ST2137: Statistical Computing and Programming

# Statistical Computing and Programming

Vik Gopal, Daisy Pham Thi Kim Cuc

2025-12-17

# Contents

**Contents** 2

**List of Figures** 9

**Preface** 12

**1 Introduction to R** 13

1.1 Introduction . . . 13
1.2 Installing R and Rstudio . . . 13
1.3 Basic Data Structures in R . . . 15
1.4 Creating Basic Objects in R . . . 15
1.4.1 Vectors . . . 15
1.4.2 Matrices . . . 17
1.4.3 Dataframes . . . 18
1.4.4 Lists . . . 20
1.5 Reading Data into R . . . 21
1.6 Accessing Parts of Dataframes . . . 23
1.7 Loops in R . . . 24
1.7.1 While Loops . . . 25
1.7.2 For Loops . . . 25
1.8 Redirecting R Output . . . 26
1.9 User-Defined Functions in R . . . 27
1.10 Miscellaneous . . . 27
1.10.1 Getting Help . . . 27
1.11 Installing Packages . . . 28
1.12 Further Readings . . . 28
1.13 Heads-Up on Differences with Python . . . 29
1.14 References . . . 29
1.14.1 Website References . . . 29

1.15 Exercises . . . . . 29

**2 Introduction to Python** **31**

2.1 Introduction . . . . . 31

2.2 Installing Python and Jupyter Lab . . . . . 31

2.3 Basic Data Structures in Python . . . . . 33

2.4 Slice Operator in Python . . . . . 34

2.5 Numpy Arrays . . . . . 35

2.6 Pandas DataFrames . . . . . 36

2.7 Reading Data into Python . . . . . 37

2.8 Subsetting DataFrames with Pandas . . . . . 38

2.9 Loops in Python . . . . . 39

2.10 User Defined Functions . . . . . 41

2.11 Miscellaneous . . . . . 41

2.11.1 Package installation . . . . . 41

2.11.2 Getting help . . . . . 41

2.12 Major Differences with R . . . . . 42

2.13 References . . . . . 42

2.13.1 Website References . . . . . 42

2.14 Exercises . . . . . 42

**3 Exploring Quantitative Data** **43**

3.1 Introduction . . . . . 43

3.2 Numerical Summaries . . . . . 45

3.3 Graphical Summaries . . . . . 48

3.3.1 Histograms . . . . . 48

3.3.2 Density Plots . . . . . 52

3.3.3 Boxplots . . . . . 54

3.3.4 QQ-plots . . . . . 56

3.4 Correlation . . . . . 59

3.5 Scatterplot Matrices . . . . . 61

3.6 References . . . . . 64

3.6.1 Website References . . . . . 64

3.7 Exercises . . . . . 64

**4 Exploring Categorical Data** **65**

4.1 Introduction . . . 65

4.2 Contingency Tables . . . 65

4.3 Visualisations . . . 66

4.3.1 Bar charts . . . 66

4.3.2 Mosaic plots . . . 68

4.3.3 Conditional Density Plots . . . 70

4.4 Tests for Independence . . . 71

4.4.1 $\chi^2$-Test for Independence . . . 72

4.4.2 Fisher's Exact Test . . . 75

4.4.3 $\chi^2$-Test for $r \times c$ Tables . . . 77

4.5 Measures of Association . . . 78

4.5.1 Odds Ratio . . . 78

4.5.2 For Ordinal Variables . . . 79

4.6 Further readings . . . 81

4.7 References . . . 82

4.7.1 Website References . . . 82

4.8 Exercises . . . 82

**5 Robust Statistics** **83**

5.1 Introduction . . . 83

5.2 Notation . . . 83

5.3 Datasets . . . 84

5.4 Assessing Robustness . . . 85

5.4.1 Asymptotic Relative Efficiency . . . 86

5.5 Requirements of Robust Summaries . . . 87

5.6 Measures of Location . . . 87

5.6.1 M-estimators . . . 87

5.6.2 Trimmed mean . . . 89

5.6.3 Winsorised Mean . . . 89

5.7 Measures of Scale . . . 90

5.7.1 Sample Standard Deviation . . . 90

5.7.2 Median Absolute Deviation . . . 90

5.7.3 Interquartile Range . . . 91

5.8 Examples . . . 92
5.9 Summary . . . 93
5.10 References . . . 94
5.10.1 Website References . . . 94
5.11 Exercises . . . 94

**6 Introduction to SAS** **95**
6.1 Introduction . . . 95
6.2 Registering for a SAS Studio Account . . . 95
6.3 An Overview of SAS Language . . . 96
6.4 Basic Rules for SAS Programs . . . 97
6.4.1 For SAS statements . . . 97
6.4.2 For SAS names . . . 98
6.4.3 For SAS variables . . . 98
6.5 Reading Data into SAS . . . 98
6.6 Uploading and Using Datasets . . . 99
6.7 Summarising Numerical Data . . . 101
6.7.1 Numerical Summaries . . . 101
6.7.2 Scatter Plots . . . 101
6.7.3 Histograms . . . 102
6.7.4 Boxplots . . . 103
6.7.5 QQ-plots . . . 105
6.8 Categorical Data . . . 105
6.9 References . . . 107
6.10 Website References . . . 107

**7 Two-sample Hypothesis Tests** **108**
7.1 Introduction . . . 108
7.2 Procedure for Significance Tests . . . 108
7.2.1 Step 1: Assumptions . . . 108
7.2.2 Step 2: State the hypotheses and significance level . . . 108
7.2.3 Step 3: Compute the test statistic . . . 109
7.2.4 Step 4: Compute the $p$-value . . . 109
7.2.5 Step 5: State your conclusion . . . 110
7.3 Confidence Intervals . . . 110

7.4 Parametric Tests . . . 110
7.4.1 Independent Samples Test . . . 111
7.4.2 More on Assessing Normality . . . 116
7.4.3 Paired Sample Test . . . 118
7.5 Non-parametric Tests . . . 121
7.5.1 Independent Samples Test . . . 121
7.5.2 Paired Samples Test . . . 124
7.6 Summary . . . 126
7.7 References . . . 126
7.7.1 Website References . . . 126
7.8 Exercises . . . 126

**8 ANOVA** **127**
8.1 Introduction . . . 127
8.2 One-Way Analysis of Variance . . . 129
8.2.1 Formal Set-up . . . 129
8.2.2 $F$-Test in One-Way ANOVA . . . 130
8.2.3 Assumptions . . . 130
8.3 Comparing specific groups . . . 135
8.4 Contrast Estimation . . . 137
8.5 Multiple Comparisons . . . 139
8.5.1 Bonferroni . . . 139
8.5.2 TukeyHSD . . . 139
8.6 Kruskal-Wallis Procedure . . . 141
8.6.1 Formal Set-up . . . 141
8.7 Summary . . . 143
8.8 References . . . 143
8.8.1 Website References . . . 143

**9 Linear Regression** **145**
9.1 Introduction . . . 145
9.2 Simple Linear Regression . . . 147
9.2.1 Formal Set-up . . . 147
9.2.2 Estimation . . . 147

9.3 Hypothesis Test for Model Significance . . . 149
9.3.1 Coefficient of Determination, $R^2$ . . . 150
9.4 Multiple Linear Regression . . . 157
9.4.1 Formal Setup . . . 157
9.4.2 Estimation . . . 158
9.4.3 Coefficient of Determination, $R^2$ . . . 158
9.4.4 Hypothesis Tests . . . 158
9.5 Indicator Variables . . . 162
9.5.1 Including a Categorical Variable . . . 162
9.6 Interaction term . . . 165
9.7 Residual Diagnostics . . . 168
9.7.1 Standardised Residuals . . . 168
9.7.2 Normality . . . 169
9.7.3 Scatterplots . . . 170
9.7.4 Influential Points . . . 172
9.8 Further Reading . . . 174
9.9 References . . . 175
9.9.1 Website References . . . 175

**10 Simulation** **176**
10.1 Introduction . . . 176
10.2 Theory . . . 177
10.3 Generating Random Variables in R and Python . . . 178
10.4 Monte-Carlo Integration . . . 180
10.5 Simulation Studies . . . 181
10.5.1 Confidence Intervals . . . 182
10.5.2 Type I Error . . . 183
10.5.3 Newspaper Inventory . . . 184
10.6 Resampling Methods . . . 185
10.6.1 Permutation Test . . . 185
10.6.2 Bootstrapping . . . 186
10.7 Summary . . . 188
10.8 References . . . 188
10.8.1 Website References . . . 188

# List of Figures


1.1 R: Base R GUI . . . . . 14
1.2 R: Rstudio interface . . . . . 15

2.1 Py: JupyterLab interface . . . . . 32
2.2 Py: Positive and negative indexing . . . . . 35

3.1 Data types . . . . . 43
3.2 Three vastly different datasets, all with mean 0 and s.d. 1 . . . . . 48
3.3 Histogram with outliers . . . . . 49
3.4 Bimodal histogram . . . . . 49
3.5 Skewed histograms . . . . . 49
3.6 R: Histogram of G3 scores . . . . . 50
3.7 Py: Histogram of G3 scores . . . . . 50
3.8 R: Lattice plot of G3 scores by Medu . . . . . 51
3.9 Py: Lattice plot of G3 scores by Medu . . . . . 52
3.10 R: Estimated densities for G3 scores, by Medu . . . . . 53
3.11 Py: Estimated densities for G3 scores, by Medu . . . . . 53
3.12 Boxplot construction . . . . . 54
3.13 R: Boxplot of G3 scores, by gout variable . . . . . 55
3.14 Py: Boxplot of G3 scores, by gout variable . . . . . 56
3.15 QQ-plots . . . . . 56
3.16 Comparing data with reference Normal (blue) . . . . . 58
3.17 R: QQ plot for compressive strength . . . . . 58
3.18 Py: QQ plot for compressive strength . . . . . 59
3.19 Varying linear relations and corresponding $r$-values . . . . . 60
3.20 Non-linear relations and corresponding $r$-values . . . . . 60
3.21 R: Scatterplot matrix for concrete dataset . . . . . 61
3.22 Py: Scatterplot matrix for concrete dataset . . . . . 62
3.23 R: Heatmap of correlation matrix for concrete data . . . . . 63

4.1 R: Stacked barchart for political association data . . . . . 67
4.2 Py: Barchart for Claritin data . . . . . 68
4.3 R: Mosaic plot for political association data . . . . . 69
4.4 Py: Mosaic plot for political association data . . . . . 70
4.5 R: Spine plot for heart failure data . . . . . 71
4.6 R: Conditional density plot for heart failure data . . . . . 71

5.1 Copper measurements dataset . . . . . 84
5.2 Self-awareness study timing . . . . . 85
5.3 Contaminated Normal . . . . . 88

6.1 SAS: Login page . . . . . 95
6.2 SAS: Studio interface . . . . . 96
6.3 SAS: Simple data creation . . . . . 97
6.4 SAS: Inline data created . . . . . 98
6.5 SAs: PROC MEANS output . . . . . 98
6.6 SAS: New library creation icon . . . . . 100
6.7 SAS: Upload data icon . . . . . 100
6.8 SAS: Common ST2137 procedures . . . . . 101
6.9 SAS: 5-number-summaries and boxplots for bike rental data . . . . . 102
6.10 SAS: Scatterplot for bike rental data . . . . . 102
6.11 SAS: Histograms for bike rental data . . . . . 103
6.12 SAS: Boxplots for bike rentals, by season . . . . . 104
6.13 SAS: Boxplots for bike rentals on log-scale, by season . . . . . 104
6.16 SAS: Observed & expected counts, student performance data . . . . . 106
6.17 SAS: Test statistics and p-values for student performance data . . . . . 106
6.18 SAS: Mosaic plot for student performance data . . . . . 106
6.19 SAS: Categorical variables' association, Walc by Dalc . . . . . 107

7.1 SAS: Independent samples t-test, Abalone data . . . . . 113
7.2 R: Histograms for males and females . . . . . 115
7.3 R: QQ-plot for females . . . . . 115
7.4 R: QQ-plot for males . . . . . 115
7.5 SAS: Normality tests . . . . . 118
7.6 SAS: HR paired t-test . . . . . 120

7.7 SAS: Paired profiles . . . . . 121
7.8 SAS: Agreement plot . . . . . 121
7.9 SAS: Wilcoxon rank sum output . . . . . 123
7.10 SAS: Signed rank test output . . . . . 125

8.1 . . . . . 128

9.1 Scatterplot with simple linear regression model . . . . . 146
9.2 Scatterplot of registered vs. casual bike renters . . . . . 146

# Preface

> One more robot learns to feel,
>
> Something more than a machine..

The lyrics above are from the song "One More Robot/Sympathy 3000-21", written by The Flaming Lips in 2002. When I started preparing to teach this course in mid-2023, AI had been around for a year or two already. I was stuck for a long time wondering if this course is still relevant; I played the song quite a few times..

Here's what I mean: The course is meant to introduce statistics majors (and minors) to three computing languages: R, Python and SAS. But.. why does anyone need to know this when we can just ask Gemini? You can read more about this course on NUSmods.

If you are taking this course, should you pretend that AI does not exist? Obviously not! If you really need help, don't hesitate to use AI. But if you do use AI, I would hope that you take some time to dissect it's solution to aid in more long-term understanding and recall.

I believe there is a need for data analysts to know how to code from scratch, if only because of the value of the process. By coding from nought, and by digging into data ourselves, we sharpen our minds.

This textbook is used in a 13-week semester at the National University of Singapore NUS. As mentioned, the course introduces R, Python and SAS to statistics majors. If you are an instructor and are planning to use it, but only plan to use one of the languages (R or Python only), the flow of material will still work - just leave out chapters 1 or 2, and chapter 6. Chapters 3 to 10 contain R and Python code, while chapters 6 to 9 contain SAS code as well. In general, later chapters build on results and findings from earlier chapters.

If you are a student taking the course, try to go through the material in the book on your own. Lectures will contain more in-depth explanations of concepts and code. As part of your revision, try to give yourself small problems, that build on the examples in the book. Challenging yourself in this way will improve you as a data analysts.

It's ok if you cannot solve every problem that you have set for yourself. There is great value in simply struggling with a problem. As part of that journey, you will learn *some* skills that will be useful in *other* problems. I sincerely hope you find joy in working with data, R, Python and SAS during this course!

If you find something useful in this textbook, please do let me know at vik.gopal@nus.edu.sg. The book, along with slides and scripts/notebooks can all be found in the github repository. If you need more details about anything, do feel free to write as well.

So long, and thanks for reading!

Vik

https://blog.nus.edu.sg/stavg

# 1 Introduction to R

## 1.1 Introduction

R evolved from the **S** language, which was first developed by Rick Becker, John Chambers and Allan Wilks. S was created with the following goals in mind:

1. S was conceived as a powerful tool for statistical modelling. It enabled you to specify and fit statistical models to your data, assess the goodness of fit and then display parameter estimates, standard errors and predicted values derived from the model. It provided the means to define and manipulate data. The intent was to provide the user with maximum control over the model-fitting process.
2. S was to be used for data exploration (tabulating, sorting data, drawing plots to look for trends, etc.)
3. It was to be used as a sophisticated calculator to evaluate complex arithmetic expressions, and as a very flexible and general object-oriented programming language to perform more extensive data manipulation.

Later on, S evolved into S-PLUS, which then became rather costly. **R**oss Ihaka and **R**obert Gentleman from the University of Auckland decided to write a stripped-down version of S. They named the new language R. Five years later, version 1.0.0 of R was released on 29 Feb 2000. As of Dec 2025, the latest version of R is 4.5.2 It is maintained by the (R Core Team 2025).

## 1.2 Installing R and Rstudio

To download R, go to CRAN, the Comprehensive R Archive Network, download the version for your operating system and install it.

A new major version is released once a year, and there are 2 - 3 minor releases each year. Upgrading is painful, but it gets worse if you wait to upgrade.

> **!** Important
>
> For our class, please ensure that you have version 4.5.1 or later. Functions in older versions work differently, so you might face problems or differences with some of the codes in the notes.

After installing, you can start using R straightaway. However, the basic GUI is not very user-friendly, as you can see from Figure 1.1. Instead of using the basic GUI for R, we are going to use RStudio. RStudio is an Integrated Development Environment (IDE) for R. It provides several features that base R does not, including:

- A history of previous plots made.
- The ability to browse the objects in our workspace more easily.

