   ROC    :2
##  Max.   :326.0   Max.   :0   Max.   :  0.00   AMA    : 0   CLE    :1
##  NA's   :3                                    (Other): 0   (Other):3
##     Distance        TaxiIn          TaxiOut        Cancelled
##  Min.   : 185   Min.   : 3.000   Min.   : 5.00   Min.   :0
##  1st Qu.: 343   1st Qu.: 4.000   1st Qu.:16.00   1st Qu.:0
##  Median : 496   Median : 7.000   Median :16.00   Median :0
##  Mean   : 983   Mean   : 7.941   Mean   :18.35   Mean   :0
##  3rd Qu.:1890   3rd Qu.:11.000   3rd Qu.:21.00   3rd Qu.:0
##  Max.   :2300   Max.   :21.000   Max.   :40.00   Max.   :0
##                 NA's   :3        NA's   :3
##  CancellationCode    Diverted  CarrierDelay  WeatherDelay    NASDelay
##      : 5          Min.   :0   Min.   :0     Min.   :0     Min.   :0
##  A   : 0          1st Qu.:0   1st Qu.:0     1st Qu.:0     1st Qu.:0
##  B   : 0          Median :0   Median :0     Median :0     Median :0
##  C   : 0          Mean   :0   Mean   :0     Mean   :0     Mean   :0
##  NA's:15          3rd Qu.:0   3rd Qu.:0     3rd Qu.:0     3rd Qu.:0
##                   Max.   :0   Max.   :0     Max.   :0     Max.   :0
##                               NA's   :15    NA's   :15    NA's   :15
##  SecurityDelay LateAircraftDelay IsArrDelayed IsDepDelayed  groupDepDelay
##  Min.   :0     Min.   :0         NO :20       NO :20       (-16,-2]:15
##  1st Qu.:0     1st Qu.:0         YES: 0       YES: 0       (-2,1]  : 5
##  Median :0     Median :0                                   (1,10]  : 0
##  Mean   :0     Mean   :0                                   (10,473]: 0
##  3rd Qu.:0     3rd Qu.:0
##  Max.   :0     Max.   :0
##  NA's   :15    NA's   :15
```

```r
## Histogram of Distance on filter dataset
with(data.filter2,hist(Distance))
```

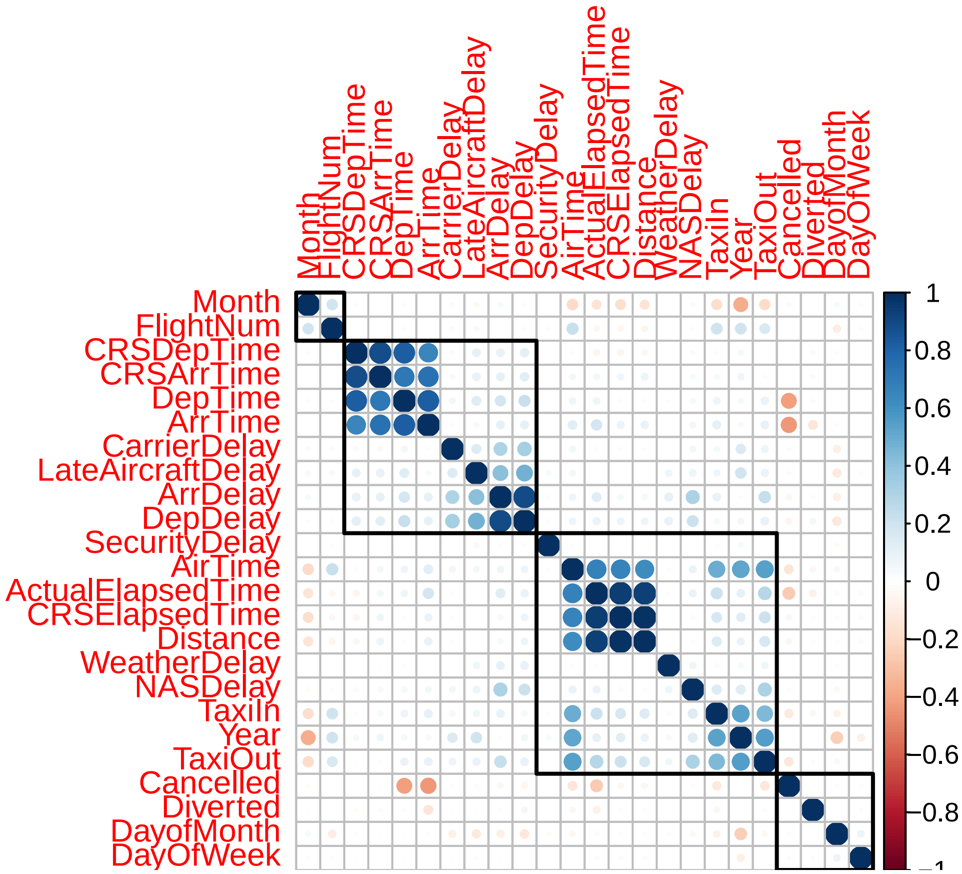


Visualization of a correlation matrix

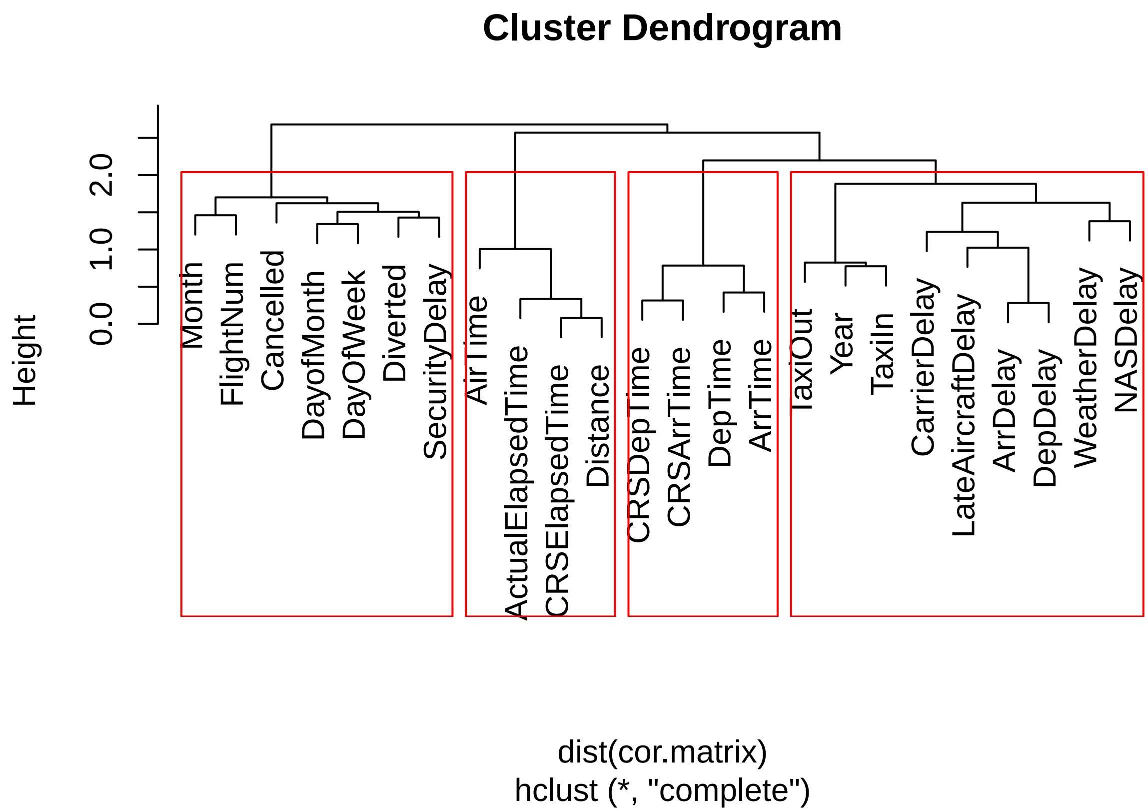
```
## Filter and List only numeric columns from a data frame
nums <- sapply(data, is.numeric)
data.filter3 <- data[,nums]

