



Data Pre-Processing: Loading Data and Basic Visualisations

Learning Objectives:

At the end of the class participants should be able to understand different ways of loading data from different sources such as database and other file systems into Rstudio environments for preprocessing and visualisation.

Packages to install

We will be working with following Packages

```
library(rmarkdown)
```

```
library(tidyverse)#(xml2)
```

```
library(readxl)
```

```
library(DBI)
```

```
library(RMySQL) #linux(sudo apt-get install libmysqlclient-dev)
```

```
library(haven)
```




Data

We have a series of data files that we will be working with some are inbuilt while others can be read directly into Rstudio.

For the dataset we shall be looking at

1. Fertility
2. `who_suicide_statistics`

Reading Comma delimited files (csv) using read.table()



```
who_suicide <- read.table(file.choose(),sep = “ , ”, header=TRUE)  
head(who_suicide)
```

Generally used to load data in different format into R

- file: Path to the file containing the data to be imported into R.
- sep: field separator character. \t is used for tab-delimited file.
- header: logical value. If TRUE, read.table() assumes that

your file has a header row, so row 1 is the name of each column. If that's not the case, you can add the argument header = FALSE.

- dec: the character used in the file for decimal points.

Reading Comma Separated File (txt) using read.table()



```
auto <- read.table("https://s3.amazonaws.com/assets.datacamp.com/blog_assets/test.txt",  
                  header = FALSE)  
  
head(auto[,2:3])
```

`read.table()` can be used to read .txt or a tab-delimited text file.

The `header` argument is always set at `TRUE`, which indicates that the first line of the file being read contains the header with the variable names;

The `fill` argument is also set as `TRUE`, which means that if rows have unequal length, blank fields will be added implicitly.

Dealing with Missing Values and outliers

To check for missing values in your dataset you use the command

- `sum(is.na(data))` or
- `mean(is.na(data))`

For the sum it tells you the total numbers of missing value in your data set, while the mean tells you the average of missing values in your dataset which will help you to determine how to deal with the missing dataset of whether to omit it or use the weighted mean or median.

Once you detect a missing value(NA, NaN) in your dataset you can use the command:

`na.omit(auto).`

You can only omit the missing value when you are sure that the missing value will not influence your result at the end of the day.



Dealing with Missing Values and outliers (Cont'n)

But for your model to work properly, there is always the need to fill the missing value with either of the following metrics

- mean
- median
- weighted mean

And is more advisable to fill your missing value with either the weighted mean or media because the median and weighted mean is less affected by outliers compare to mean.

```
auto$speed[is.na(auto$speed)]<-median(auto$speed,na.rm=TRUE)
```

```
auto
```

Reading comma seperated files using read.csv



```
who_suicide<-read.csv(file.choose(), header=TRUE, sep = ",", strip_white= TRUE)
```

```
head(who_suicide)
```

You can use the `read.csv()` or `read.csv2()` functions. The former function is used if the separator is a `,` the latter if `;` is used to separate the values in your data file.

The `strip.white` argument allows you to indicate whether you want the white spaces from **unquoted** character fields stripped. It is only used when `sep` has been specified and only takes on a logical value. The `na.strings` indicates which strings should be interpreted as NA values. In this case, the string `"EMPTY"` is to be interpreted as an NA value.

Reading semicolon separated files using read.csv2

```
who_suicide_1<-read.csv2(file.choose(), header=TRUE)
```

```
head((who_suicide_1)
```

The read.csv2 is often used to read semicolon separated files into R.

Setting sep argument is optional when using read.csv2 as seen in the case of read.csv

Reading “Tab-separated value” files (“.txt”) using read.delim()



```
potato <- read.delim(file.choose(),sep = "\t", header=F,
```

```
    row.names = c("M", "N", "O"),
```

```
    col.names= c("X", "Y", "Z", "A","B"))
```

```
head(potato[,4:9])
```

read.delim() is used for reading TAB delimited files (txt) by

specifying the argument sep = " ,"

The skip argument specifies that the first two rows of the dataset are not read into R.



Using readr Package

- a. In terms of speed, readr is ~10x faster than base read.table() functions (read.csv, read.csv2).
- b. By default, strings are untouched and common date/time formats are automatically passed.



readr

- `read_csv()`: comma delimited files
- `read_csv2()`: semicolon separated files
- `read_tsv()`: tab delimited files
- `read_delim()`: files with any delimiter



Import Excel files

We can always use the readxl package to get data out of Excel and into R. The readxl package supports both .xls format and the modern xml-based .xlsx format.

To import excel sheet into R, we use the function read_excel() and specify the sheet number in the arguments.