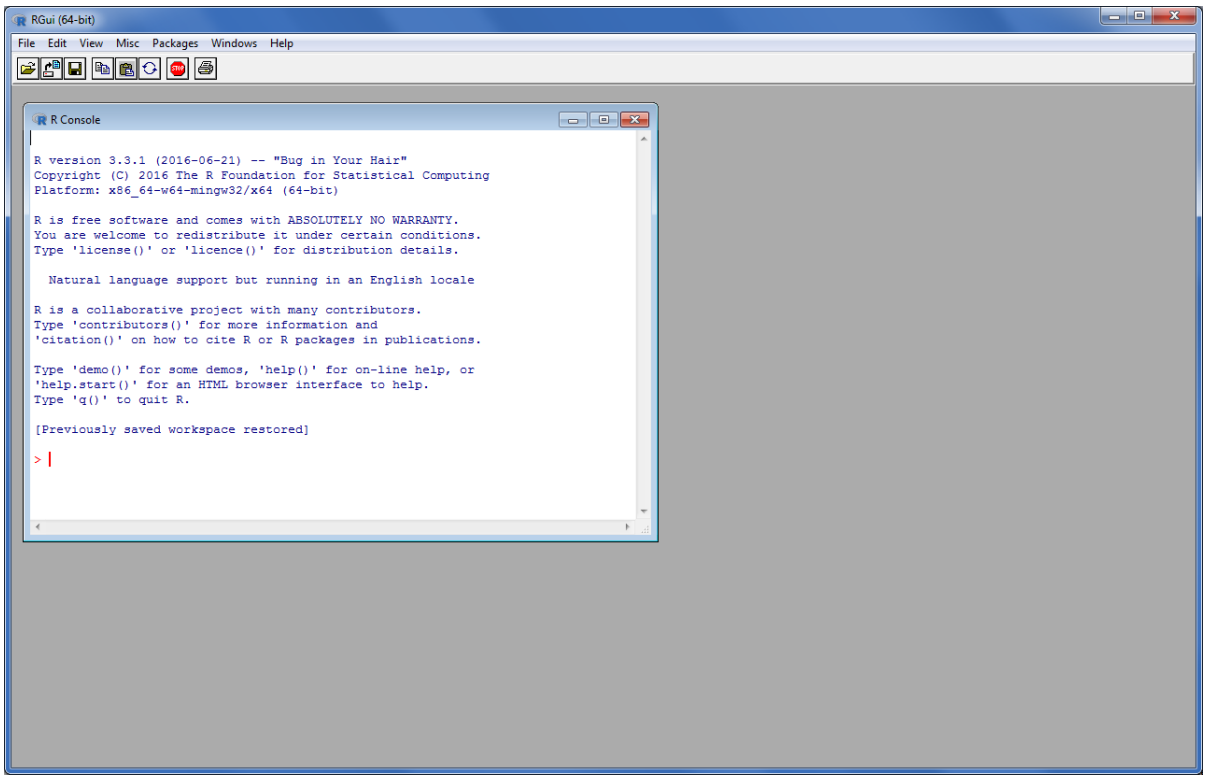



Figure 1.1: R: Base R GUI

The installation file for RStudio can be obtained from this URL. It is updated a couple of times a year. Make sure you have at least version 2025.09.x for our course.

Here's a quick orientation of the panels in Rstudio, with reference to Figure 1.2.

- Panel 1 is the **console**.
  - This is where you type R commands.
  - The output from these commands or functions will also be seen here.
  - Use the ↑ key to scroll through previously entered commands.
- Panel 2 contains the *History* and *Environment* tabs.
  - The *History* tab displays all commands that have been previously entered in the current session.
  - These commands can be sent directly to the source code panel or the console panel.
  - The *Environment* tab in this panel has a list of items that have been created in the current session.
- Panel 3 contains the *Files*, *Plots* and *Help* tabs.
  - The *Files* tab contains a directory structure that allows one to choose and open files in the source code editor.
  - Through the *Plots* tab, one can access all plots that have been created in the current session.
  - The *Help* tab displays the documentation for R functions.
- Panel 4 contains the source code editor.
  - This where you edit R scripts.

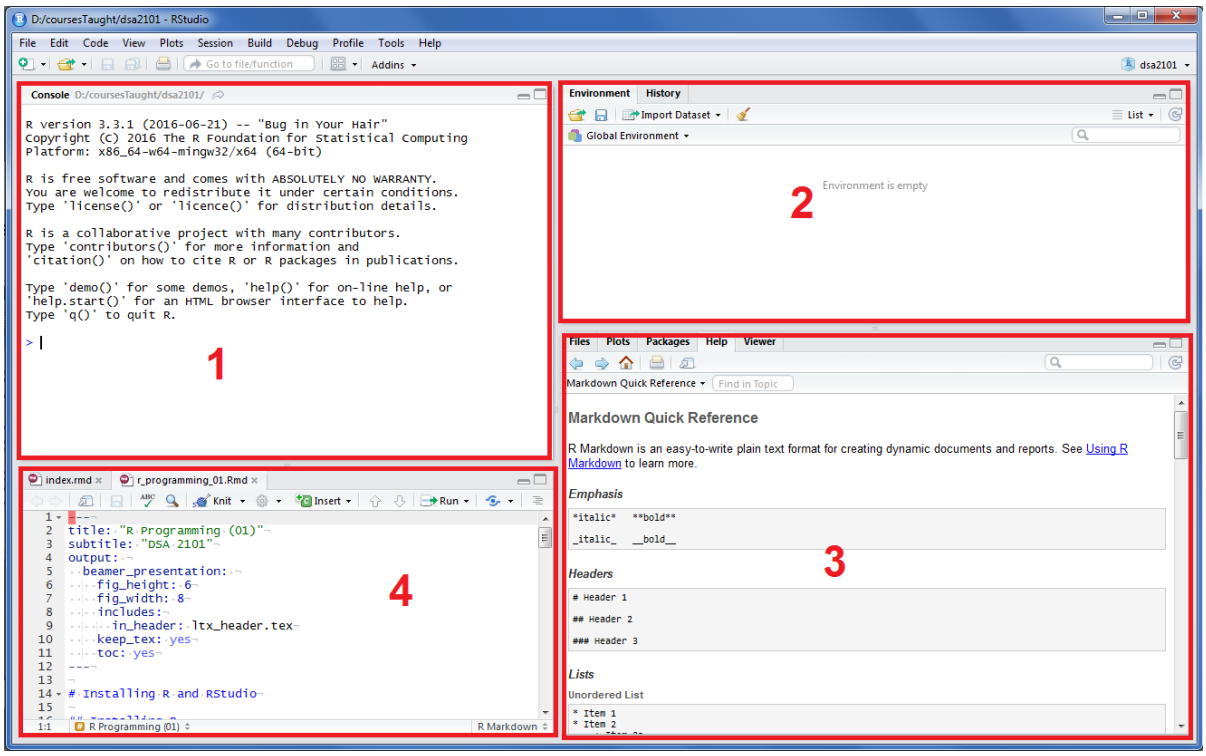



Figure 1.2: R: Rstudio interface

- You can hit **Ctrl-Enter** to execute a command (in the console panel) while your cursor is in the source code panel.
- You can also highlight code and the editor and execute it directly in the console panel.

## 1.3 Basic Data Structures in R

Probably the four most frequently used data structures in R are the following:

- **Vector**: A set of elements of the same mode (logical; numeric; character; factor).
- **Matrix**: A set of elements appearing in rows and columns, where the elements are of the same mode.
- **Dataframe**: This is similar to the matrix object in that it is 2-dimensional. However, columns in a dataframe can have different modes. Rows typically contain different observations from your study or measurements from your experiment. The columns contain the values of different variables which may be of different modes.
- **List**: A list is a generalization of a vector – it represents a collection of data objects.

## 1.4 Creating Basic Objects in R

### 1.4.1 Vectors

To create a vector in R, the simplest way is to use the combine function `c()`.

```r
#creating a vector of numbers:
numeric_vec <- c(2,4,6,8,10)
numeric_vec
```

```
[1]  2  4  6  8 10
```

```r
# creating a vector of strings/characters:
string_vec <-c("weight", "height", "gender")
string_vec
```

```
[1] "weight" "height" "gender"
```

```r
# creating a Boolean vector (T/F):
logical_vec <- c(TRUE, TRUE, FALSE)
logical_vec
```

```
[1]  TRUE  TRUE FALSE
```

```r
# creating factors:
factors_vec <- factor(c("male", "male", "female"))
factors_vec
```

```
[1] male   male   female
Levels: female male
```

Factors are slightly different from strings. In R, they are used to represent categorical variables in linear models.

When we need to create a vector that defines groups (of Females followed by Males, for instance), we can turn to a convenient function called `rep()`. This function replicates elements of vectors and lists. The syntax is as follows: `rep(a, b)` will replicate the item `a`, `b` times. Here are some examples:

```r
r1 <- rep(2,3)
r1
```

```
[1] 2 2 2
```

```r
r2 <- rep(c(1,2),3)
r2
```

```
[1] 1 2 1 2 1 2
```

```r
r3 <- rep(c(6,3),c(2,4))
r3
```

```
[1] 6 6 3 3 3 3
```

```r
r4 <- rep(string_vec, 2)
r4
```

```
[1] "weight" "height" "gender" "weight" "height" "gender"
```

On other occasions, we may need to create an index vector, along the rows of a dataset. The `seq()` function is useful for this purpose. It creates a sequence of numbers that are evenly spread out.

```r
seq(from=2, to=10, by=2)
```

```
[1]  2  4  6  8 10
```

```r
seq(from=2, to=10, length = 5)
```

```
[1]  2  4  6  8 10
```

```r
seq(2, 5, 0.8)
```

```
[1] 2.0 2.8 3.6 4.4
```

```r
seq(2, 5, 0.8) * 2
```

```
[1] 4.0 5.6 7.2 8.8
```

The final example above, where the sequence vector of length 4 is multiplied by a scalar 2, is an example of the *recycling rule* in R – the shorter vector is recycled to match the length of the longer one. This rule applies within all built-in R functions. Try to use this rule to your advantage when using R.

If you only need to create a vector of integers that increase by 1, you do not even need `seq()`. The `:` colon operator will handle the task.

```r
s1 <- 2:5
s1
```

```
[1] 2 3 4 5
```

### 1.4.2 Matrices

Thus far, we have been creating vectors. Matrices are higher dimensional objects. To create a matrix, we use the `matrix()` function. The syntax is as follows: `matrix(v,r,c)` will take the values from vector `v` and create a matrix with `r` rows and `c` columns. R is *column-major*, which means that, by default, the matrix is filled column-by-column, not row-by-row.

```r
v <- c(1:6)
m1 <- matrix(v, nrow=2, ncol=3)
m1
```

```
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```r
m2 <- matrix(v, nrow=2, ncol=3, byrow=TRUE)
m2
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

New rows (or columns) can be added to an existing matrix using the command `rbind()` (resp. `cbind()`).

```r
a <- c(1,2,3,4)
b <- c(5,6,7,8)
ab_row <- rbind(a,b)
ab_row
```

```
  [,1] [,2] [,3] [,4]
a    1    2    3    4
b    5    6    7    8
```

```r
ab_col <- cbind(m1, c(7,8))
ab_col
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```

### 1.4.3 Dataframes

Now let's turn to dataframes, which are the most common object we are going to use for storing data in R. A dataframe is a tabular object, like a matrix, but the columns can be of different types; some can be numeric and some can be character, for instance. Think of a dataframe as an object with rows and columns:

- The rows contain different **observations or measurements**;
- The columns contain the values of different **variables**.

As a general guideline, we should try to store our data in a format where *a single variable is not spread across columns.*

**Example 1.1** (Tidy Data)**.** Consider an experiment where there are three treatments (control, pre-heated and pre-chilled), and two measurements per treatment. The response variable is store in the following dataframe:

| Control | Pre_heated | Pre_chilled |
|---|---|---|
| 6.1 | 6.3 | 7.1 |
| 5.9 | 6.2 | 8.2 |

The above format is probably convenient for recording data. However, the response variable has been spread across three columns. The *tidy* version of the dataset is:

| Response | Treatment |
|---|---|
| 6.1 | control |
| 5.9 | control |
| 6.3 | pre_heated |
| 6.2 | pre_heated |
| 7.1 | pre_chilled |
| 8.2 | pre_chilled |

The second version is more amenable to computing conditional summaries, making plots and for modeling in R.

Dataframes can be created from matrices, and they can also be created from individual vectors. The function `as.data.frame()` converts a matrix into a dataframe, with generic column names assigned.

```
df1 <- as.data.frame(m1)
df1
```

```
  V1 V2 V3
1  1  3  5
2  2  4  6
```

If we intend to pack individual vectors into a dataframe, we use the function `data.frame()`. We can also specify custom column names when we call this function.

```
a <- c(11,12)
b <- c(13,14)
df2 <- data.frame(col1 = a, col2 = b)
df2
```

```
  col1 col2
1   11   13
2   12   14
```

### 1.4.4 Lists

Finally, we turn to lists. You can think of a list in R as a very general basket of objects. The objects do not have to be of the same type or length. The objects can be lists themselves. Lists are created using the `list( )` function; elements within a list are accessed using the `$` notation, or by using the names of the elements in the list.

```
ls1 <- list(A=seq(1, 5, by=2), B=seq(1, 5, length=4))
ls1
```

```
$A
[1] 1 3 5

$B
[1] 1.000000 2.333333 3.666667 5.000000
```

```
ls1[[2]]
```

```
[1] 1.000000 2.333333 3.666667 5.000000
```

```
ls1[["B"]]
```

```
[1] 1.000000 2.333333 3.666667 5.000000
```

```
ls1$A
```

```
[1] 1 3 5
```

**Example 1.2** (Extracting $p$-values)**.** The `iris` dataset is a very famous dataset that comes with R. It contains measurements on the flowers of three different species. Let us conduct a 2-sample $t$-test, and extract the $p$-value.

```
setosa <- iris$Sepal.Length[iris$Species == "setosa"]
virginica <- iris$Sepal.Length[iris$Species == "virginica"]

t_test_out <- t.test(setosa, virginica)
str(t_test_out)
```

```
List of 10
 $ statistic  : Named num -15.4
  ..- attr(*, "names")= chr "t"
 $ parameter  : Named num 76.5
  ..- attr(*, "names")= chr "df"
 $ p.value    : num 3.97e-25
 $ conf.int   : num [1:2] -1.79 -1.38
  ..- attr(*, "conf.level")= num 0.95
 $ estimate   : Named num [1:2] 5.01 6.59
```

```
  ..- attr(*, "names")= chr [1:2] "mean of x" "mean of y"
 $ null.value : Named num 0
  ..- attr(*, "names")= chr "difference in means"
 $ stderr     : num 0.103
 $ alternative: chr "two.sided"
 $ method     : chr "Welch Two Sample t-test"
 $ data.name  : chr "setosa and virginica"
 - attr(*, "class")= chr "htest"
```

`str( )` prints the *structure* of an R object. From the output above, we can tell that the output object is a list with 10 elements. The particular element we need to extract is `p.value`.

```
t_test_out$p.value
```

```
[1] 3.966867e-25
```

## 1.5 Reading Data into R

It is uncommon that we will be creating dataframes by hand, as we have been doing so far. It is more likely that we will be reading in a dataset from a file in order to perform analysis on it. Thus, at this point, let's sidetrack a little and discuss how we can read data into R as a dataframe.

The two most common functions for this purpose are `read.table()` and `read.csv()`. The former is used when our data is contained in a text file, with spaces or tabs separating columns. The latter function is for reading in files with comma-separated-values. If our data is stored as a text file, it is always a good idea to open it and inspect it before getting R to read it in. Text files can be opened with any text editor; `csv` files can also be opened by Microsoft Excel. When we do so, we should look out for a few things:

- Are there column names in the first row, or does the data actually begin in line 1?
- Is it spaces or commas that separate columns?
- Are there trailing values in the last few lines of the file?

The file `crab.txt` contains measurements on crabs. If you open up the file outside of R, you should observe that the first line of the file contains column names. By the way, `head()` is a convenient function for inspecting the first few rows of a dataframe. There is a similar function `tail()` for inspecting the *last* few rows.

```
data1 <- read.table("data/crab.txt")
head(data1)
```

```
     V1    V2    V3     V4     V5
1 color spine width satell weight
2     3     3  28.3      8  3.050
3     4     3  22.5      0  1.550
4     2     1  26.0      9  2.300
5     4     3  24.8      0  2.100
6     4     3  26.0      4  2.600
```

The data has not been read in correctly. To fix this, we need to inform R that the first row functions as the column names/headings.

```
data1 <- read.table("data/crab.txt", header=TRUE)
head(data1)
```

```
  color spine width satell weight
1     3     3  28.3      8   3.05
2     4     3  22.5      0   1.55
3     2     1  26.0      9   2.30
4     4     3  24.8      0   2.10
5     4     3  26.0      4   2.60
6     3     3  23.8      0   2.10
```

If the first line of the data file does not contain the names of the variables, we can create a vector beforehand to store and then use the names.

```
varnames <- c("Subject", "Gender", "CA1", "CA2", "HW")
data2 <- read.table("data/ex_1.txt", header = FALSE,
                    col.names = varnames)
data2
```

```
  Subject Gender CA1 CA2 HW
1      10      M  80  84  A
2       7      M  85  89  A
3       4      F  90  86  B
4      20      M  82  85  B
5      25      F  94  94  A
6      14      F  88  84  C
```

The use of `read.csv()` is very similar, but it is applicable when the fields within each line of the input file are separated by commas instead of tabs or spaces.

```
data3 <- read.csv("data/ex_1_comma.txt",  header = FALSE)
data3
```

```
  V1 V2 V3 V4 V5
1 10  M 80 84  A
2  7  M 85 89  A
3  4  F 90 86  B
4 20  M 82 85  B
5 25  F 94 94  A
6 14  F 88 84  C
```

## 1.6 Accessing Parts of Dataframes

We now turn to the task of accessing a subset of rows and/or columns of a dataframe. The notation uses rectangular brackets, along with a comma inside these brackets to demarcate the row and column specifiers.