# Replace NAs in data with 0
data.filter3[is.na(data.filter3)] <- 0
cor.matrix <-cor(data.filter3,use = "na.or.complete")
corrplot(cor.matrix)
```




```r
# ?corrplot # for more details

corrplot(cor.matrix, order="hclust", hclust.method="complete", addrect = 4)
```




```
# names(cor.matrix) <- colnames(data.filter3)
cor.clust <- hclust(dist(cor.matrix), method = "complete")
plot(cor.clust)
rect.hclust(cor.clust, k = 4)
```

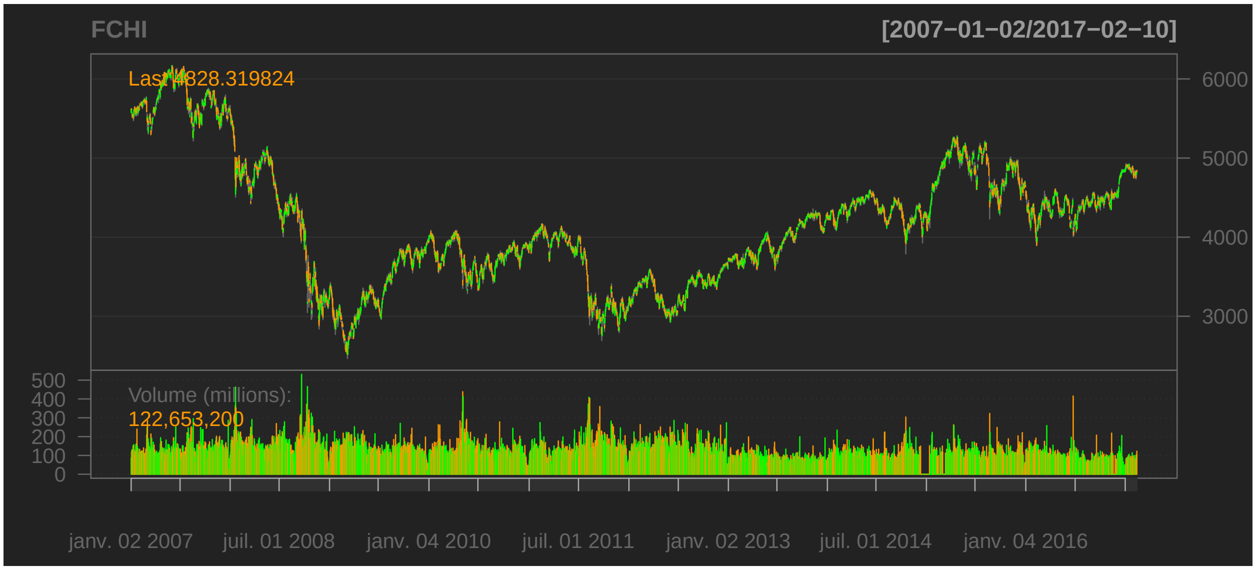


Time Series Analysis

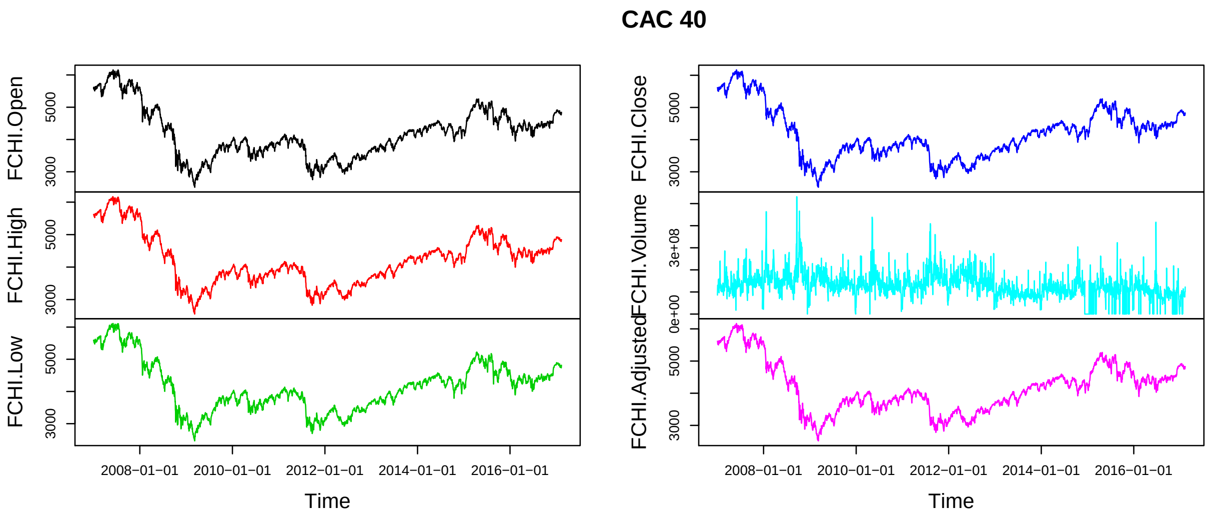
```
library(quantmod)
library(fBasics)
library(tseries)
library(Hmisc)