Import Excel files using read_excel() in readxl Package

```
sht1 <- read_excel(file.choose(), sheet = 1)
```

```
sht2 <- read_excel(file.choose(), sheet = 2)
```

```
str(sht1)
```



Importing data from databases

To import data from a database you first have to create a connection to it. The package DBI is used to connect to SQL server through R and RMySQL to perform SQL queries within R. The Function `dbConnect()` creates a connection between your R session and a SQL database. The first argument is mapping the data between R and the database. For hosted SQL DB, we need to specify the following arguments in `dbConnect()`: `dbname`, `host`, `port`, `user` and `password`.



Establish a connection

To extract data from database in a remote server, we need to first establish the connection to the server in R.

```
host <- "courses.csrrinzqubik.us-east-1.rds.amazonaws.com"
```

```
connect <- dbConnect(RMySQL::MySQL(), dbname = "tweater",
```

```
host = host, port = 3306, user = "student", ' password = "datacamp")“
```


List the database tables




Once we are connected to the database. We can use `dbListTables()` to see what tables the database contains:

```
tables <- dbListTables(connect)
```

```
## [1] "comments" "tweets" "users"
```

Import data from tables



We can use the `dbReadTable()` function to import data from the database tables.

```
users <- dbReadTable(connect, "users")
```

```
users
```

Task 1

Connect the database with the

- `host="car-simulation-station.c9az8e0qjbgo.us-east-1.rds.amazonaws.com"`
- Name of database: `"car_simulation_station"`
- `user="datacamp_user"`
- `password="learn tabular data for fun and profit"`
- read the tables in the datables
- Print at least two tables and save as csv

Importing data from the web

We can also import csv file hosted on remote server directly into R.


You can use `read_xlsx` to directly import excel files from the web.

```
bcancer <-
```

```
read.csv("https://raw.githubusercontent.com/mGalarnyk/Python_Tutorials/master/Kaggle/BreastCancerWisconsin/data/data.csv", header = T)
```

```
str(house)
```

Importing data from other statistical software



To import data from other statistical software such as Stata, SPSS, Sas. We use the package called haven.

- SAS: `read_sas()`
- STATA: `read_dta()`
- SPSS: `read_sav()`

read_sas() to read SAS data file



we use the `read_sas()` function in `haven` package to read sas files into R

```
birth <- read_sas(file.choose())
```

```
head(birth)
```

read_dta() to read Stata data file



we use the read_dta() function in haven package to Stata files into R

```
alcohol <- read_dta(file.choose())
```

```
head(alcohol)
```

read_sav() to read SPSS data file into R \small



we use the read_sav() function in haven package to read SPSS, Stata files into R

```
pers <- read_sav("data/personality.sav")
```

```
head(pers)
```

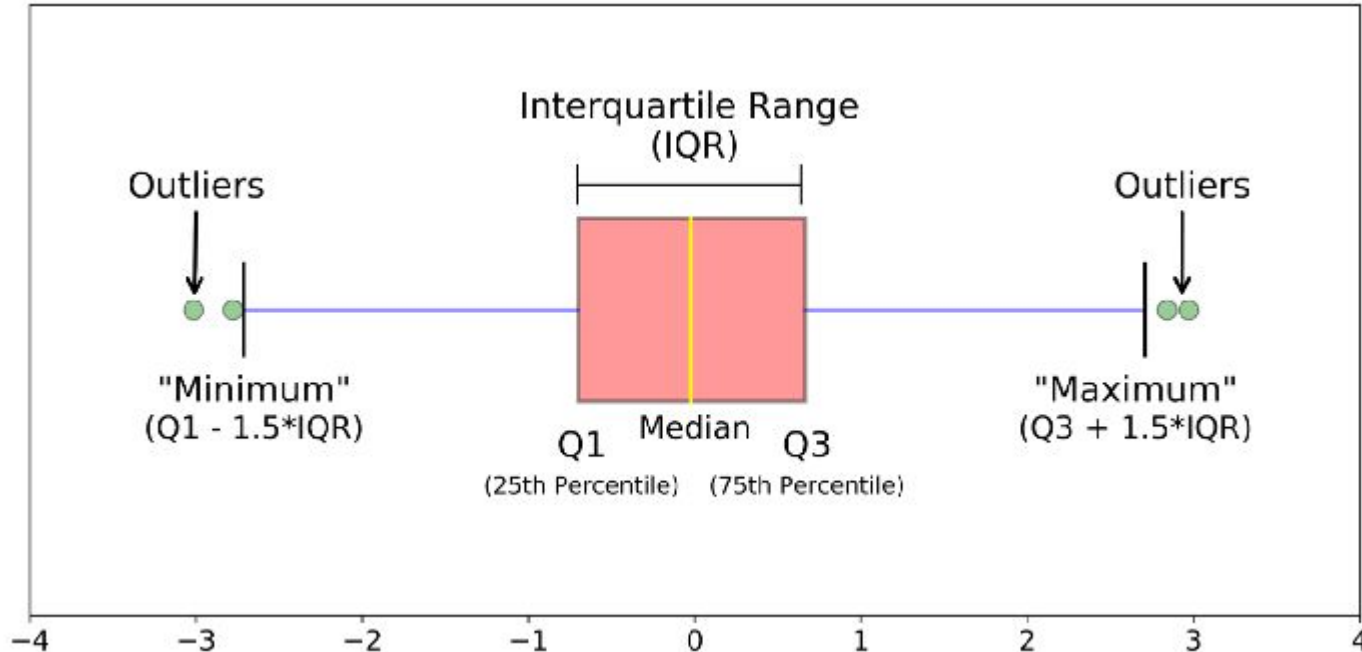
Saving Files as csv in R

```
write.csv(pers, file = "spss.csv")
```