To access all rows from a particular set of columns, we leave the row column specification empty.

```
data3[, 2:4]
```

```
  V2 V3 V4
1  M 80 84
2  M 85 89
3  F 90 86
4  M 82 85
5  F 94 94
6  F 88 84
```

To retrieve a subset of rows, we use the space *before* the comma.

```
data3[1:3, ]
```

```
  V1 V2 V3 V4 V5
1 10  M 80 84  A
2  7  M 85 89  A
3  4  F 90 86  B
```

In R, dataframes are special kinds of lists. Hence, individual columns can be retrieved from a dataframe (as a vector) using the `$` operator. These columns can then be used to retrieve only the rows that satisfy certain conditions. In order to achieve this task, we turn to logical vectors. The following code returns only the rows corresponding to Gender equal to "M".

```
data2[data2$Gender == "M", ]
```

```
  Subject Gender CA1 CA2 HW
1      10      M  80  84  A
2       7      M  85  89  A
4      20      M  82  85  B
```

Logical vectors contain `TRUE`/`FALSE` values in their components. These vectors can be combined with `&` (AND) and `|` (OR) operations to yield only the rows that satisfy all conditions. Below, we return all rows where Gender is equal to "M" and CA2 is greater than 85.

```
data2[data2$Gender == "M" & data2$CA2 > 85, ]
```

```
  Subject Gender CA1 CA2 HW
2       7      M  85  89  A
```

For your reference , the table below contains all the logical operators in R.

| Operator | Description |
|---|---|
| `<` | less than |
| `>` | greater than |
| `<=` | less than or equal to |
| `>=` | greater than or equal to |
| `==` | equal to |
| `!=` | not equal to |
| `x \| y` | (vectorised) OR |
| `x & y` | (vectorised) AND |

The `$` operator is both a getter and a setter, which means we can also use it to add new columns to the dataframe. The following command creates a new column named `id`, that contains a running sequence of integers beginning from 1; the new column essentially contains row numbers.

```
data2$id <- 1:NROW(data2)
```

Before we leave this section on dataframes, we shall touch on how we can rearrange the dataframe according to particular columns in either ascending or descending order.

```
data2[order(data2$CA1), ]
```

```
  Subject Gender CA1 CA2 HW id
1      10      M  80  84  A  1
4      20      M  82  85  B  4
2       7      M  85  89  A  2
6      14      F  88  84  C  6
3       4      F  90  86  B  3
5      25      F  94  94  A  5
```

```
# arranges in reverse order:
data2[order(data2$CA1, decreasing = TRUE), ]
```

```
  Subject Gender CA1 CA2 HW id
5      25      F  94  94  A  5
3       4      F  90  86  B  3
6      14      F  88  84  C  6
2       7      M  85  89  A  2
4      20      M  82  85  B  4
1      10      M  80  84  A  1
```

## 1.7 Loops in R

`while` loops execute a set of instructions as long as a particular condition holds true. On the other hand, `for` loops iterate over a set of items, executing the instructions a fixed number of times.

### 1.7.1 While Loops

The syntax for a `while` loop is as follows. The condition is checked at the beginning of each iteration. As long as it is TRUE, the block of R expressions within the curly braces will be executed.

```
while( <logical condition> ) {
  R expressions
  ...
}
```

Here is an example of a `while` loop that increments a value until it reaches 10.

```
x <- 0
S <- 0
while(x<=10) {
  S <- S + x
  x <- x + 1
}
S
```

```
[1] 55
```

### 1.7.2 For Loops

The general syntax for a `for` loop is as follows:

```
for(<index> in <set> ) {
  R expressions
  ...
}
```

The "set" can be a sequence generated by the colon operator, or it can be any vector. Here is the same code from the `while` loop. Notice that `x` does not have to be initialised.

```
S <- 0
for(x in 1:10){
  S <- S + x
}
S
```

```
[1] 55
```

As a second example, consider these lines of R code, which prints out all squares of integers from 1 to 5. The `cat( )` function concatenates a given sequence of strings and prints them to the console. We shall see more of it in Section 1.8.

```r
x <- 0
test <- TRUE

while(test) {
  x <- x+1
  test <- x<6
  cat(x^2, test, "\n")
}
```

```
1 TRUE
4 TRUE
9 TRUE
16 TRUE
25 TRUE
36 FALSE
```

## 1.8 Redirecting R Output

The `cat()` function can be used to print informative statements as our loop is running. This can be very helpful in debugging our code. It works by simply joining any strings given to it, and then printing them out to the console. The argument `"\n"` instructs R to print a newline character after the strings.

```r
cat("The current number is", x^2, "\n")
```

```
The current number is 36
```

When we are running a job in the background, we may want the output to print to a file so that we can inspect it later at our convenience. That is where the `sink()` function comes in.

```r
sink("data/datasink_ex1.txt")      # turn the sink on
x <- 0
test <- TRUE

while(test) {
  x <- x+1
  test <- isTRUE(x<6)
  cat(x^2, test, "\n")             # This will be written to the file.
}

sink()                             # turn the sink off
```

When we have finished working with a dataframe, we may want save it to a file. For this purpose, we can use the following code. Once executed, the dataframe `data2` will be written to a csv file named `ex_1_with_IQ.csv` in the `data/` directory.

```
write.csv(data2, "data/ex_1_with_IQ.csv")
```

## 1.9 User-Defined Functions in R

We have already seen several useful functions in R, e.g. `read.csv` and `head`. Here is a list of other commonly used functions.

| Function | Description |
|---|---|
| `max(x)` | Maximum value of x |
| `min(x)` | Minimum value of x |
| `sum(x)` | Total of all the values in x |
| `mean(x)` | Arithmetic average values in x |
| `median(x)` | Median value of x |
| `range(x)` | Vector of length 2: min(x), max(x) |
| `var(x)` | Sample variance of x |
| `cor(x, y)` | Correlation between vectors x and y |
| `sort(x)` | Sorted version of x |

R is a fully-fledged programming language, so it is also possible to write our own functions in R. To define a function for later use in R, the syntax is as follows:

```
fn_name <- function(arguments) {
  R expressions
  ...
  Returned object

}
```

The final line of the function definition will determine what gets returned when the function is executed. Here is an example of a function that computes the circumference of a circle of given radius.

```
circumference <- function(r) {
  2*pi*r
}
circumference(1.3)
```

```
[1] 8.168141
```

## 1.10 Miscellaneous

### 1.10.1 Getting Help

All functions in R are documented. When you need to find out more about the arguments of a function, or if you need examples of code that is guaranteed to work, then do look up the help

page (even before turning to stackexchange). The help pages within R are accessible even if you are offline.

To access the help page for a particular function, use the following command:

```
?mean
```

If you are not sure about the name of the function, you can use the following fuzzy search operator to return a page with a list of matching functions:

```
??mean
```

## 1.11 Installing Packages

R is an open-source software. Many researchers and inventors of new statistical methodologies contribute to the software through packages[1]. At last check (Dec 2025), there are 23067 such packages. The packages can be perused by name, through this link.

To install one of these packages, you can use `install.packages()`. For instance, the following command will install `stringr` (a package for string manipulations) and all its dependencies on your machine.

```
install.packages("stringr")
```

Once the installation is complete, we still need to *load* the package whenever we wish to use the functions within it. This is done with the `library()` function:

```
library(stringr)
```

To access a list of all available functions from a package, use:

```
help(package="stringr")
```

## 1.12 Further Readings

In our course, we will only be using basic R syntax, functions and plots. You may have heard of the `tidyverse` set of packages, which are a suite of packages that implement a particular paradigm of data manipulation and plotting. You can read and learn more about that approach by taking DSA2101, or by learning from Wickham, Çetinkaya-Rundel, and Grolemund (2023).

The DataCamp courses cover a little more on R e.g. use of `apply` family of functions. These will be included in our course, so please pay close attention in the DataCamp course.

[1] packages are simply *collections* of functions.

## 1.13 Heads-Up on Differences with Python

If you are coming from a Python background, please remember the following key differences:

- The colon operator in R is not a slice operator (like in Python)
- A list in R is similar to Python in that it is a generic collection, but accessing the elements is done with a `$` notation.
- The assignment operator in R is `<-`, but in Python it is `=`.
- To create vectors in R, you need to use `c( )`.

## 1.14 References

### 1.14.1 Website References

1. Iris data: More information on this classic dataset.
2. Installing R
3. Installing Rstudio

## 1.15 Exercises

1. Use the `rep()` function to create the following vector:

```
 [1] 1 1 1 2 2 3 1 1 1 2 2 3
```

2. What do the functions `saveRDS()` and `readRDS()` do?
3. From the `data2` dataframe in Section 1.5, return all rows where either `Gender` is equal to "M", or `HW` is "A".
4. Create a list object that, when printed, returns:

```
$`1`
$`1`[[1]]
[1] "Will"

$`1`[[2]]
[1] "Mike"

$`1`[[3]]
[1] "Dustin"

$`1`[[4]]
[1] "Lucas"


[[2]]
[1] "Max"
```

5. Why does this line of code return an error? When should `&&` be used?

```r
data2[data2$Gender == "M" && data2$CA2 > 85, ]
```

# 2 Introduction to Python

## 2.1 Introduction

Python is a general-purpose programming language. It is a higher-level language than C, C++ and Java in the sense that a Python program does not have to be compiled before execution.

It was originally conceived back in the 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands. The language is named after a BBC TV show (Guido's favorite program) "Monty Python's Flying Circus", not the snake!

Python reached version 1.0 in January 1994. Python 2.0 was released on October 16, 2000. Python 3.0, which is backwards-*incompatible* with earlier versions, was released on 3 December 2008.

Python is a very flexible language; it is simple to learn yet is fast enough to be used in production. Over the past ten years, more and more comprehensive data science toolkits (e.g. scikit-learn, NLTK, tensorflow, keras) have been written in Python and are now the standard frameworks for those models.

Just like R, Python is an open-source software. It is free to use and extend.

## 2.2 Installing Python and Jupyter Lab

To install Python, navigate to the official Python download page to obtain the appropriate installer for your operating system.

> **!** Important
>
> For our class, please ensure that you are using at least Python 3.12, and that you have installed the package versions in `requirements.txt`.

The next step is to create a virtual environment for this course. Virtual environments are specific to Python. They allow you to retain multiple versions of Python, and of packages, on the same computer. Go through the videos on Canvas relevant to your operating system to create a virtual environment and install Jupyter Lab on your machine.

Jupyter notebooks are great for interactive work with Python, but more advanced users may prefer a full-fledged IDE. If you are an advanced user, and are comfortable with an IDE of your own choice (e.g. Spyder or VSCode), feel free to continue using that to run the codes for this course.

> **Important**
>
> Even if you are using Anaconda/Spyder/VSCode, you still need to create a virtual environment, and you still need to install packages using the `requirements.txt` file.

Jupyter notebooks consist of cells, which can be of three main types:

- code cells,
- output cells, and
- markdown cells.

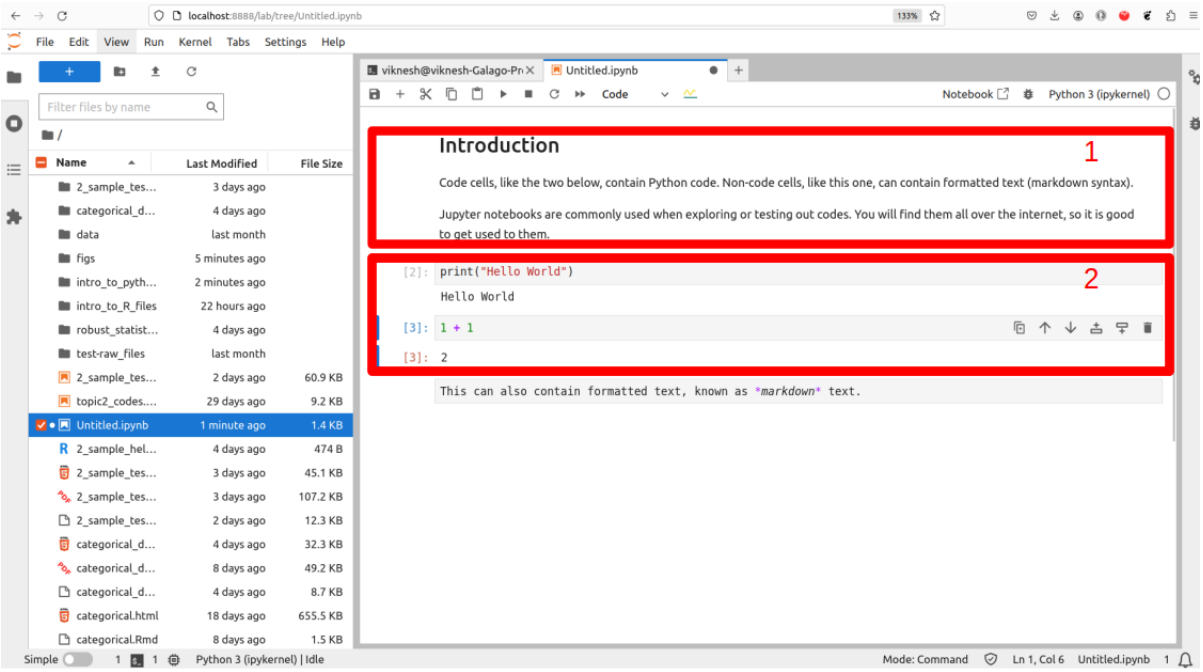



Figure 2.1: Py: JupyterLab interface

In Figure 2.1, the red box labelled 1 is a markdown cell. It can be used to contain descriptions or summary of the code. The cells in the box labelled 2 are code cells. To run the codes from our notes, you can copy and paste the codes into a new cell, and then execute them with Ctrl-Enter.

Try out this Easter egg that comes with any Python installation:

```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
```

```
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

More information on using Jupyter notebooks can be obtained from this link.

## 2.3 Basic Data Structures in Python

The main objects in native[1] Python that contain data are

- **Lists**, which are defined with [ ]. Lists are mutable.
- **Tuples**, which are defined with ( ). Tuples are immutable.
- **Dictionaries**, which are defined with { }. Dictionaries have keys and items. They are also mutable.

Very soon, we shall see that for statistics, the more common objects we shall deal with are dataframes (from pandas) and arrays (from numpy). However, the latter two require add-on packages; the three object classes listed above are baked into Python.

Here is how we create lists, tuples and dictionaries. Elements in lists and tuples can be accessed using a square bracket notation, just like in R.

```
x_list = [1, 2, 3]
x_tuple = (1, 2, 3)
x_dict = {'a': 1, 'b': 2, 'c': 3} # access with x_dict['a']
```

Suppose we have the following list:

```
x = [1, 3, 5, 7, 8, 9, 10]
```

The following assignment is OK, because `x` is a list, and hence mutable.

```
x[3] = 17
print(x)
```

[1] i.e., Python without any packages imported.

```
[1, 3, 5, 17, 8, 9, 10]
```

However, the following will return an error, because `x_tuple` is a tuple, and hence immutable.

```
x_tuple = (1, 3, 5, 6, 8, 9, 10)
x_tuple[3] = 17
```

> **Note**
>
> Note that we do *not* need the `c( )` function, like we did in R. This is a common mistake when switching between the two languages.

## 2.4 Slice Operator in Python

One important point to take note is that, in contrast to R, Python indexes objects starting with 0. Second, the slice operator in Python is a little more powerful than in R. It can be used to extract regular sequences from a list, tuple or string easily.

In general, the syntax is `<list-like object>[a:b]`, where `a` and `b` are integers. Such a call would return the elements at indices `a`, `a+1` until `b-1`. Take note that the end point index is not included.

```
char_list = ['P', 'y', 't', 'h', 'o', 'n']
char_list[0]            # returns first element
```

```
'P'
```

```
char_list[-1]           # returns last element
```

```
'n'
```

```
len(char_list)          # returns number of elements in list-like object.
```

```
6
```

```
char_list[::2]          # from first to last, every 2 apart.
```

```
['P', 't', 'o']
```

This indexing syntax is used in the additional packages we are going to use as well, so it is good to know about it. Take note that in Python, a negative index does not correspond to the element being dropped; it corresponds to counting from the end of the array-like object. Figure 2.2 displays a pictorial representation of how positive and negative indexes work together.

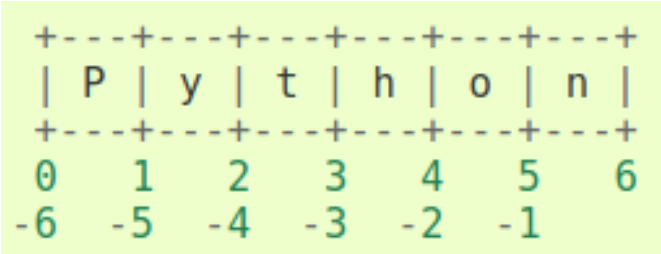



Figure 2.2: Py: Positive and negative indexing

## 2.5 Numpy Arrays

Just like R, Python has several contributed packages that are essential for statistics and data analysis. These include **numpy** and **pandas**. These correct versions of these packages would have been installed if you had used the provided requirements file.

```
import numpy as np

array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([6, 7, 8, 9, 10])
matrix1 = np.array([array1, array2])
print(matrix1)
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
```

The slice operator can then be used in each dimension of the matrix to subset it.

```
matrix1[0, 0::3]        # returns first row, columns 1 and 4
```

```
array([1, 4])
```

```
matrix1[1, 1:3]         # returns second row, columns 2 and 3
```

```
array([7, 8])
```

The numpy arrays are objects in Python, with several attributes and methods associated with them. Here are a couple of attributes:

```
# To obtain the dimensions of an array:
matrix1.shape
```

```
(2, 5)
```

```
# To transpose a 2-D array
matrix1.T
```

```
array([[ 1,  6],
       [ 2,  7],
       [ 3,  8],
       [ 4,  9],
       [ 5, 10]])
```

Here is a table with some common methods associated with numpy arrays. The objects referred to in the second column are from the earlier lines of code.

| Method | Description |
|---|---|
| `mean` | Computes col- or row-wise means, e.g. `matrix1.mean(axis=0)` or `matrix1.mean(axis=1)` |
| `sum` | Computes col- or row-wise means, e.g. `matrix1.sum(axis=0)` or `matrix1.sum(axis=1)` |
| `argmax` | Return the index corresponding to the max within the specified dimension, e.g. `matrix1.argmax(axis=0)` for the position with the max within each column. |
| `reshape` | To change the dimensions, e.g. `array1.reshape((5,1))` converts the array into a 5x1 matrix |

To combine arrays, we use the functions `vstack` and `hstack`. These are analogous to `rbind` and `cbind` in R.