getSymbols("^FCHI",src="yahoo")
```

```
## [1] "FCHI"
```

```
chartSeries(FCHI) # CAC 40
```




```r
write.zoo(FCHI,"FCHI.csv",index.name="Date",sep=",") # save an xts object to csv

FCHI.import<- read.csv('FCHI.csv')
FCHI <- as.timeSeries(FCHI.import)
plot(FCHI, main="CAC 40")
```



Checking Basic Statistics

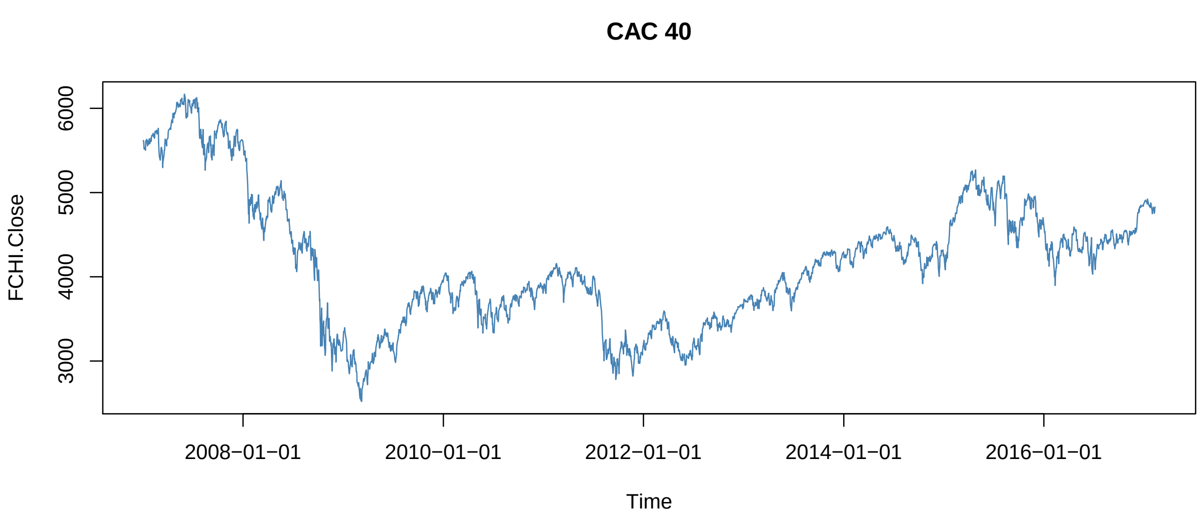
```r
basicStats(FCHI) # Basic Statistics
```

```
##                  FCHI.Open     FCHI.High      FCHI.Low    FCHI.Close
## nobs          2.591000e+03  2.591000e+03  2.591000e+03  2.591000e+03
## NAs           0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
## Minimum       2.519430e+03  2.552990e+03  2.465460e+03  2.519290e+03
## Maximum       6.157330e+03  6.168150e+03  6.112290e+03  6.168150e+03
## 1. Quartile   3.632895e+03  3.660780e+03  3.591075e+03  3.625420e+03
```