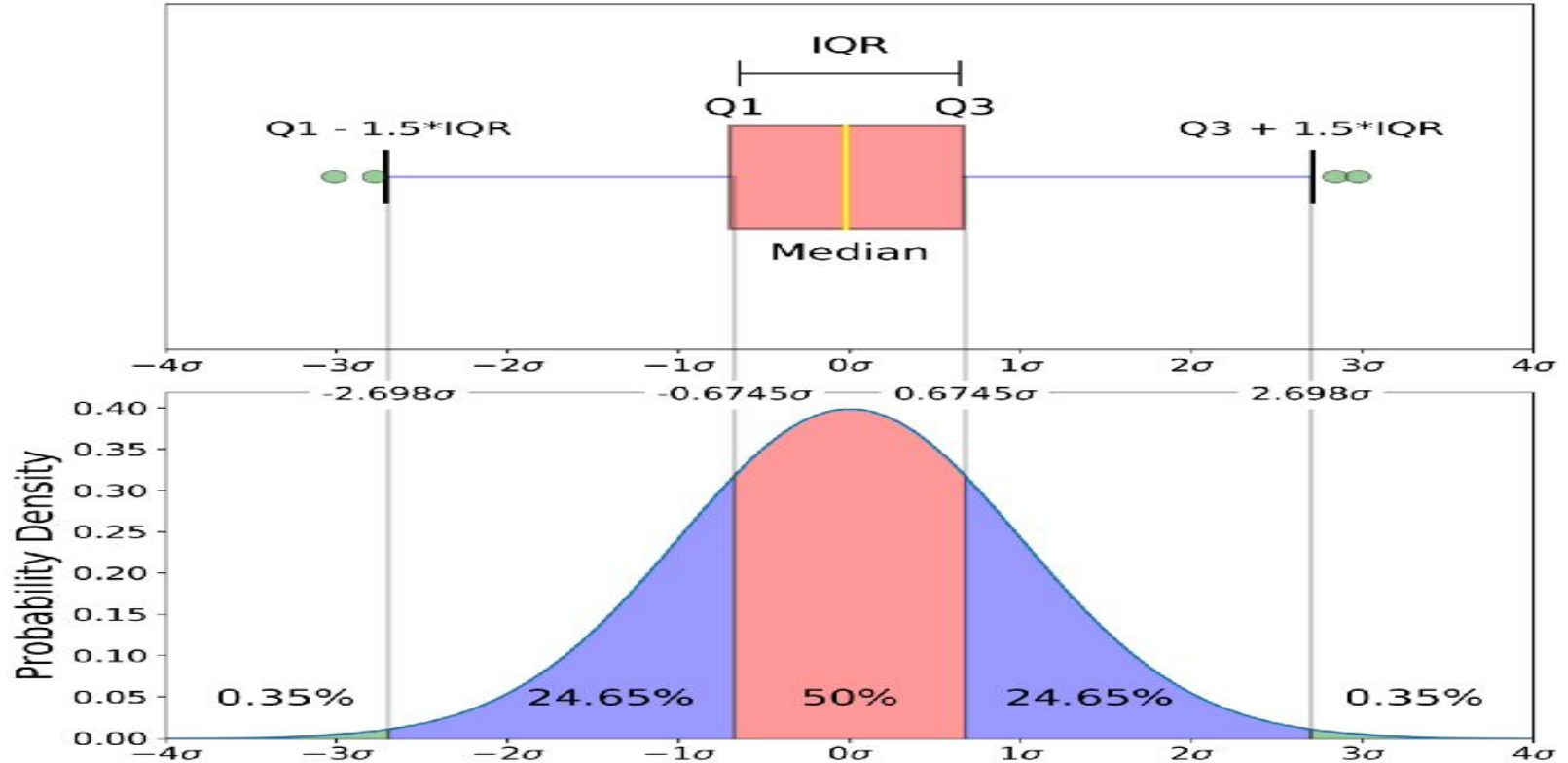

Basic Visualisations

Box Plot in R

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). It can tell you about your outliers and what their values are. It can also tell you if your data is symmetrical, how tightly your data is grouped, and if and how your data is skewed.