```
np.vstack([matrix1, array1])
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [ 1,  2,  3,  4,  5]])
```

```
np.hstack([array1.reshape((5,1)),
           array2.reshape((5,1)),
           matrix1.T])
```

```
array([[ 1,  6,  1,  6],
       [ 2,  7,  2,  7],
       [ 3,  8,  3,  8],
       [ 4,  9,  4,  9],
       [ 5, 10,  5, 10]])
```

## 2.6 Pandas DataFrames

The next important add-on package that we shall work with is `pandas`. It provides a `DataFrame` class of objects for working with tabular data, just like `data.frame` within R. However, there are some syntactic differences with R that we shall soon get to. The following command creates a simple pandas dataframe.

```
import pandas as pd

data = {'X': [1,2,3,4,5,6], 'Y': [6,5,4,3,2,1]}
df = pd.DataFrame(data, columns =['X', 'Y'])
print(df)
```

```
   X  Y
0  1  6
1  2  5
2  3  4
3  4  3
4  5  2
5  6  1
```

We will get into the syntax for accessing subsets of the dataframe soon, but for now, here is how we can extract a single column from the dataframe. The resulting object is a pandas Series, which is a lot like a 1-D array, and can be indexed like one as well.

```
col_x = df.X
col_x[0:3]
```

```
0    1
1    2
2    3
Name: X, dtype: int64
```

> **Note**
>
> The built-in objects in Python are lists, tuples and dictionaries. Lists and tuples can contain elements of different types, e.g. strings and integers in a single object. However, they have no dimensions, so to speak of. Numpy arrays can be high-dimensional structures. In addition, the elements have to be homogeneous. For instance, in a 2x2x2 numeric numpy array, every one of the 8 elements has to be a float point number. Pandas DataFrames are tabular objects, where, within a column, each element has to be of the same type.

## 2.7 Reading Data into Python

Let us begin with the same file that we began with in the topic on R: `crab.txt`. In Section 1.5, we observed that this file contained headings, and that the columns were separated by spaces. The `pandas` function to read in such text files is `read_table()`. It has numerous optional arguments, but in this case we just need these two:

```
data1 = pd.read_table('data/crab.txt', header=0, sep="\\s+")
data1.head()
```

```
   color  spine  width  satell  weight
0      3      3   28.3       8    3.05
1      4      3   22.5       0    1.55
2      2      1   26.0       9    2.30
3      4      3   24.8       0    2.10
4      4      3   26.0       4    2.60
```

Do take note of the differences with R - the input to the header argument corresponds to the line number containing the column names. Secondly, the `head()` function is a method belonging to the DataFrame object.

When the file does not contain column names, we can supply them (as a list or numpy array) when we read the data in. Here is an example:

```
varnames = ["Subject", "Gender", "CA1", "CA2", "HW"]
data2 = pd.read_table('data/ex_1.txt', header=None,
                      names=varnames, sep="\\s+")
data2
```

```
   Subject Gender  CA1  CA2 HW
0       10      M   80   84  A
1        7      M   85   89  A
2        4      F   90   86  B
3       20      M   82   85  B
4       25      F   94   94  A
5       14      F   88   84  C
```

## 2.8 Subsetting DataFrames with Pandas

DataFrames in pandas are indexed for efficient searching and retrieval. When subsetting them, we have to add either `.loc` or `.iloc` and use it with square brackets.

The `.loc` notation is used when we wish to index rows and columns according to their names. The general syntax is `<DataFrame>.loc[ , ]`. A slice operator can be used separately for the row subset and the column subset to be retrieved.

For instance, to retrieve rows 0,1,2 and columns from color to width:

```
data1.loc[0:2, 'color':'width']
```

```
   color  spine  width
0      3      3   28.3
1      4      3   22.5
2      2      1   26.0
```

To retrieve every second row starting from row 0 until row 5, and all columns:

```
data1.loc[0:5:2, ]
```

```
   color  spine  width  satell  weight
0      3      3   28.3       8    3.05
2      2      1   26.0       9    2.30
4      4      3   26.0       4    2.60
```

The `.iloc` notation is used when we wish to index rows and columns using integer values. The general syntax is similar; try this and observe the difference with `.loc`.

```
data1.iloc[0:2, 0:2]
```

```
   color  spine
0      3      3
1      4      3
```

If you notice, the `.iloc` notation respects the rules of the in-built slice operator, in the sense that the end point is *not* included in the output. On the other hand, the `.loc` notation includes the end point.

In data analysis, a common requirement is to subset a dataframe according to values in columns. Just like in R, this is achieved with logical values.

```
data2[data2.Gender == "M"]
```

```
   Subject Gender  CA1  CA2 HW
0       10      M   80   84  A
1        7      M   85   89  A
3       20      M   82   85  B
```

```
data2[(data2.Gender == "M") & (data2.CA2 > 85)]
```

```
   Subject Gender  CA1  CA2 HW
1        7      M   85   89  A
```

## 2.9 Loops in Python

It is extremely efficient to execute "for" loops in Python. Many objects in Python are *iterators*, which means they can be iterated over. Lists, tuples and dictionaries can all be iterated over very easily.

Before getting down to examples, take note that Python does not use curly braces to denote code blocks. Instead, these are defined by the number of indentations in a line.

```
for i in x[:2]:
  print(f"The current element is {i}.")
```

```
The current element is 1.
The current element is 3.
```

Notice how we do not need to set up any running index; the object is just iterated over directly. The argument to the `print()` function is an f-string. It is the recommended way to create string literals that can vary according to arguments.

Here is another example of iteration, this time using dictionaries which have key-value pairs. In this case, we iterate over the keys.

```
dict1 = {'holmes': 'male', 'watson': 'male', 'mycroft': 'male',
         'hudson': 'female', 'moriarty': 'male', 'adler': 'female'}
# dict1['hudson']

for x in dict1.keys():
    print(f"The gender of {x} is {dict1[x]}")
```

```
The gender of holmes is male
The gender of watson is male
The gender of mycroft is male
The gender of hudson is female
The gender of moriarty is male
The gender of adler is female
```

In Section 1.7, we wrote a block of code that incremented an integer until the square was greater than 36. Here is the Python version of that code:

```
x = 0
test = True

while test:
    x += 1
    test = x < 6
    print(f"{x**2:2d}, {test}")
```

```
 1, True
 4, True
 9, True
16, True
25, True
36, False
```

It is also straightforward to write a for-loop to perform the above, since we know when the break-point of the loop will be. The `np.arange()` function generates evenly spaced integers.

```
for y in np.arange(1, 7):
    print(f"The square of {y} is {y**2:2d}.")
```

```
The square of 1 is  1.
The square of 2 is  4.
The square of 3 is  9.
The square of 4 is 16.
The square of 5 is 25.
The square of 6 is 36.
```

## 2.10 User Defined Functions

The syntax for creating a new function in Python is as follows:

```
def fn_name(arguments):
  Python statements
  ...
  Returned object
```

Here is the Python version of the function from earlier, computing the circumference of a circle with a given radius.

```
import math

def circumference(radius):
    return 2 * math.pi * radius

circumference(1.3)
```

```
8.168140899333462
```

## 2.11 Miscellaneous

### 2.11.1 Package installation

So far, we have used numpy and pandas, but we shall need to call upon a few other add-on packages we proceed in the course. These include statsmodels, scipy and matplotlib.

### 2.11.2 Getting help

Most functions in Python are well-documented. In order to access this documentation from within a Jupyter notebook, use the `?` operator. For more details, including the source code, use the `??` operator. For instance, for more details of the `pd.read_csv()` function, you can execute this command:

```
pd.read_csv?
```

The internet is full of examples and how-to's for Python; help is typically just a Google search or a chatGPT query away. However, it is always better to learn from the ground up instead of through snippets for specific tasks. Please look through the websites in Section 2.13.1 below.

## 2.12 Major Differences with R

Before we leave this topic, take note of some very obvious differences with R:

1. The assignment operator in R is `<-`; for Python it is `=`.
2. When creating vectors in R, you will need `c( )`, but in Python, this is not the case.
3. R implements it's object oriented mechanism in a different manner from Python. For instance, when plotting with R, you would call `plot(<object>)` but in Python, you would call `<object>.plot()`. In Python, the methods belong to the class, but not in R.

## 2.13 References

### 2.13.1 Website References

1. Beginner's guide to Numpy: This is from the official numpy documentation website.
2. 10 minutes to Pandas: This is a quickstart to pandas, from the official website. You can find more tutorials on this page too.
3. Python official documentation: This is from the official Python page. It contains a tutorial, an overview of all built-in packages, and several howto's, including on regular expression. A very good website to learn from.
4. Python download: The official download page for Python.
5. Jupyter Lab help: The documentation site for Jupyter Lab.

## 2.14 Exercises

1. Write a Python function to return the area of a circle. It should take the radius as the input argument.
2. Refer to `data1` from Section 2.7. Use logical indexing to retrieve all rows corresponding to `color` equals 2, or `spine` equals 1.
3. Consider once more, the dataframe `data1` from Section 2.7. Complete the following code to result in the output that follows. The output corresponds to rows where `satell` is greater than 10.

```
for z in data1.iterrows():
    if _____ > 10:
        print(_____)
```

```
Row 12: The weight is 3.05
Row 14: The weight is 2.3
Row 55: The weight is 3.0
Row 116: The weight is 3.225
```

4. Use the slice operator, along with `.iloc` or `.loc`, to return every second row from `data1`.
5. What does the following code do? How different is it from a conventional loop in Python?

```
olist = [dict1[x] for x in dict1.keys()]
```

# 3 Exploring Quantitative Data

## 3.1 Introduction

This is the first topic where we are going to see how to perform statistical analysis using two software: R and Python. Every language has its strengths and weaknesses, so instead of attempting to exactly replicate the same chart/output in each software, we shall try to understand their respective approaches better.

Our end goal is to analyse data. Toward that end, we should be versatile and adaptable. Focus on learning how to be fast and productive in whichever environment you are told to work in. If you have a choice, then be aware of what each framework can do best, so that you can choose the right one.

Although it is a simplistic view, Figure 3.1 provides a useful taxonomy of the types of columns we might have in our dataset. Having at least an initial inkling of the type of data matters, because it helps decide what type of summary to generate or what type of plot to make. Quantitative data are sometimes also referred to as *numerical data*.

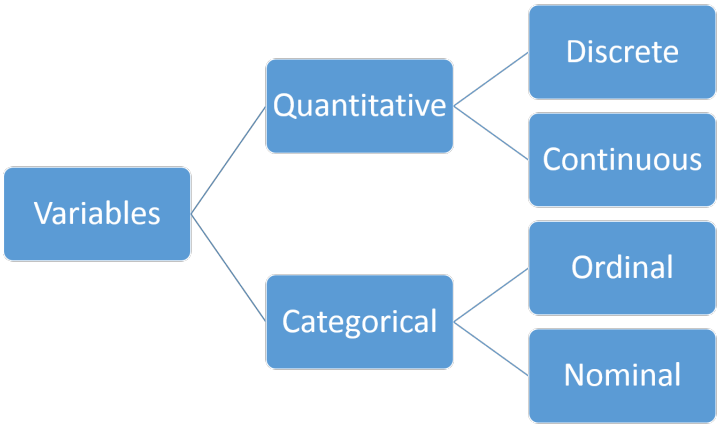



Figure 3.1: Data types

There are two main ways of summarising data: numerically and graphically. This topic will cover:

1. Numerical summaries for univariate quantitative variables.
2. Numerical summary for association between two quantitative variables.
3. Useful graphs for univariate and bivariate quantitative variables.

Techniques for categorical variables will be covered in Section 4.1.

Before proceeding, let us introduce one of the datasets that we'll be using in this textbook. The dataset comes from the UCI Machine Learning Repository, which is a very useful place to get datasets for practice.

**Example 3.1** (Student Performance: Dataset Introduction)**.** The full dataset can be downloaded from this page. Once you unzip the file, you will find two `csv` files in the `student/` folder:

- `student-mat.csv` (performance in Mathematics)
- `student-por.csv` (performance in Portugese)

Each dataset was collected using school reports and questionnaires. Each row corresponds to a student. The columns include student grades, demographic information, and other social and school-related information. Each file corresponds to the students' performance in one of the two subjects. For more information, you can refer to Cortez and Silva (2008).

| # | Feature | Description (Type) | Details |
|---|---|---|---|
| 1 | school | student's school (binary) | "GP" - Gabriel Pereira, "MS" - Mousinho da Silveira |
| 2 | sex | student's sex (binary) | "F" - female, "M" - male |
| 3 | age | student's age (numeric) | from 15 to 22 |
| 4 | address | student's home address type (binary) | "U" - urban, "R" - rural |
| 5 | famsize | family size (binary) | "LE3" - less or equal to 3, "GT3" - greater than 3 |
| 6 | Pstatus | parent's cohabitation status (binary) | "T" - living together, "A" - apart |
| 7 | Medu | mother's education (numeric) | 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education, 4 - higher education |
| 8 | Fedu | father's education (numeric) | 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education, 4 - higher education |
| 9 | Mjob | mother's job (nominal) | teacher, health, civil services, at_home, other |
| 10 | Fjob | father's job (nominal) | teacher, health, civil services, at_home, other |
| 11 | reason | reason to choose this school (nominal) | close to home, school reputation, course preference, other |
| 12 | guardian | student's guardian (nominal) | mother, father, other |
| 13 | traveltime | home to school travel time (numeric) | 1 - <15 min, 2 - 15 to 30 min, 3 - 30 min to 1 hour, 4 - >1 hour |
| 14 | studytime | weekly study time (numeric) | 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, 4 - >10 hours |
| 15 | failures | number of past class failures (numeric) | n if 1<=n<3, else 4 |
| 16 | schoolsup | extra educational support (binary) | yes or no |
| 17 | famsup | family educational support (binary) | yes or no |
| 18 | paid | extra paid classes within the course subject (Math or Portuguese) (binary) | yes or no |
| 19 | activities | extra-curricular activities (binary) | yes or no |
| 20 | nursery | attended nursery school (binary) | yes or no |

| # | Feature | Description (Type) | Details |
|---|---|---|---|
| 21 | higher | wants to take higher education (binary) | yes or no |
| 22 | internet | Internet access at home (binary) | yes or no |
| 23 | romantic | with a romantic relationship (binary) | yes or no |
| 24 | famrel | quality of family relationships (numeric) | from 1 - very bad to 5 - excellent |
| 25 | freetime | free time after school (numeric) | from 1 - very low to 5 - very high |
| 26 | goout | going out with friends (numeric) | from 1 - very low to 5 - very high |
| 27 | Dalc | workday alcohol consumption (numeric) | from 1 - very low to 5 - very high |
| 28 | Walc | weekend alcohol consumption (numeric) | from 1 - very low to 5 - very high |
| 29 | health | current health status (numeric) | from 1 - very bad to 5 - very good |
| 30 | absences | number of school absences (numeric) | from 0 to 93 |

In the explanatory variables above, notice that although several variables have been stored in the dataset file as numbers, they are in fact *categorical variables*. Examples of these are variables 24 to 29. However, it does seem fair to treat the three output columns as numeric. G3 is the main output variable.

| # | Feature | Description (Type) | Details |
|---|---|---|---|
| 31 | G1 | first period grade (numeric) | from 0 to 20 |
| 32 | G2 | second period grade (numeric) | from 0 to 20 |
| 33 | G3 | final grade (numeric, output target) | from 0 to 20 |

## 3.2 Numerical Summaries

Numerical summaries include:

1. Basic information about the data, e.g. number of observations and missing values.
2. Measures of central tendency, e.g. mean, median
3. Measures of spread, e.g. standard deviation, IQR (interquartile range), range.

**Example 3.2** (Student Performance: Numerical Summaries)**.** Let us read in the dataset and generate numerical summaries of the output variable of interest (G3).

#### R code

```r
stud_perf <- read.table("data/student/student-mat.csv", sep=";",
                        header=TRUE)
summary(stud_perf$G3)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00    8.00   11.00   10.42   14.00   20.00 
```

```r
sum(is.na(stud_perf$G3))
```

```
[1] 0
```

**Python code**

```python
import pandas as pd
import numpy as np

stud_perf  = pd.read_csv("data/student/student-mat.csv", delimiter=";")
stud_perf.G3.describe()
```

```
count    395.000000
mean      10.415190
std        4.581443
min        0.000000
25%        8.000000
50%       11.000000
75%       14.000000
max       20.000000
Name: G3, dtype: float64
```

```python
#stud_perf.G3.info()
```

From the output, we can understand that we have 395 observations, ranging from 0 to 20. I would surmise that the data is more or less symmetric in the middle (distance from 3rd-quartile to median is identical to distance from median to 1st-quartile). There are no missing values in the data.

However, summaries of a single variable are rarely useful since we do not have a basis for comparison. In this dataset, we are interested in how the grade varies with one or some of the *other* variables. Let's begin with Mother's education.

**R code**

```r
round(aggregate(G3 ~ Medu, data=stud_perf, FUN=summary), 2)
```

```
  Medu G3.Min. G3.1st Qu. G3.Median G3.Mean G3.3rd Qu. G3.Max.
1    0    9.00      12.00     15.00   13.00      15.00   15.00
2    1    0.00       7.50     10.00    8.68      11.00   16.00
3    2    0.00       8.00     11.00    9.73      13.00   19.00
4    3    0.00       8.00     10.00   10.30      13.00   19.00
5    4    0.00       9.50     12.00   11.76      15.00   20.00
```