```
## 3. Quartile  4.572955e+03  4.599495e+03  4.533880e+03  4.565015e+03
## Mean         4.171737e+03  4.202615e+03  4.135949e+03  4.170186e+03
## Median       4.110850e+03  4.136610e+03  4.080150e+03  4.108270e+03
## Sum          1.080897e+07  1.088897e+07  1.071624e+07  1.080495e+07
## SE Mean      1.512094e+01  1.510276e+01  1.514308e+01  1.512350e+01
## LCL Mean     4.142087e+03  4.173000e+03  4.106255e+03  4.140531e+03
## UCL Mean     4.201387e+03  4.232229e+03  4.165643e+03  4.199842e+03
## Variance     5.924133e+05  5.909896e+05  5.941499e+05  5.926142e+05
## Stdev        7.696839e+02  7.687585e+02  7.708112e+02  7.698144e+02
## Skewness     4.860190e-01  4.953230e-01  4.721540e-01  4.859360e-01
## Kurtosis    -2.516240e-01 -2.599530e-01 -2.459210e-01 -2.510850e-01
##              FCHI.Volume FCHI.Adjusted
## nobs        2.591000e+03  2.591000e+03
## NAs         0.000000e+00  0.000000e+00
## Minimum     0.000000e+00  2.519290e+03
## Maximum     5.312476e+08  6.168150e+03
## 1. Quartile 1.011688e+08  3.625420e+03
## 3. Quartile 1.571855e+08  4.565015e+03
## Mean        1.321279e+08  4.170186e+03
## Median      1.276144e+08  4.108270e+03
## Sum         3.423435e+11  1.080495e+07
## SE Mean     1.092376e+06  1.512350e+01
## LCL Mean    1.299859e+08  4.140531e+03
## UCL Mean    1.342699e+08  4.199842e+03
## Variance    3.091802e+15  5.926142e+05
## Stdev       5.560398e+07  7.698144e+02
## Skewness    9.072960e-01  4.859360e-01
## Kurtosis    4.371497e+00 -2.510850e-01
```

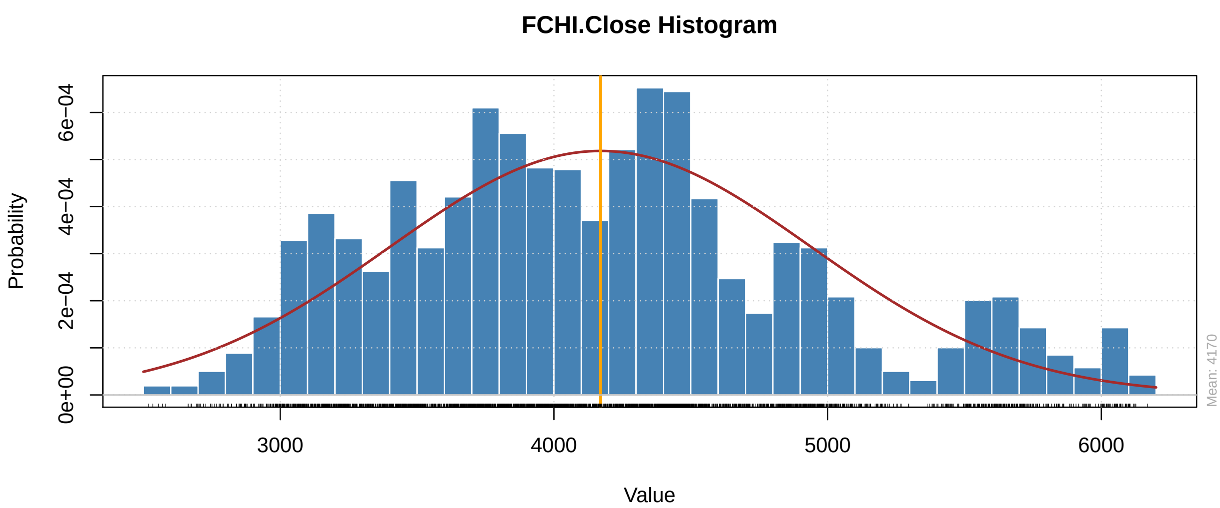
Visualise the Closing Price of CAC 40

```r
cac40.close <- as.timeSeries(FCHI[,4]) #cac40.close <- as.ts(FCHI[,4],start=c(2007,01,02))
plot(cac40.close, type = "l", col = "steelblue", main = "CAC 40")
abline(h = 0, col = "grey")
```

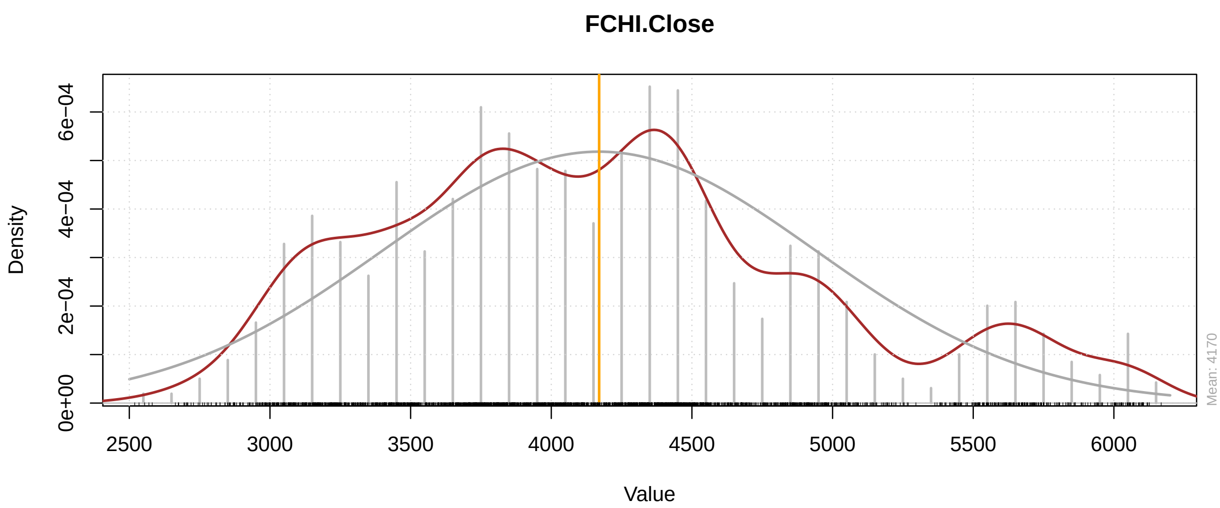


Histogram & Density Plots:

```r
histPlot(cac40.close)
```




```r
densityPlot(cac40.close)
```



Descriptive Statistics on the Closing Price of CAC 40

```r
summary(cac40.close)
```

```
##    FCHI.Close
##  Min.   :2519
##  1st Qu.:3625
##  Median :4108
##  Mean   :4170
##  3rd Qu.:4565
```

```
## Max.   :6168
```

```r
describe(cac40.close)
```

```
## cac40.close [FCHI.Close]  Format:%Y-%m-%d
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     2591        0     2577        1     4170    867.2     3052     3186
##      .25      .50      .75      .90      .95
##     3625     4108     4565     5381     5682
##
## lowest : 2519.29 2534.45 2554.55 2569.63 2581.46
## highest: 6117.96 6120.20 6125.60 6125.81 6168.15
```

Hypothesis Testing

Tests the null of normality:

```r
jarque.bera.test(cac40.close) # Tests the null of normality
```

```
##
##  Jarque Bera Test
##
## data:  cac40.close
## X-squared = 108.78, df = 2, p-value < 2.2e-16
```

Test for Trend Stationarity

```r
kpss.test(cac40.close, null = "Trend") # KPSS Test for Trend Stationarity
```

```
##
##  KPSS Test for Trend Stationarity
##
## data:  cac40.close
## KPSS Trend = 3.433, Truncation lag parameter = 11, p-value = 0.01
```

Augmented Dickey-Fuller Test for Stationarity:

```r
tseries::adf.test(cac40.close, k = 10)  # Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  cac40.close
## Dickey-Fuller = -1.9931, Lag order = 10, p-value = 0.5813
## alternative hypothesis: stationary
```

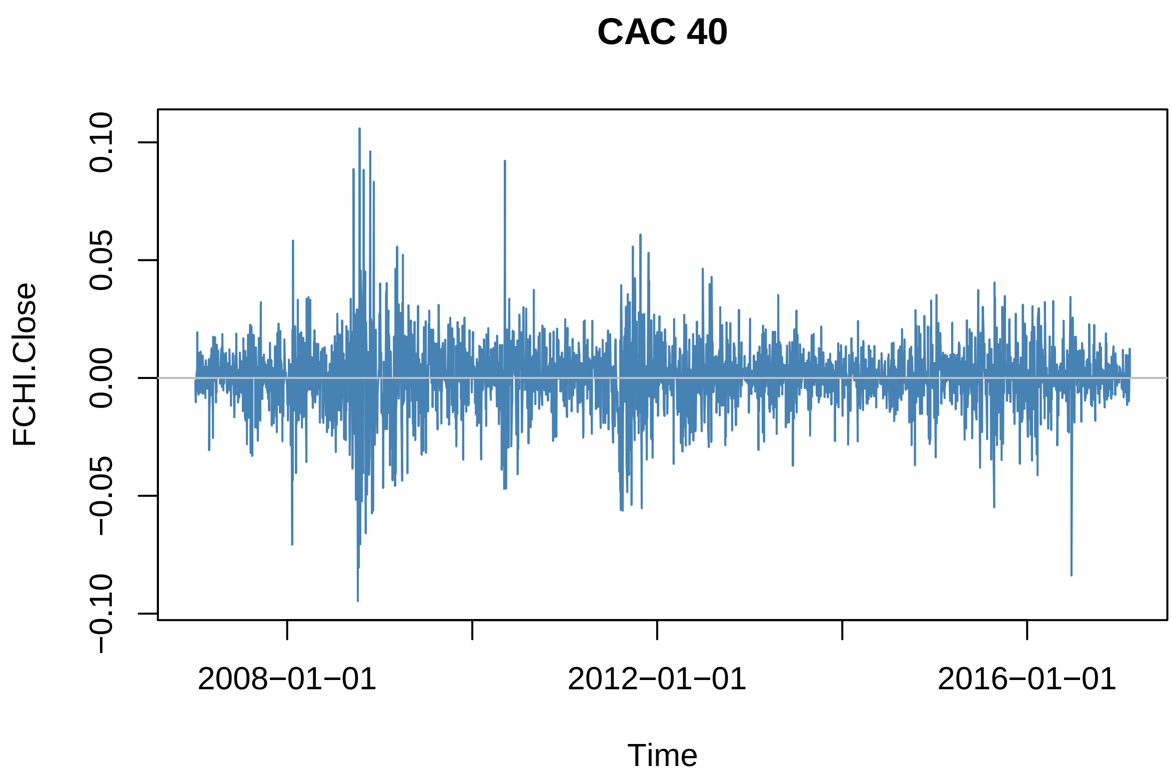
```r
adf.test(diff(log(cac40.close))[-1,], alternative="stationary", k=0)  # Augmented Dickey-Fuller Test
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(log(cac40.close))[-1, ]
```