Distribution of BoxPlot



BoxPlot

median (Q2/50th Percentile): the middle value of the dataset.

first quartile (Q1/25th Percentile): the middle number between the smallest number (not the “minimum”) and the median of the dataset.

third quartile (Q3/75th Percentile): the middle value between the median and the highest value (not the “maximum”) of the dataset.

interquartile range (IQR): 25th to the 75th percentile.

whiskers (shown in blue)

outliers (shown as green circles)

“maximum”: $Q3 + 1.5 * IQR$

“minimum”: $Q1 - 1.5 * IQR$

A box plot is a graph that gives you a good indication of how the values in the data are spread out. An advantage of boxplot is it taking up less space, which is useful when comparing distributions between many groups or datasets.

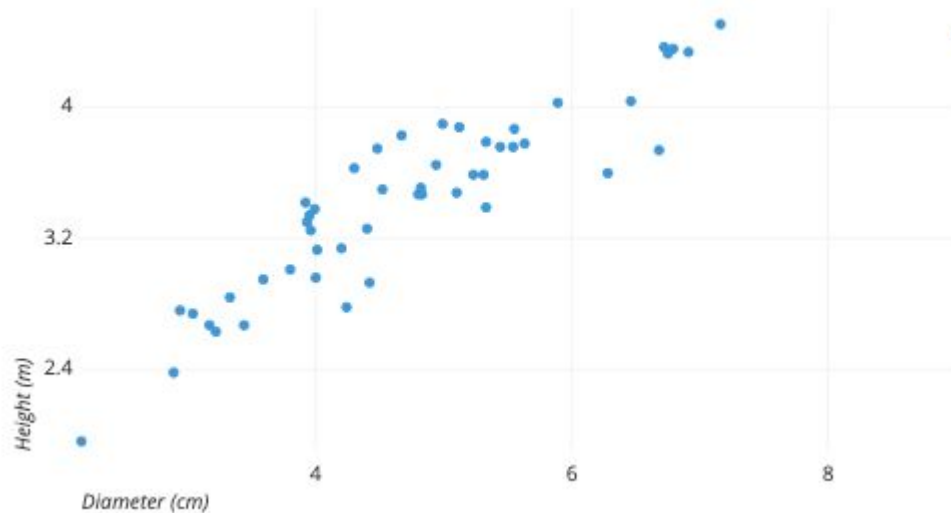
```
hs<-read.csv('https://raw.githubusercontent.com/mGalarnyk/Python_Tutorials/master/Kaggle/BreastCancerWisconsin/data/data.csv',  
header=T)
```

```
str(hs)
```


```
boxplot(Price~Status, data = hs, main="Boxplot of Price based on HS Status")
```

Scatter plot

A scatter plot (aka scatter chart, scatter graph) uses dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables.



When you should use a scatter plot



Scatter plots' primary uses are to observe and show relationships between two numeric variables. The dots in a scatter plot not only report the values of individual data points, but also patterns when the data are taken as a whole.

A scatter plot can also be useful for identifying other patterns in data. We can divide data points into groups based on how closely sets of points cluster together. Scatter plots can also show if there are any unexpected gaps in the data and if there are any outlier points. This can be useful if we want to segment the data into different parts, like in the development of user personas.

Relationship between the Price of the House and SQFT



```
plot(SQFT,Price, xlab="SQFT",ylab="Price", main="ScatterPlot of Price and SQFT")
```


When plotting a graph in R or using any statistical packages or language, it is necessary for your graph to be self-descriptive in order for stakeholders to be able to tell what is going on just with a first glance without you necessarily explaining to that person what you are trying to do.

Xlab represent the X axis

ylab represent the y axis

Main: is used to title the graph for easy and better understanding

Histogram



Histogram is used to plot continuous variable. It breaks the data into bins and shows frequency distribution of these bins. We can always change the bin size and see the effect it has on visualization. It's often used to see the distribution of a variable.

```
hist(hs$Price, xlab="Price", main="Histogram of Price")
```

Task 2

Given the following set of data

- Cancer.csv
- child_data.csv

Clean and plot the histogram and boxplot of both dataset and give a brief analysis.



The End of Session 2

Any Question?