```r
table(stud_perf$Medu)
```

```
  0   1   2   3   4
  3  59 103  99 131
```

**Python code**

```python
stud_perf[['Medu', 'G3']].groupby('Medu').describe()
```

```
         G3
      count       mean       std  min   25%   50%   75%   max
Medu
0       3.0  13.000000  3.464102  9.0  12.0  15.0  15.0  15.0
1      59.0   8.677966  4.364594  0.0   7.5  10.0  11.0  16.0
2     103.0   9.728155  4.636163  0.0   8.0  11.0  13.0  19.0
3      99.0  10.303030  4.623486  0.0   8.0  10.0  13.0  19.0
4     131.0  11.763359  4.267646  0.0   9.5  12.0  15.0  20.0
```

Now we begin to understand the context of G3 a little better. As the education level of the mother increases, the mean does increase. The middle 50-percent of the grade does seem to increase as well. The exception is the case where the mother has no education, but we can see that there are only 3 observations in that category, so we should not read too much into it.

Here are some things to note about numerical summaries:

- If the mean and the median are close to each other, it indicates that the distribution of the data is close to symmetric.
- The mean is sensitive outliers but the median is not. We shall see more about this in Section 5.1.
- When the mean is much larger than the median, it suggests that there could be a few very large observations. It has resulted in a *right-skewed* distribution. Conversely, if the mean is much smaller than the median, we probably have a *left-skewed* distribution.

While numerical summaries provide us with some basic information about the data, they also leave out a lot. Even for experts, it is possible to have a wrong mental idea about the data from the numerical summaries alone. For instance, all three histograms in Figure 3.2 have a mean of 0 and standard deviation of 1!

That's why we have to turn to *graphics* as well, in order to summarise our data.

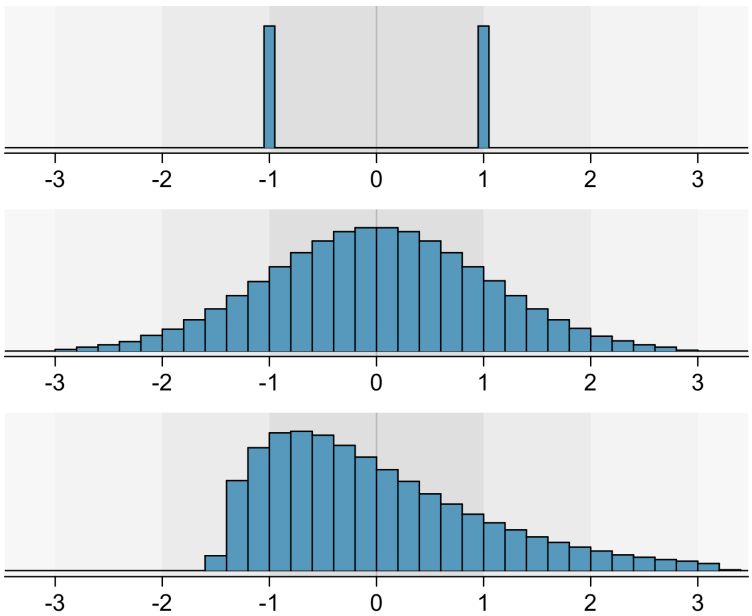



Figure 3.2: Three vastly different datasets, all with mean 0 and s.d. 1

## 3.3 Graphical Summaries

### 3.3.1 Histograms

The first chart type we shall consider is a histogram. A histogram is a graph that uses bars to portray the frequencies or relative frequencies of the possible outcomes for a quantitative variable.

When we create a histogram, here are some things that we look for:

1. *What is the overall pattern?*: Do the data cluster together, or is there a gap such that one or more observations deviate from the rest?
2. *Are there any suspected outliers?* See Figure 3.3.
3. *Do the data have a single mound or peak?* If yes, then we have what is known as a unimodal distribution. Data with two 'peaks' are referred to as bimodal, and data with many peaks are referred to as multimodal. See

    1.

4. *Is the distribution symmetric or skewed?* See Figure 3.5.

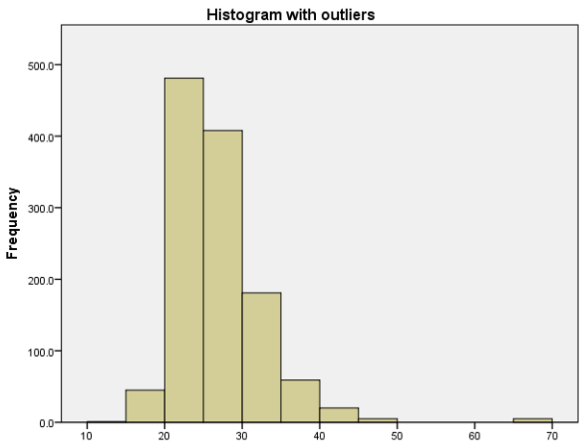



Figure 3.3: Histogram with outliers

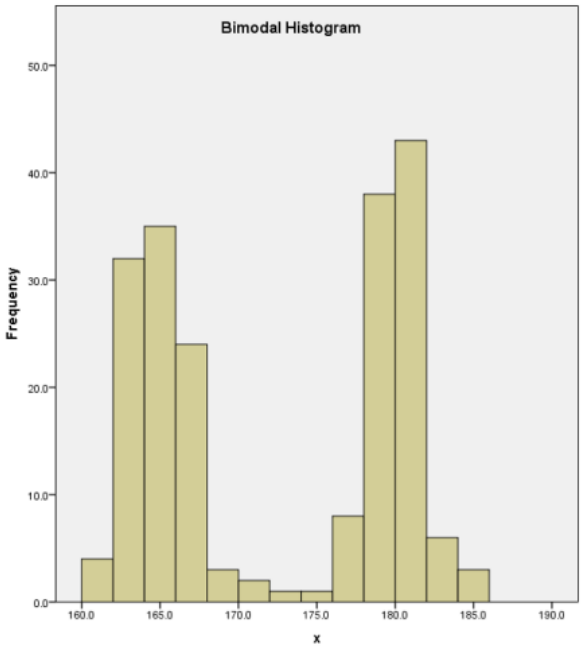



Figure 3.4: Bimodal histogram

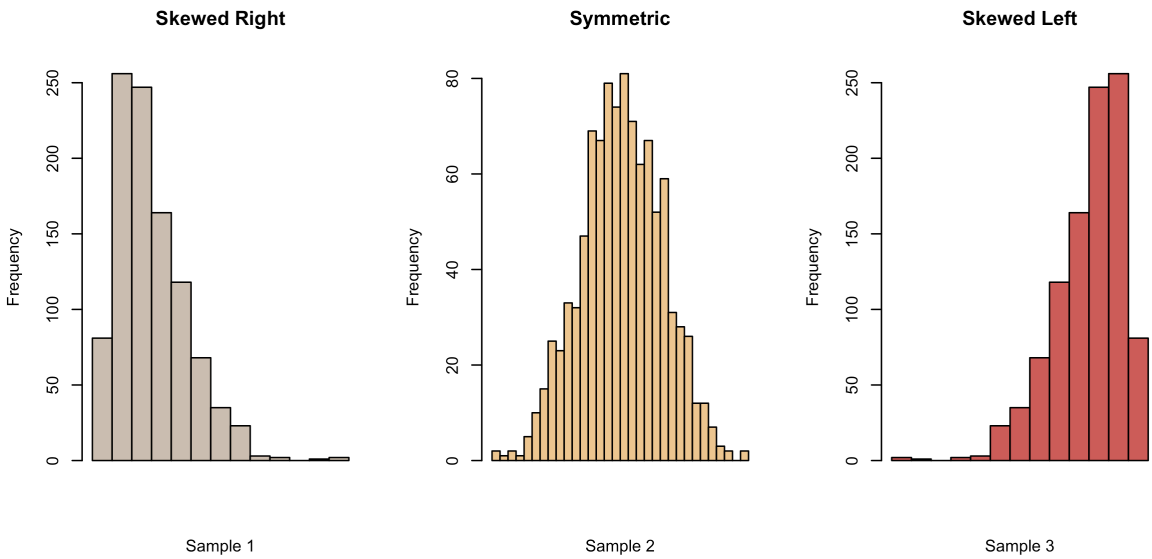



Figure 3.5: Skewed histograms

**Example 3.3** (Student Performance: Histograms)**.** Now let us return to the student performance dataset.

### R code

```
hist(stud_perf$G3, main="G3 histogram", xlab="G3 scores")
```

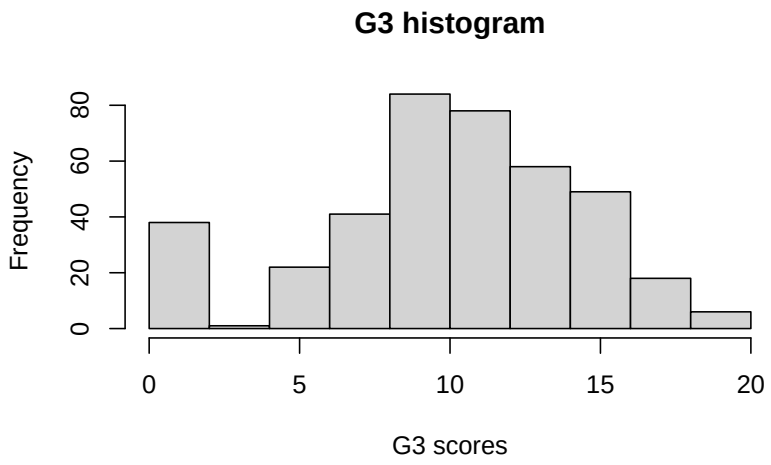



Figure 3.6: R: Histogram of G3 scores

### Python code

```
fig = stud_perf.G3.hist(grid=False)
fig.set_title('G3 histogram')
fig.set_xlabel('G3 scores');
```

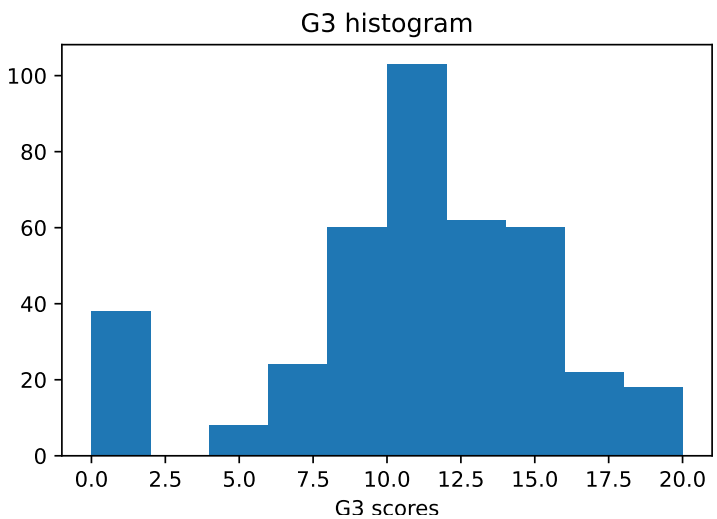



Figure 3.7: Py: Histogram of G3 scores

> **Note**
>
> Do you notice any differences between the two histograms?

In general, we now have a little more information to accompany the 5 number summaries. It appears that the distribution is not a basic unimodal one. There is a large spike of about 40 students who scored very low scores.

As we mentioned earlier, it is not useful to inspect a histogram in a silo. Hence we shall condition on Mother's education once more, to create separate histograms for each group. Remember that this is a case where the response variable `G3` is quantitative, and the explanatory variable `Medu` is ordinal.

### R code

```
library(lattice)
histogram(~G3 | Medu, data=stud_perf, type="density",
          main="G3 scores, by Medu levels", as.table=TRUE)
```

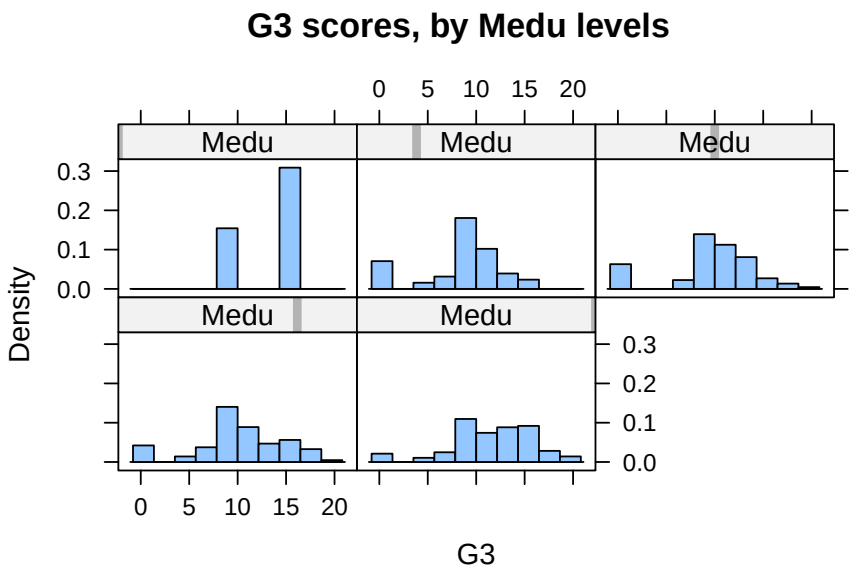



Figure 3.8: R: Lattice plot of G3 scores by Medu

### Python code

```
stud_perf.G3.hist(by=stud_perf.Medu, figsize=(15,10), density=True,
                  layout=(2,3));
```

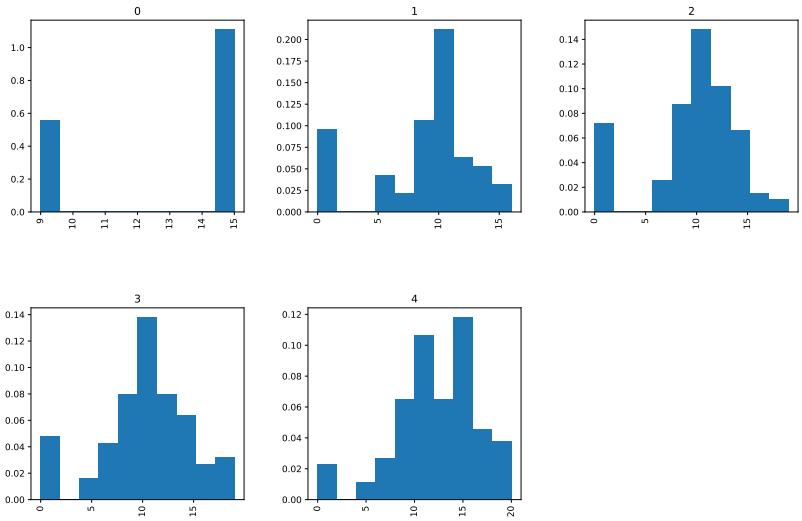



Figure 3.9: Py: Lattice plot of G3 scores by Medu

Although the heights of the panels in the two versions are not identical, we can, by looking at either one, see that the proportion of 0-scores reduces as the mother's education increases. Perhaps this is reading too much into the dataset, but there seem to be more scores on the higher end for highest educated mothers.

### 3.3.2 Density Plots

Histograms are not perfect - when using them, we have to experiment with the bin size since this could mask details about the data. It is also easy to get distracted by the blockiness of histograms. An alternative to histograms is the kernel density plot. Essentially, this is obtained by smoothing the heights of the rectangles in a histogram.

Suppose we have observed an i.i.d sample $x_1, x_2, \dots, x_n$ from a continuous pdf $f(\cdot)$. Then the kernel density estimate at $x$ is given by

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where

- $K$ is a density function. A typical choice is the standard normal. The kernel places greater weights on nearby points (to $x$).
- $h$ is a bandwidth, which determines which of the nearest points are used. The effect is similar to the number of bins in a histogram.

**Example 3.4** (Student Performance: Density Estimates)**.** Here is how we can make density plots with R and Python for the G3 scores.

**R code**

```
densityplot(~G3, groups=Medu, data=stud_perf, auto.key = TRUE,
            main="G3 scores, by Medu", bw=1.5)
```

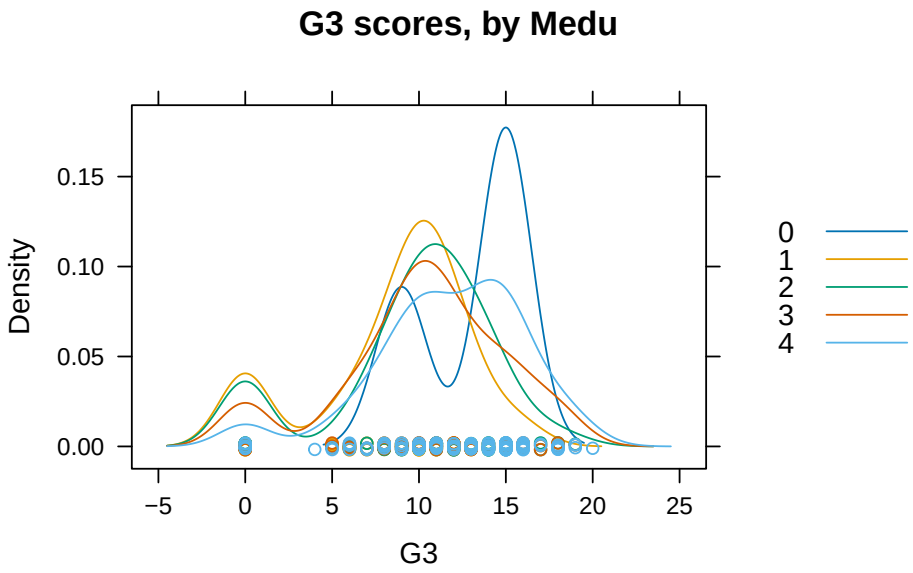



Figure 3.10: R: Estimated densities for G3 scores, by Medu

**Python code**