```
## Dickey-Fuller = -52.803, Lag order = 0, p-value = 0.01
## alternative hypothesis: stationary
```

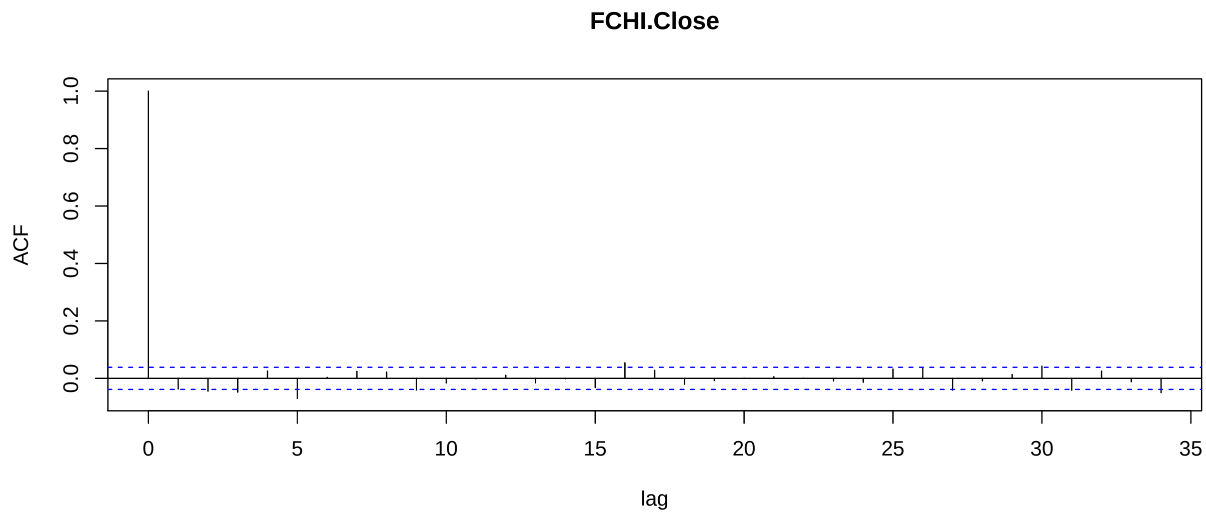
Time Series of difference of log CAC 40

```r
cac40.close.df <- diff(log(cac40.close))[-1,]
plot(cac40.close.df, type = "l", col = "steelblue", main = "CAC 40")
abline(h = 0, col = "grey")
```

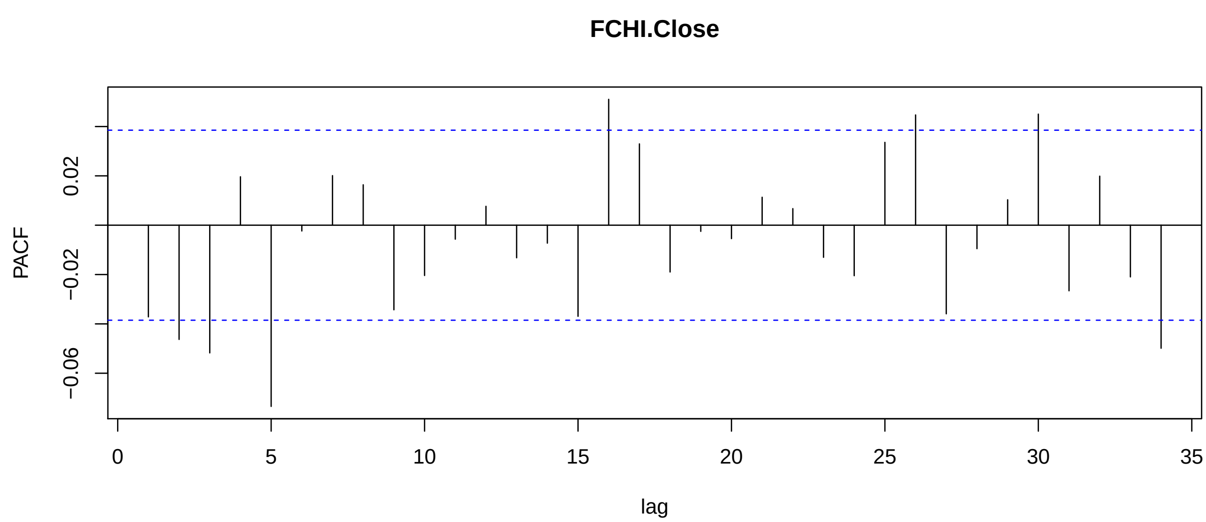


Autocorrelation Function Plots

```r
acfPlot(cac40.close.df)
```




```r
pacfPlot(cac40.close.df)
```



Modeling

```r
fit <- arima(cac40.close.df, order = c(5, 0, 0))
fit
```

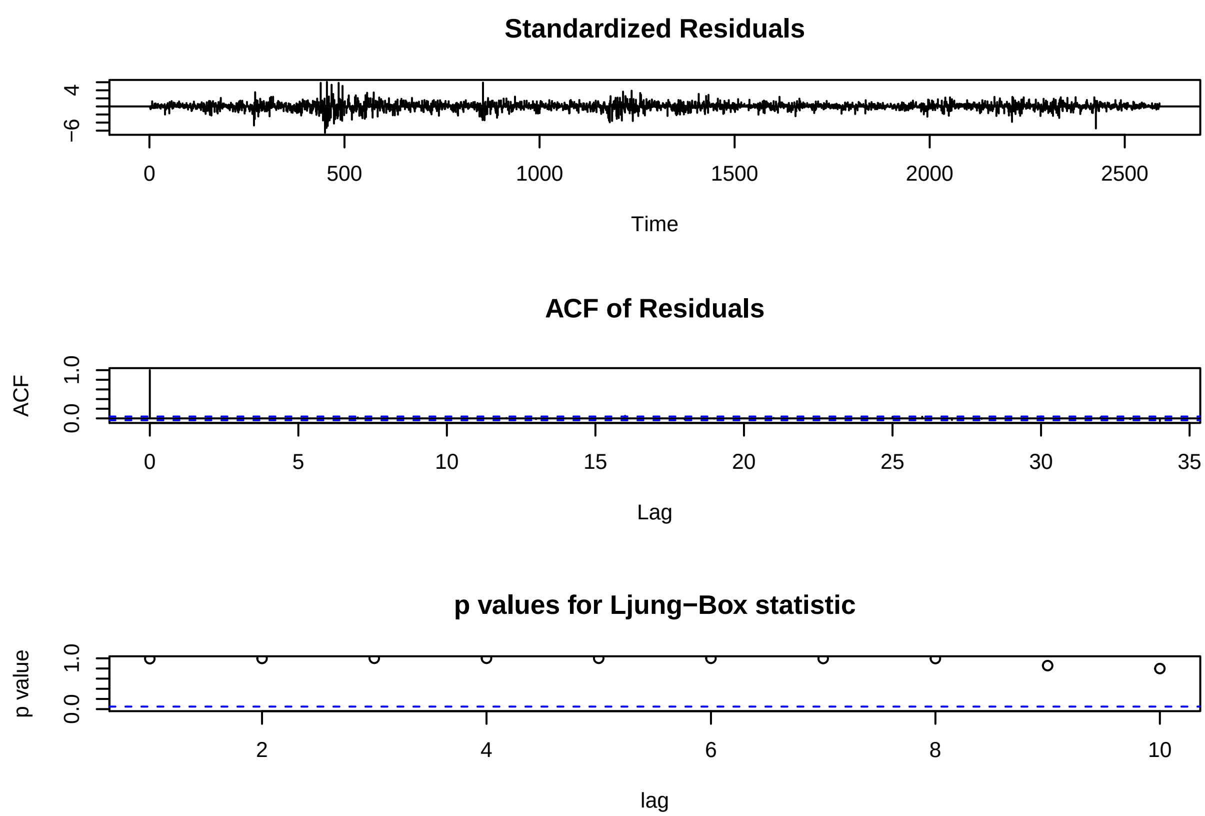
```
##
## Call:
## arima(x = cac40.close.df, order = c(5, 0, 0))
##
## Coefficients:
##           ar1      ar2      ar3     ar4      ar5  intercept
##       -0.0388  -0.0511  -0.0544  0.0167  -0.0733     -1e-04
## s.e.   0.0196   0.0196   0.0196  0.0196   0.0196      2e-04
##
```

```
## sigma^2 estimated as 0.0002277:  log likelihood = 7186.52,  aic = -14359.04
```

```r
summary(fit)
```

```
##           Length Class  Mode
## coef         6   -none- numeric
## sigma2       1   -none- numeric
## var.coef    36   -none- numeric
## mask         6   -none- logical
## loglik       1   -none- numeric
## aic          1   -none- numeric
## arma         7   -none- numeric
## residuals 2590   ts     numeric
## call         3   -none- call
## series       1   -none- character
## code         1   -none- numeric
## n.cond       1   -none- numeric
## nobs         1   -none- numeric
## model       10   -none- list
```

```r
tsdiag(fit)
```



Introduction to Data Science Algorithms in R

Ensemble Feature Selection on Titanic Dataset

```r
# install.packages("fscaret", dependencies = c("Depends", "Suggests"))

library(fscaret)
library(caret)
```

```
# list of models on fscaret packages:
data(funcRegPred)
model.list <-data.frame(models=funcRegPred)

# list of models on caret packages:
names(getModelInfo())
```

```r
# Import dataset from the UCI Machine Learning Repository
# (http://archive.ics.uci.edu/ml/datasets.html)
# titanic.data <- read.csv('http://math.ucdenver.edu/RTutorial/titanic.txt',sep='\t')
titanic.data <- read.csv('titanic.txt',sep='\t') # import data from working directory
```

```r
# creating new title feature
titanic.data$Title <- ifelse(grepl('Mr ',titanic.data$Name),'Mr',
                             ifelse(grepl('Mrs ',titanic.data$Name),'Mrs',
                                    ifelse(grepl('Miss',titanic.data$Name),'Miss','Nothing')))
titanic.data$Title <- as.factor(titanic.data$Title)

# Replace NAs in Age with the median age
titanic.data$Age[is.na(titanic.data$Age)] <- median(titanic.data$Age, na.rm=T)
```

```r
# reorder data set so response variable is last column
titanic.data <- titanic.data[c('PClass', 'Age',    'Sex',   'Title', 'Survived')]

# binarize all factors
titanicDummy <- dummyVars("~.",data=titanic.data, fullRank=F)
titanic.data <- as.data.frame(predict(titanicDummy,titanic.data))
```

We have a look at the structure of data