```
import matplotlib.pyplot as plt
f, axs = plt.subplots(2, 3, squeeze=False, figsize=(15,6))
out2 = stud_perf.groupby("Medu")
for y,df0 in enumerate(out2):
    tmp = plt.subplot(2, 3, y+1)
    df0[1].G3.plot(kind='kde')
    tmp.set_title(df0[0])
```

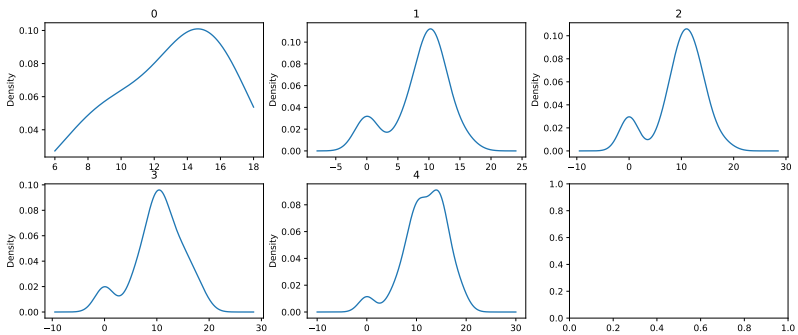



Figure 3.11: Py: Estimated densities for G3 scores, by Medu

As you can see, with density plots it is also possible to overlay them for closer comparison. This is not possible with histograms without some transparency in the rectangle colours.

### 3.3.3 Boxplots

A boxplot provides a skeletal representation of a distribution. Boxplots are very well suited for comparing multiple groups.

Here are the steps for drawing a boxplot:

1. Determine $Q_1$, $Q_2$ and $Q_3$. The box is made from $Q_1$ and $Q_3$. The median is drawn as a line or a dot within the box.
2. Determine the max-whisker reach: $Q_3 + 1.5 \times IQR$; the min-whisker reach by $Q_1 - 1.5 \times IQR$.
3. Any data point that is out of the range from the min to max whisker reach is classified as a *potential outlier*.
4. Excluding the potential outliers, the maximum point determines the *upper whisker* and the minimum point determines the *lower whisker* of a boxplot.

A boxplot helps us to identify the median, lower and upper quantiles and outlier(s) (see Figure 3.12).

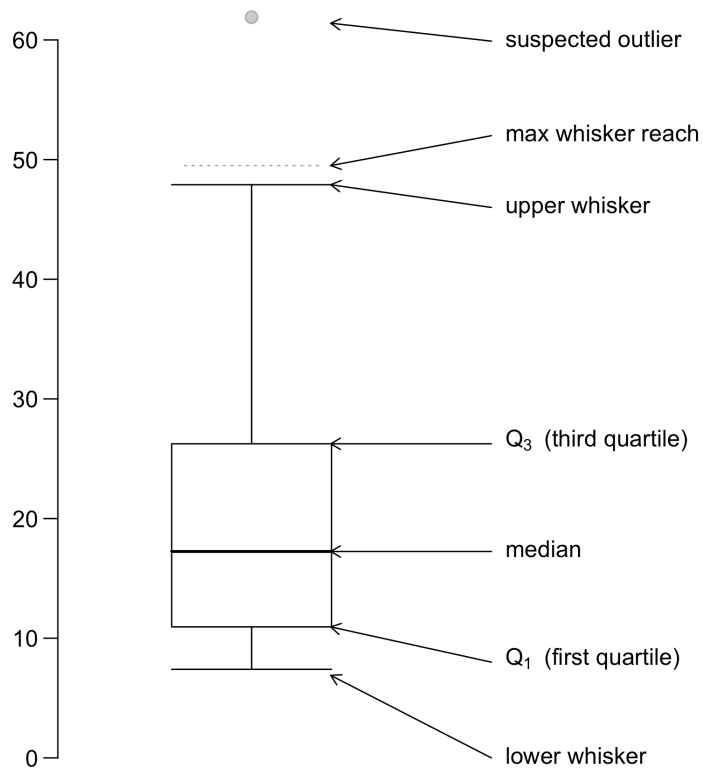



Figure 3.12: Boxplot construction

**Example 3.5** (Student Performance: Boxplots)**.** This time, instead of using mother's education, we use the number of times a student goes out (`goout`) as the explanatory variable. From

the boxplot outputs, it appears there is no strong strictly increasing/decreasing trend associated with G3. Instead, although the differences between the categories are not large, it seems as though there is an "optimal" number of times that students could go out. Too little and too much going out leads to lower median G3 scores.

### R code

```
bwplot(G3 ~ goout, horizontal = FALSE, main="G3 scores, by goout",
       xlab="No. of times the student goes out per week",
       data=stud_perf)
```

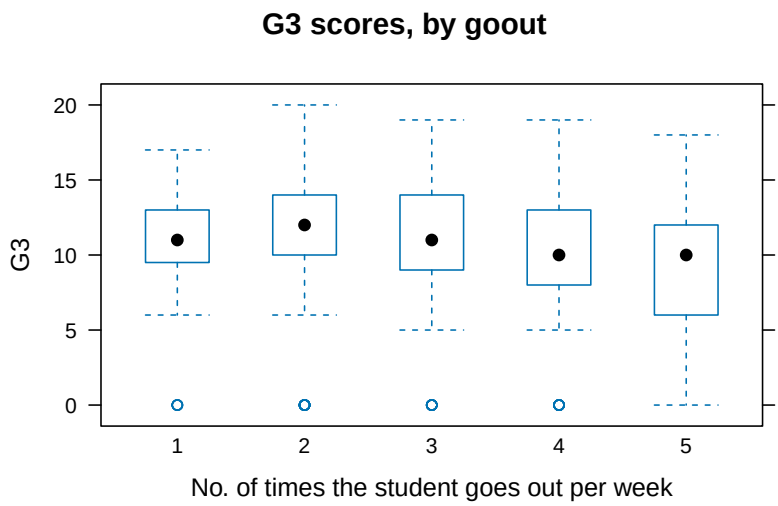



Figure 3.13: R: Boxplot of G3 scores, by gout variable

### Python code

```
stud_perf.plot.box(column='G3', by='goout',
                   xlabel='No. of times student goes out per week');
```

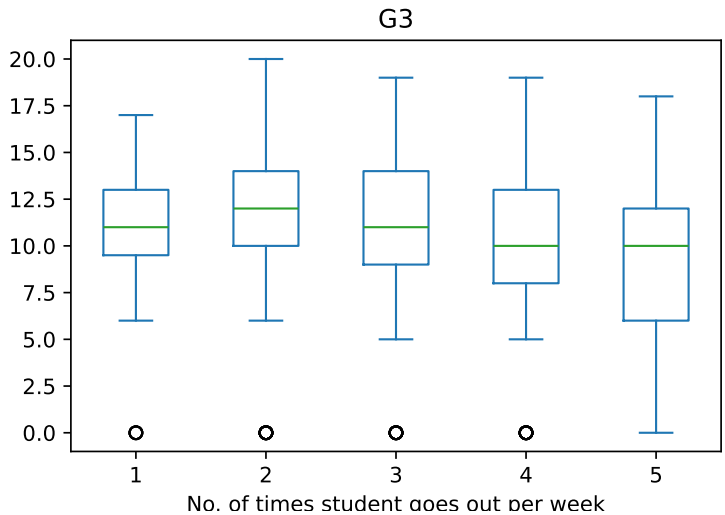


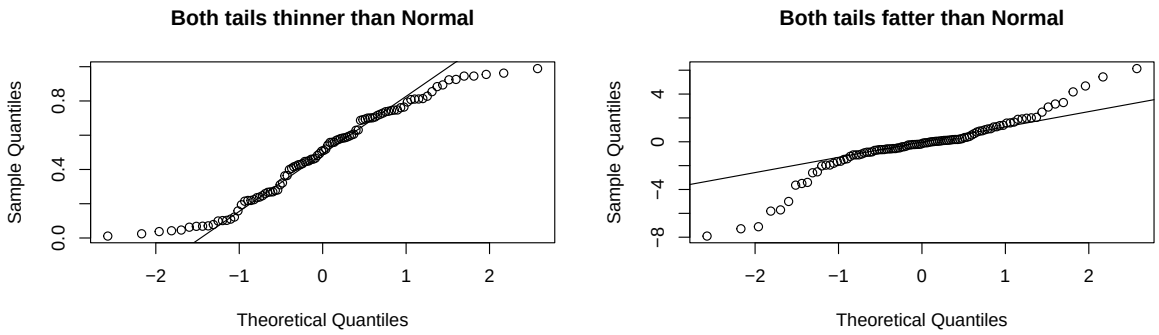

Figure 3.14: Py: Boxplot of G3 scores, by gout variable

### 3.3.4 QQ-plots

Finally, we turn to QQ-plots. A Quantile-Quantile plot is a graphical diagnostic tool for assessing if a dataset follows a particular distribution. Most of the time we would be interested in comparing against a Normal distribution.

A QQ-plot plots the standardized sample quantiles against the theoretical quantiles of a N(0; 1) distribution. If the points fall on a straight line, then we say there is evidence that the data comes from a Normal distribution.

Especially for unimodal datasets, the points in the middle will typically fall close to the line. The value of a QQ-plot is in judging if the tails of the data are fatter or thinner than the tails of the Normal.




(a) Thinner than Normal

(b) Fatter than Normal

Figure 3.15: QQ-plots

> **Note**
>
> Please be careful! Some software/packages will switch the axes (i.e. plot the sample quantiles on the x-axis instead of the y-axis, unlike Figure 3.15). Please observe and

interpret accordingly.

**Example 3.6** (Concrete Slump: Dataset Introduction)**.** Concrete is a highly complex material. The slump flow of concrete is not only determined by the water content, but also by other ingredients. The UCI page for this particular dataset is here. The reference article on this dataset is Yeh (2007).

The data set includes 103 data points. There are 7 input variables, and 3 output variables in the data set. These are the input columns in the data, all in units of $kg/m^3$ concrete:

| # | Feature | Details |
|---|---|---|
| 1 | Cement | |
| 2 | Slag | |
| 3 | Fly ash | |
| 4 | Water | |
| 5 | SP | A super plasticizer to improve consistency. |
| 6 | Coarse Aggr. | |
| 7 | Fine Aggr. | |

There are three output variables in the dataset. You can read more about Slump and Flow from this wikipedia page.

| # | Feature | Units |
|---|---|---|
| 1 | SLUMP | cm |
| 2 | FLOW | cm |
| 3 | 28-day Compressive Strength | MPa |

### R code

To read the data into R:

```
concrete <- read.csv("data/concrete+slump+test/slump_test.data")
names(concrete)[c(1,11)] <- c("id", "Comp.Strength")
```

### Python code

The corresponding Python code is:

```
concrete = pd.read_csv("data/concrete+slump+test/slump_test.data")
concrete.rename(columns={'No':'id',
                         'Compressive Strength (28-day)(Mpa)':'Comp_Strength'},
                inplace=True)
```

Let us consider the Comp.Strength output variable. The histogram overlay in Figure 3.16 suggests some skewness and fatter tails than the Normal.

**Example 3.7** (Concrete: QQ-plots)**.** The next chart is a QQ-plot, for assessing deviations from Normality.

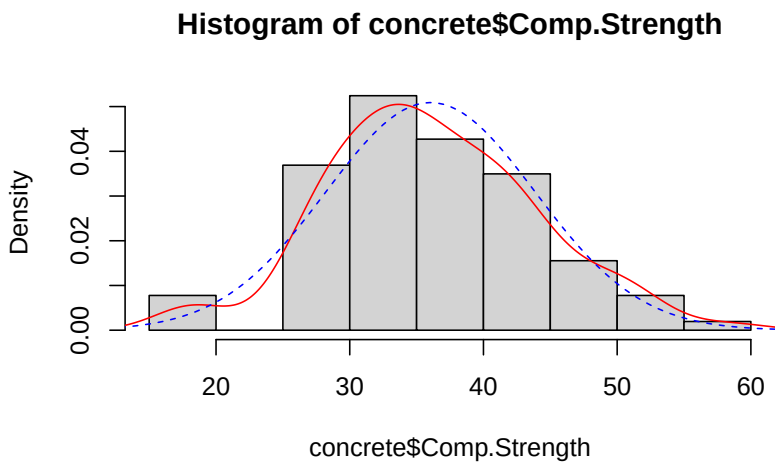



Figure 3.16: Comparing data with reference Normal (blue)

### R code

```
qqnorm(concrete$Comp.Strength)
qqline(concrete$Comp.Strength)
```

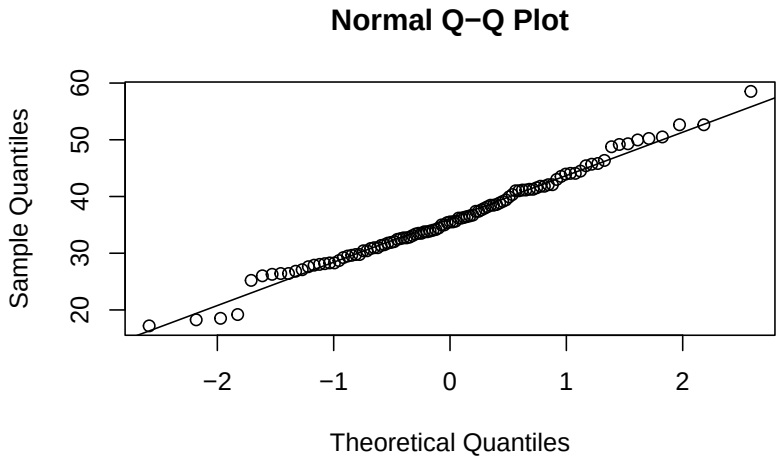



Figure 3.17: R: QQ plot for compressive strength

### Python code