```r
str(titanic.data)
```

```
## 'data.frame':    1313 obs. of  11 variables:
##  $ PClass.1st   : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PClass.2nd   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ PClass.3rd   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Age          : num  29 2 30 25 0.92 47 63 39 58 71 ...
##  $ Sex.female   : num  1 1 0 1 0 0 1 0 1 0 ...
##  $ Sex.male     : num  0 0 1 0 1 1 0 1 0 1 ...
##  $ Title.Miss   : num  1 1 0 0 0 0 1 0 0 0 ...
##  $ Title.Mr     : num  0 0 1 0 0 1 0 1 0 1 ...
##  $ Title.Mrs    : num  0 0 0 1 0 0 0 0 1 0 ...
##  $ Title.Nothing: num  0 0 0 0 1 0 0 0 0 0 ...
##  $ Survived     : num  1 0 0 0 1 1 1 0 1 0 ...
```

```r
# split data set into train and test portion
set.seed(1234)
```

```r
splitIndex <- createDataPartition(titanic.data$Survived, p = .75, list = FALSE, times = 1)
# splitIndex <- sample(nrow(titanic.data), floor(0.75*nrow(titanic.data)))
trainDF <- titanic.data[ splitIndex,]
testDF  <- titanic.data[-splitIndex,]
```

We now run an ensemble feature selection specifying list of models available on 'fscaret' package:

```r
# limit models to use in ensemble and run fscaret
featureSelection.models <- c("glm", "gbm", "treebag", "ridge", "lasso")
featureSelection<-fscaret(trainDF, testDF, myTimeLimit = 40, preprocessData=TRUE,
              Used.funcRegPred = featureSelection.models, with.labels=TRUE,
              supress.output=FALSE, no.cores=2)
```

```r
# analyze results
print(featureSelection$VarImp)
```

```
## $rawMSE
##         gbm       glm     lasso     ridge   treebag
## 1 0.1286814 0.1386274 0.1386274 0.1386278 0.1298043
##
## $rawRMSE
##         gbm      glm    lasso     ridge   treebag
## 1 0.3587219 0.372327 0.372327 0.3723276 0.3602836
##
## $matrixVarImp.RMSE
##          gbm glm      lasso      ridge   treebag       SUM       SUM%
## 5 28.1448932   0 25.0403181 25.0400842 13.307300 91.532596 100.000000
## 7 27.2664317   0 24.2347615 24.2353151 13.670037 89.406545  97.677275
## 3 26.0342484   0 15.2132114 15.2127921 24.635434 81.095686  88.597604
## 1  5.4238748   0 11.2699282 11.2694387 13.467735 41.430977  45.263631
## 8  0.4340408   0 12.1964247 12.1966233  6.545685 31.372774  34.274974
## 4 11.2236464   0  0.1074810  0.1075533 18.074952 29.513633  32.243850
## 6  0.8201142   0  6.3783194  6.3784115  2.427869 16.004714  17.485262
## 2  0.0000000   0  1.4203760  1.4204060  4.552760  7.393542   8.077497
## 9  0.6527505   0  0.4850858  0.4851359  2.884740  4.507713   4.924708
##     ImpGrad Input_no
## 5  0.000000        5
## 7  2.322725        7
## 3  9.295582        3
## 1 48.910998        1
## 8 24.277012        8
## 4  5.925968        4
## 6 45.771792        6
## 2 53.803973        2
## 9 39.031759        9
##
## $matrixVarImp.MSE
##          gbm glm      lasso      ridge   treebag       SUM       SUM%
## 5 28.1448932   0 24.1253213 24.1250595 13.249615 89.644889 100.000000
## 7 27.2664317   0 23.3492005 23.3496986 13.610779 87.576110  97.692251
## 3 26.0342484   0 14.6573064 14.6568801 24.528643 79.877078  89.103884
## 1  5.4238748   0 10.8581144 10.8576264 13.409354 40.548970  45.232885
## 8  0.4340408   0 11.7507559 11.7509294  6.517310 30.453036  33.970745
## 4 11.2236464   0  0.1035535  0.1036230 17.996600 29.427423  32.826660
## 6  0.8201142   0  6.1452496  6.1453290  2.417345 15.528038  17.321721
```

```
## 2  0.0000000   0  1.3684741  1.3685009  4.533025  7.270000   8.109776
## 9  0.6527505   0  0.4673603  0.4674078  2.872236  4.459754   4.974912
##     ImpGrad Input_no
## 5  0.000000        5
## 7  2.307749        7
## 3  8.791247        3
## 1 49.235787        1
## 8 24.898126        8
## 4  3.367852        4
## 6 47.232764        6
## 2 53.181465        2
## 9 38.655374        9
##
## $model
## list()
```

```r
print(featureSelection$PPlabels)
```

```
##   Orig Input No        Labels
## 1             1    PClass.1st
## 2             2    PClass.2nd
## 3             3    PClass.3rd
## 4             4           Age
## 5             6      Sex.male
## 6             7    Title.Miss
## 7             8      Title.Mr
## 8             9     Title.Mrs
## 9            10 Title.Nothing
```

References (Some useful R links)

I do recommend that you simply use seach engines to find out the links to resources that suits you. Here are a few of such links:

Introduction:

- Introduction to R https://cran.r-project.org/doc/manuals/R-intro.html
- Interactive intro to R programming language https://www.datacamp.com/courses/introduction-to-r
- for data manipulation https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

Graphics:

- Tutorial on plots http://cran.r-project.org/doc/contrib/Rossiter-RIntro-ITC.pdf

Statistics:

- Quick-R http://www.statmethods.net/
- Beginner's tutorial for Time Series http://www.stat.pitt.edu/stoffer/tsa2/index.html

Last updated on February 2017, SNCF Division MMI