```
from scipy import stats
import statsmodels.api as sm
sm.qqplot(concrete.Comp_Strength, line="q");
```

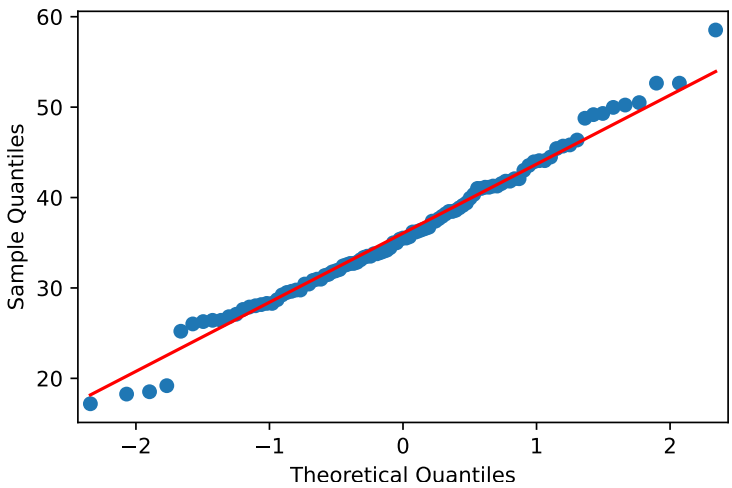



Figure 3.18: Py: QQ plot for compressive strength

The deviation of the tails does not seem to be that large, judging from the QQ-plot.

## 3.4 Correlation

When we are studying two quantitative variables, the most common numerical summary to quantify the relationship between them is the correlation coefficient. Suppose that $x_1, x_2, \dots, x_n$ and $y_1, \dots, y_n$ are two variables from a set of $n$ objects or people. The sample correlation between these two variables is computed as:

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \frac{(x_i - \bar{x})(y_i - \bar{y})}{s_x s_y}$$

where $s_x$ and $s_y$ are the sample standard deviations. $r$ is an estimate of the correlation between random variables $X$ and $Y$.

A few things to note about the value $r$, which is also referred to as the Pearson correlation:

- $r$ is always between -1 and 1.
- A positive value for $r$ indicates a positive association and a negative value for $r$ indicates a negative association.
- Two variables have the same correlation, no matter which one is coded as $X$ and which one is coded as $Y$.

Figure 3.19 and Figure 3.20 contain sample plots of data and their corresponding $r$ values. Notice how $r$ does not reflect strong non-linear relationships.

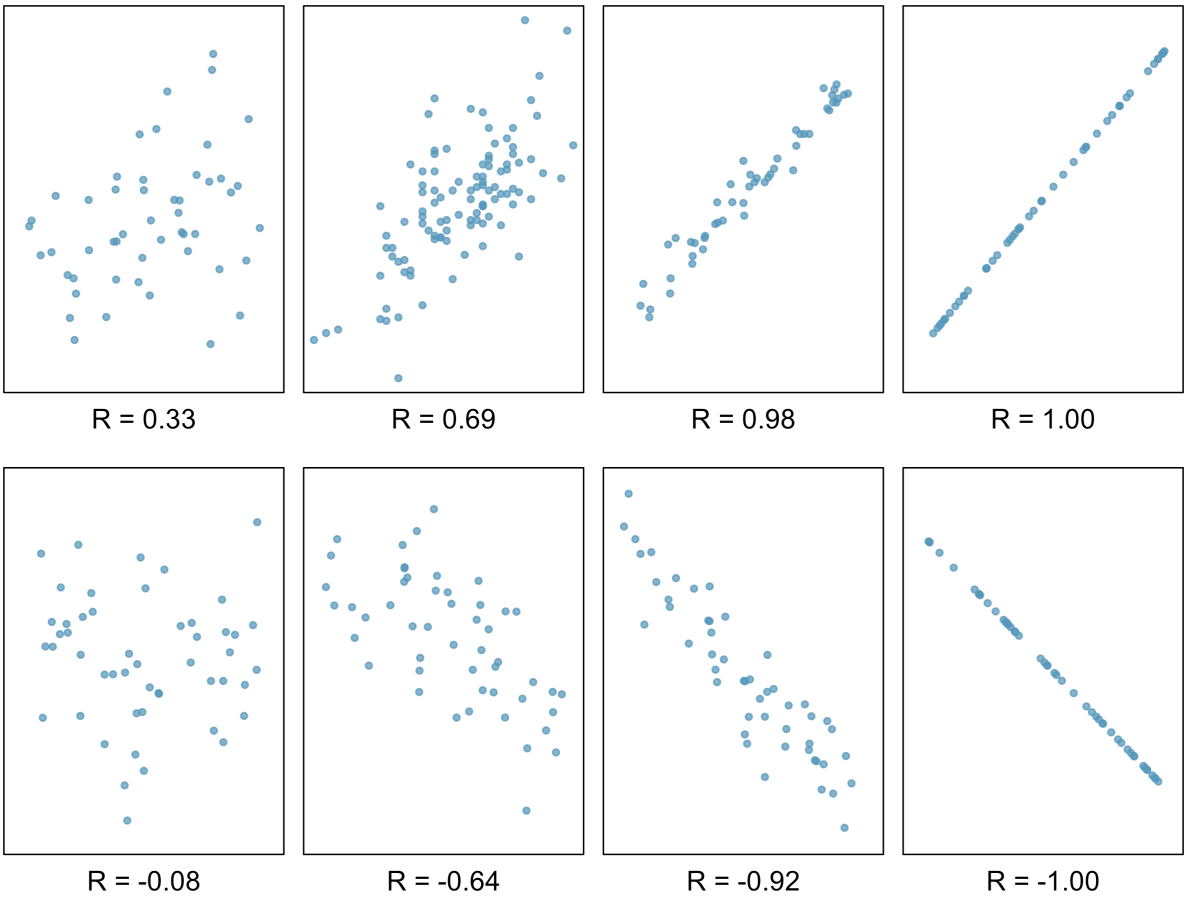



Figure 3.19: Varying linear relations and corresponding $r$-values

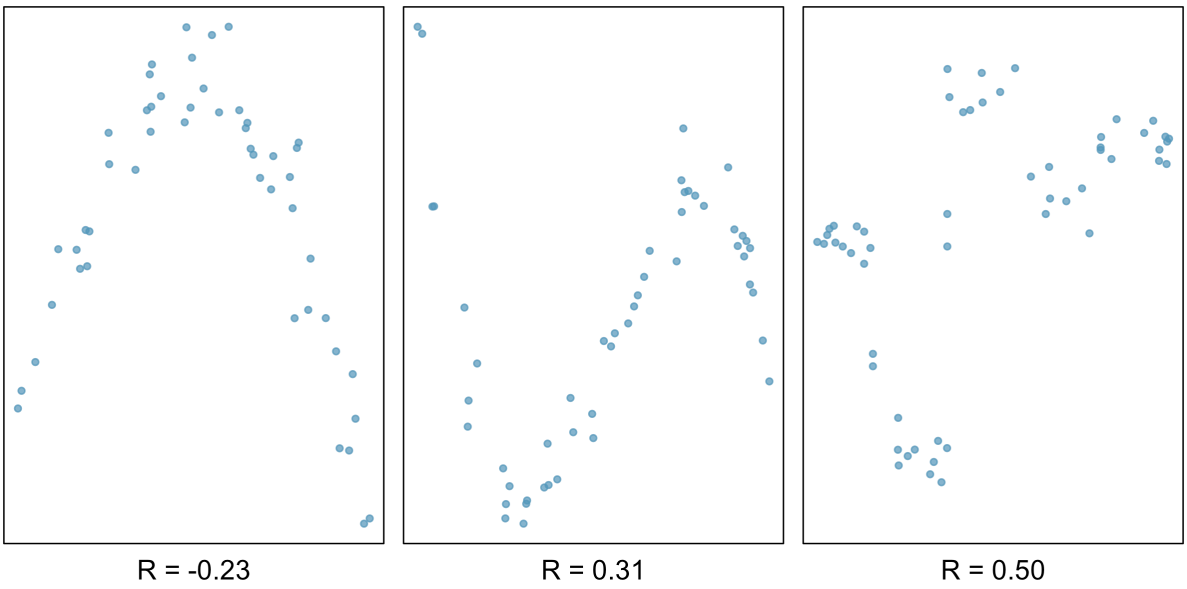



Figure 3.20: Non-linear relations and corresponding $r$-values

## 3.5 Scatterplot Matrices

**Example 3.8** (Concrete: Scatterplots)**.** When we have multiple quantitative variables in a dataset, it is common to create a matrix of scatterplots. This allows for simultaneous inspection of bivariate relationships.

### R code

```
col_to_use <- c("Cement", "Slag", "Comp.Strength", "Water", "SLUMP.cm.",
                "FLOW.cm.")
pairs(concrete[, col_to_use], panel = panel.smooth)
```

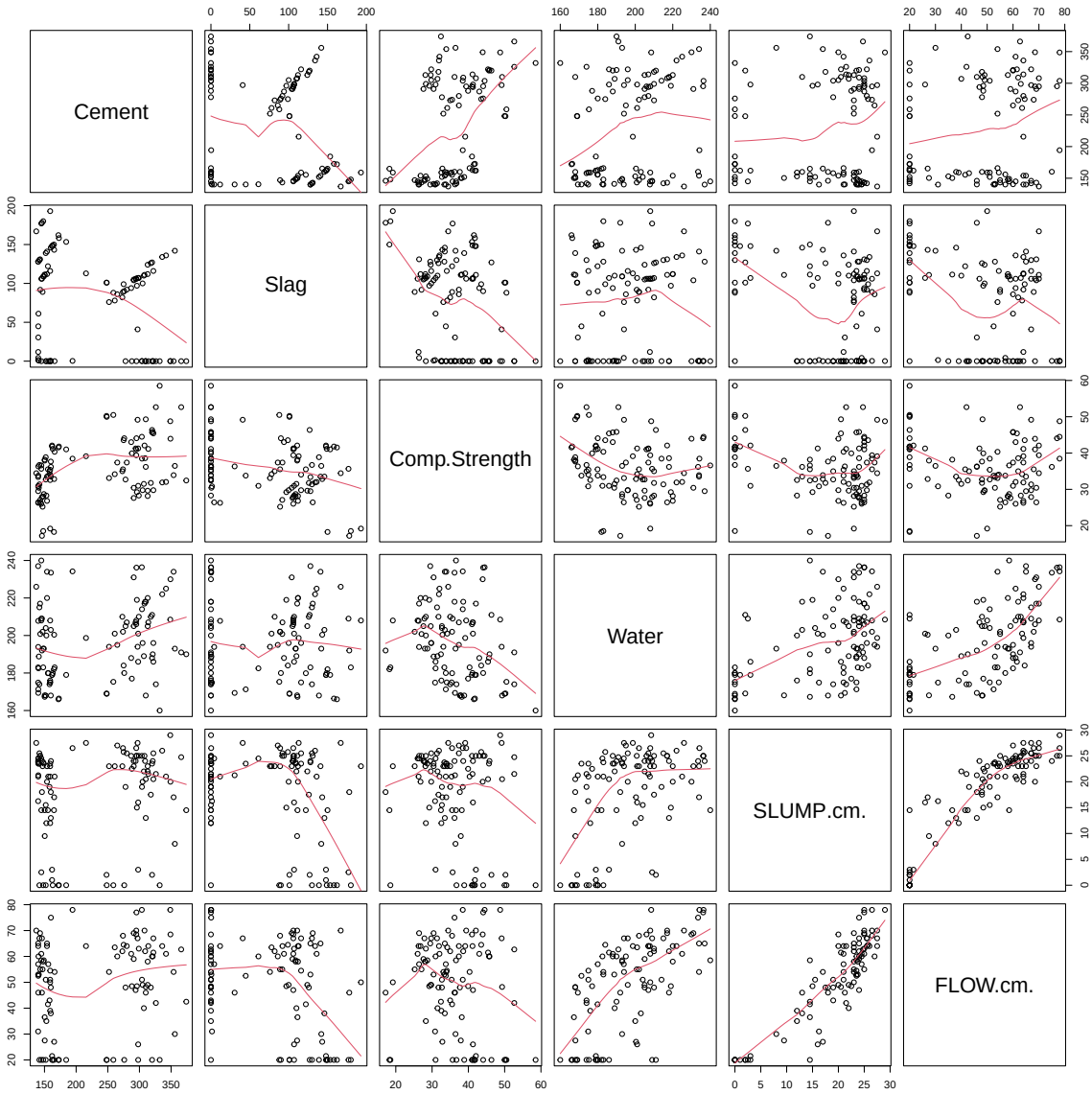



Figure 3.21: R: Scatterplot matrix for concrete dataset

**Python code**

```
pd.plotting.scatter_matrix(concrete[['Cement', 'Slag', 'Comp_Strength', 'Water',
                                     'SLUMP(cm)', 'FLOW(cm)']],
                           figsize=(12,12));
```

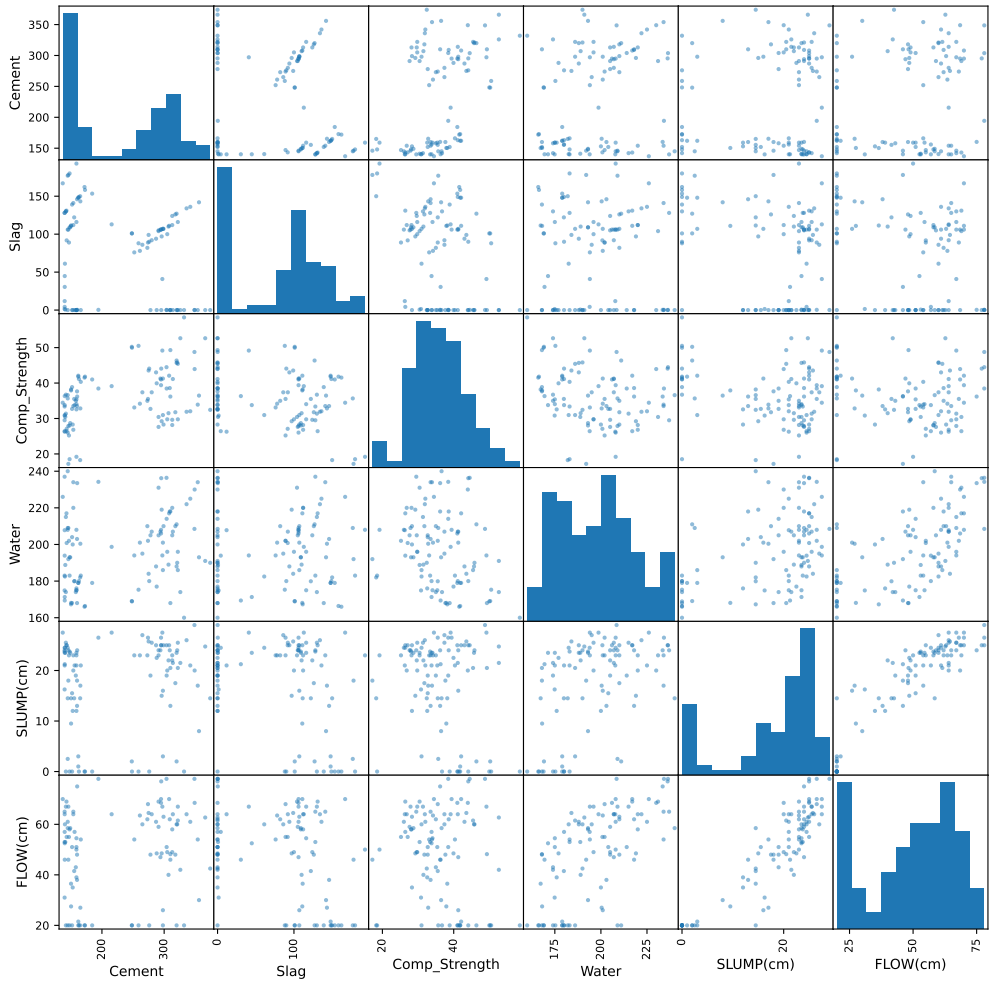



Figure 3.22: Py: Scatterplot matrix for concrete dataset

We can see that a few of the variables have several 0 values - cement, slag, slump and flow. Water appears to have some relation with slump and with flow.

The scatterplots allow a visual understanding of the patterns, but it is usually also good to compute the correlation of all pairs of variables.

### R code

```
library(psych)
corPlot(cor(concrete[, col_to_use]), cex=0.8, cex.axis=0.6,
        show.legend = FALSE)
```

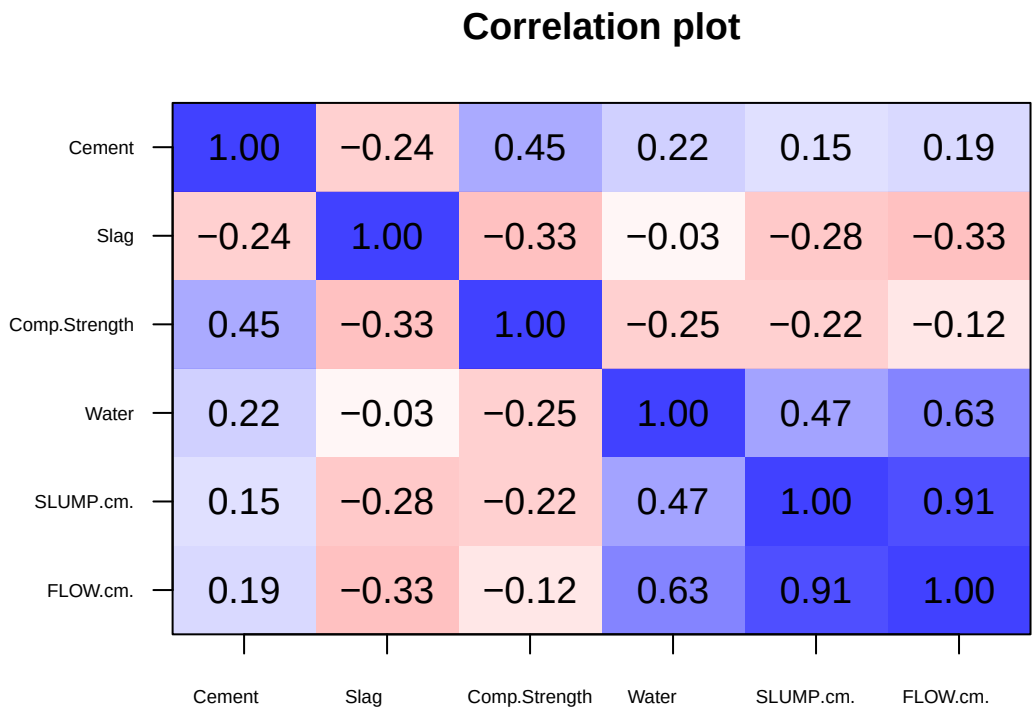



Figure 3.23: R: Heatmap of correlation matrix for concrete data

### Python code

The interactive heatmap created by Python is only available in the HTML format of the textbook.

```
corr = concrete[['Cement', 'Slag', 'Comp_Strength', 'Water',
                 'SLUMP(cm)', 'FLOW(cm)']].corr()
corr.style.background_gradient(cmap='coolwarm_r')
```

Table 3.5

| | Cement | Slag | Comp_Strength | Water | SLUMP(cm) | FLOW(cm) |
|---|---|---|---|---|---|---|
| Cement | 1.000000 | -0.243553 | 0.445725 | 0.221091 | 0.145913 | 0.186461 |
| Slag | -0.243553 | 1.000000 | -0.331588 | -0.026775 | -0.284037 | -0.327231 |
| Comp_Strength | 0.445725 | -0.331588 | 1.000000 | -0.254235 | -0.223358 | -0.124029 |
| Water | 0.221091 | -0.026775 | -0.254235 | 1.000000 | 0.466568 | 0.632026 |
| SLUMP(cm) | 0.145913 | -0.284037 | -0.223358 | 0.466568 | 1.000000 | 0.906135 |
| FLOW(cm) | 0.186461 | -0.327231 | -0.124029 | 0.632026 | 0.906135 | 1.000000 |

The plots you see above are known as heatmaps. They enable us to pick out groups of variables that are similar to one another. As you can see from the blue block in the lower right corner, `Water`, `SLUMP.cm` and `FLOW.cm` are very similar to one another.

## 3.6 References

### 3.6.1 Website References

1. UCI Machine Learning Repository: This site is an invaluable learning resource for data analysts. It contains several datasets for practice. In this chapter, we used the following two:
   - Student Performance Dataset.
   - Concrete Dataset
2. Kernel density estimation: Wikipedia contains a good overview explanation of the technique.

## 3.7 Exercises

In the questions, do try to answer each question using both R and Python whenever possible.

1. From Figure 3.6 alone, can you make a rough guess: What proportion of students scored 6 or less?
2. Divide the concrete data into two subsets - one with Water less than or equal to 200 and one with Water greater than 200. Compute the Pearson correlations with compressive strength. How different are these values with the value for the full dataset?
3. Remove all rows with zero values for Slag from the concrete dataset. Create a QQ-plot for Slag and assess if the data is Normal.
4. How does bandwidth affect a density estimate (`bw` argument)? Experiment with different values to understand more.
5. Create numerical summaries of `G3` scores, by `goout`, and compare the groups.

# 4 Exploring Categorical Data

## 4.1 Introduction

A variable is known as a *categorical variable* if each observation belongs to one of a set of categories. Examples of categorical variables are: gender, religion, race and type of residence.

Categorical variables are typically modeled using discrete random variables, which are strictly defined in terms whether or not the support is countable. The alternative to categorical variables are quantitative variables, which were introduced in the previous chapter.

Another method for distinguishing between quantitative and categorical variables is to ask if there is a meaningful distance between any two points in the data. If such a distance is meaningful then we have quantitative data. For instance, it makes sense to compute the difference in systolic blood pressure between subjects, but it does not make sense to consider the mathematical operation ("smoker" - "non-smoker").

It is important to identify which type of data we have (quantitative or categorical), since it affects the exploration techniques that we can apply.

There are two sub-types of categorical variables:

- A categorical variable is *ordinal* if the observations can be ordered, but do not have specific quantitative values.
- A categorical variable is *nominal* if the observations can be classified into categories, but the categories have no specific ordering.

In this chapter, we shall discuss techniques for identifying the presence, and for measuring the strength, of the association between two categorical variables. We shall also demonstrate common visualisations used with categorical data.

## 4.2 Contingency Tables

**Example 4.1** (Chest Pain and Gender)**.** Suppose that 1073 NUH patients who were at high risk for cardiovascular disease (CVD) were randomly sampled. They were then queried on two things:

1. Had they experienced the onset of severe chest pain in the preceding 6 months? (yes/no)
2. What was their gender? (male/female)

The data would probably have been recorded in the following format (only first few rows shown):

| Patient | Gender | Pain |
|---|---|---|
| 1 | female | no pain |
| 2 | male | no pain |

| Patient | Gender | Pain |
|---|---|---|
| 3 | female | no pain |
| 4 | male | no pain |
| 5 | female | no pain |
| 6 | male | pain |

However, it would probably be summarised and presented in this format, which is known as a *contingency table*.

| | pain | no pain |
|---|---|---|
| male | 46 | 474 |
| female | 37 | 516 |

In a contingency table, each observation from the dataset falls in exactly one of the cells. The sum of all entries in the cells equals the number of independent observations in the dataset.

All the techniques we shall touch upon in this chapter are applicable to contingency tables.

## 4.3 Visualisations

### 4.3.1 Bar charts

A common method for visualising cross combinations of categorical data is to use a bar chart.

**Example 4.2** (Political Association and Gender Barchart)**.** Consider data given in the table below where both variables are nominal. It cross-classifies poll respondents according to their gender and their political party affiliation.

| | Dem | Ind | Rep |
|---|---|---|---|
| female | 762 | 327 | 468 |
| male | 484 | 239 | 477 |

Here is R code to make a barchart for this data.

```
library(lattice)
x <- matrix(c(762,327,468,484,239,477), ncol=3, byrow=TRUE)
dimnames(x) <- list(c("female", "male"),
                    c("Dem", "Ind", "Rep"))
political_tab <- as.table(x)
barchart(political_tab/rowSums(political_tab),
         main="Political association by gender",
         horizontal = FALSE, auto.key=TRUE)
```

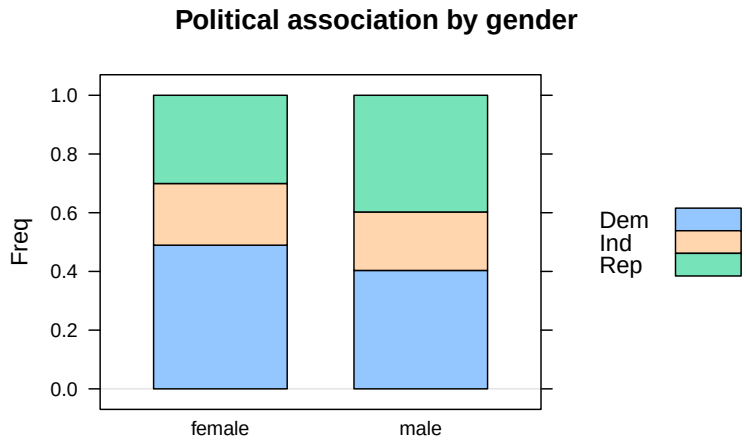



Figure 4.1: R: Stacked barchart for political association data

We can see that the proportion of Republicans is higher for males than for females. The corresponding proportion of Democrats is lower. However, note that the barchart does not reflect that the marginal count for males was much less than that for females.

Let us turn to making bar charts with pandas and Python.

**Example 4.3** (Claritin and Nervousness Barchart)**.** Claritin is a drug for treating allergies. However, it has a side effect of inducing nervousness in patients. From a sample of 450 subjects, 188 of them were randomly assigned to take Claritin, and the remaining were assigned to take the placebo. The observed data was as follows:

| | nervous | not nervous |
|---|---|---|
| claritin | 4 | 184 |
| placebo | 2 | 260 |

A barchart can be created in Python with the following code. This barchart is slightly different from the one in Example 4.2 because it is not stacked.

```
import numpy as np
import pandas as pd

claritin_tab = np.array([[4, 184], [2, 260]])
claritin_prop = claritin_tab/claritin_tab.sum(axis=1).reshape((2,1))

xx = pd.DataFrame(claritin_prop,
                  columns=['nervous', 'not_nervous'],
                  index=['claritin', 'placebo'])

ax = xx.plot(kind='bar', stacked=False, rot=1.0, figsize=(10,4),
             title='Nervousness by drug')
ax.legend(loc='upper left');
```

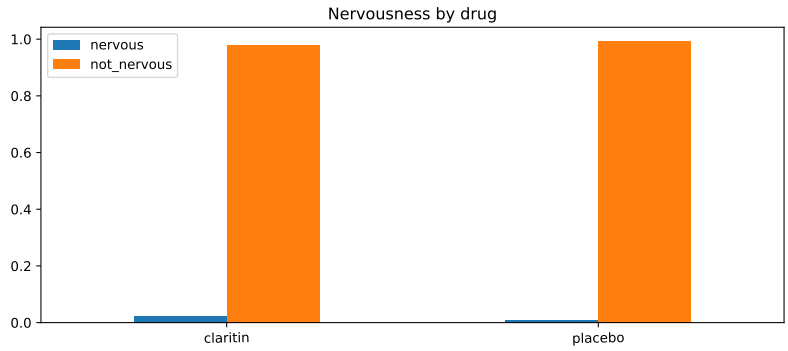



Figure 4.2: Py: Barchart for Claritin data

The proportion of patients who are not nervous is similar in both groups.

### 4.3.2 Mosaic plots

Unlike a bar chart, a mosaic plot reflects the count in each cell (through the area), along with the proportions of interest. Let us inspect how we can make mosaic plots for the political association data earlier.

**Example 4.4** (Political Association and Gender Mosaic Plot)**.** The colours in Figure 4.3 reflect the sizes of the standardised residuals. We shall discuss this in more detail in Section 4.4.3.

The greater width of the tiles in the column for females reflects the larger marginal count for females. Compare this visualisation with Figure 4.1.

#### R code

```
mosaicplot(political_tab, shade=TRUE, main="Political association by gender")
```

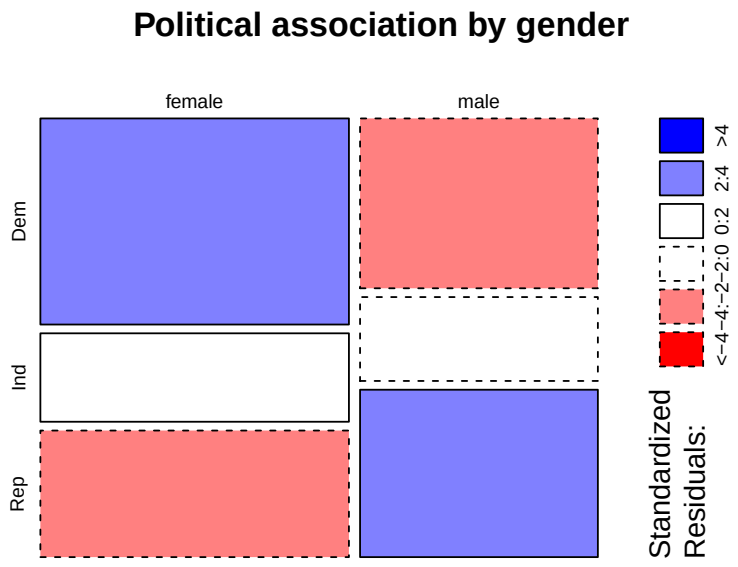



Figure 4.3: R: Mosaic plot for political association data

### Python code

```
from statsmodels.graphics.mosaicplot import mosaic
import matplotlib.pyplot as plt

political_tab = np.asarray([[762,327,468], [484,239,477]])
mosaic(political_tab, statistic=True, gap=0.05);
```

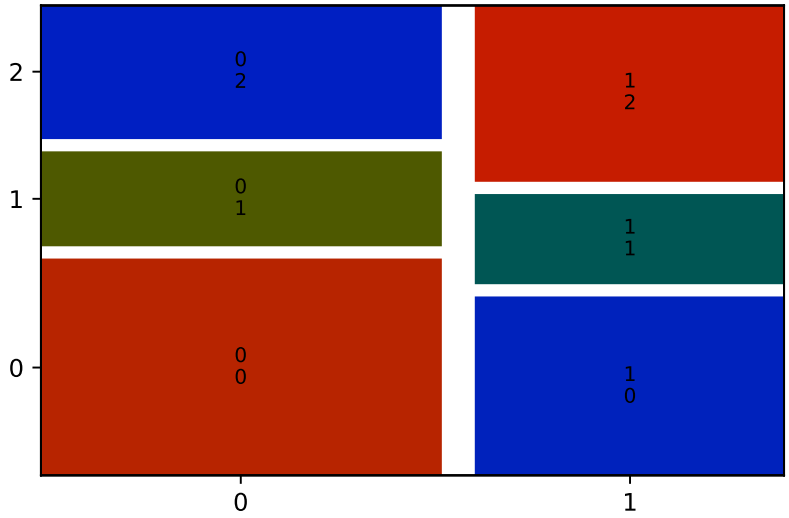



Figure 4.4: Py: Mosaic plot for political association data

### 4.3.3 Conditional Density Plots

When we have one categorical and one quantitative variable, the kind of plot we make really depends on which is the response, and which is the explanatory variable. If the response variable is the quantitative one, it makes sense to create boxplots or histograms. However, if the response variable is a the categorical one, we should really be making something along these lines:

**Example 4.5** (Heart Failure Dataset)**.** The dataset at UCI repository contains records on 299 patients who had heart failure. The data was collected during the follow-up period; each patient had 13 clinical features recorded. The primary variable of interest was whether they died or not. Suppose we wished to plot how this varied with age (a quantitative variable):

```
data_path <- file.path("data", "heart+failure+clinical+records",
                       "heart_failure_clinical_records_dataset.csv")
heart_failure <- read.csv(data_path)
spineplot(as.factor(DEATH_EVENT) ~ age, data=heart_failure,
          ylab = "Death", xlab="Age", main="Proportion dying, by age")
```

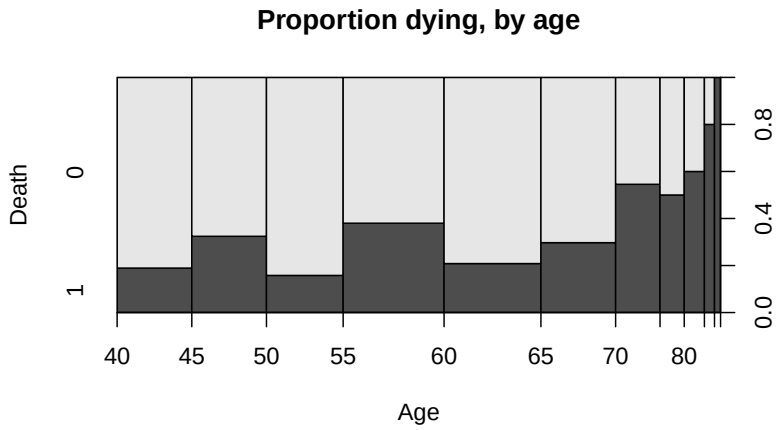



Figure 4.5: R: Spine plot for heart failure data

It reflects how the probability of an event varies with the quantitative explanatory variable. A smoothed version of this is known as the conditional density plot:

```
cdplot(as.factor(DEATH_EVENT) ~ age, data=heart_failure,
       ylab = "Death", xlab="Age", main="Proportion dying, by age")
```

Figure 4.6: R: Conditional density plot for heart failure data

From both Figure 4.5 and Figure 4.6, it is clear to see that there is an increased proportion of death during follow-up associated with older patients.

## 4.4 Tests for Independence

In the contingency table above, the two categorical variables are *Gender* and *Presence/Absence of Pain*. With contingency tables, the main inferential task usually relates to assessing the association between the two categorical variables.

> **Independent Categorical Variables**
>
> If two categorical variables are **independent**, then the joint distribution of the variables would be equal to the product of the marginals. If two variables are not independent, we say that they are **associated**.

In the remainder of this section, we are going to discuss and apply statistical hypothesis tests. Refer to Section 7.2 for a quick recap about hypothesis testing.

### 4.4.1 $\chi^2$-Test for Independence

The $\chi^2$-test uses the definition above to assess if two variables in a contingency table are associated. The null and alternative hypotheses are

$$\begin{aligned} H_0 &: \text{The two variables are indepdendent.} \\ H_1 &: \text{The two variables are not indepdendent.} \end{aligned}$$

Under the null hypothesis, we can estimate the joint distribution from the observed marginal counts. Based on this estimated joint distribution, we then compute *expected* counts (which may not be integers) for each cell. The test statistic essentially compares the deviation of *observed* cell counts from the expected cell counts.

**Example 4.6** (Chest Pain and Gender Expected Counts)**.** Continuing from Example 4.1, we can compute the estimated marginals using row and column proportions

```
chest_tab (table)

Summary:
n: 1'073, rows: 2, columns: 2

Pearson's Chi-squared test (cont. adj):
  X-squared = 1.4555, df = 1, p-value = 0.2276
Fisher's exact test p-value = 0.2089


                 pain   no pain   Sum

male     freq      46       474   520
         p.row   8.8%     91.2%     .
         p.col  55.4%     47.9% 48.5%

female   freq      37       516   553
         p.row   6.7%     93.3%     .
         p.col  44.6%     52.1% 51.5%

Sum      freq      83       990 1'073
         p.row   7.7%     92.3%     .
```

```
        p.col      .          .      .


' 95% conf. level
```

In the output above, ignore the $p$-values for now.. We'll get to those in a minute. Let $X$ represent gender and $Y$ represent chest pain. Then from the estimated proportions in the "Sum" row and colums, we can read off the following estimated probabilities

$$\begin{aligned}\widehat{P}(X = \text{male}) &= 0.485 \\ \widehat{P}(Y = \text{pain}) &= 0.077\end{aligned}$$

Consequently, *under* $H_0$, we would estimate

$$\widehat{P}(X = \text{male}, Y = \text{pain}) = 0.485 \times 0.077 \approx 0.04$$

From a sample of size 1073, the expected count for this cell is then

$$0.04 \times 1073 = 42.92$$

Using the approach above, we can derive a general formula for the expected count in each cell:

$$\text{Expected count} = \frac{\text{Row total} \times \text{Column total}}{\text{Total sample size}}$$

The formula for the $\chi^2$-test statistic (with continuity correction) is:

$$\chi^2 = \sum \frac{(|\text{expected} - \text{observed}| - 0.50)^2}{\text{expected count}} \tag{4.1}$$

The sum is taken over every cell in the table. Hence in a $2 \times 2$ table, as in Example 4.1, there would be 4 terms in the summation.

**Example 4.7** (Chest Pain and Gender $\chi^2$ Test)**.** Let us see how we can apply and interpret the $\chi^2$-test for the data in Example 4.1.

### R code

```r
x <- matrix(c(46, 37, 474, 516), nrow=2)
dimnames(x) <- list(c("male", "female"), c("pain", "no pain"))
chest_tab <- as.table(x)

chisq_output <- chisq.test(chest_tab)
chisq_output
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  chest_tab
X-squared = 1.4555, df = 1, p-value = 0.2276
```

**Python code**

```
from scipy import stats

chest_array = np.array([[46, 474], [37, 516]])

chisq_output = stats.chi2_contingency(chest_array)

print(f"The p-value is {chisq_output.pvalue:.3f}.")
## The p-value is 0.228.
print(f"The test-statistic value is {chisq_output.statistic:.3f}.")
## The test-statistic value is 1.456.
```

Since the $p$-value is 0.2276, we would not reject the null hypothesis at significance level 5%. We do not have sufficient evidence to conclude that the variables are not independent.

To extract the expected cell counts, we can use the following code:

**R code**

```
chisq_output$expected
```

```
           pain  no pain
male   40.22367 479.7763
female 42.77633 510.2237
```

**Python code**

```
chisq_output.expected_freq
```

```
array([[ 40.22367195, 479.77632805],
       [ 42.77632805, 510.22367195]])
```

The test statistic compares the above table to the *observed* table, earlier in Example 4.1.

> **Important**
>
> It is only suitable to use the $\chi^2$-test when all *expected cell counts* are larger than 5.

### 4.4.2 Fisher's Exact Test

When the condition above is not met, we turn to Fisher's Exact Test. The null and alternative hypothesis are the same, but the test statistic is not derived in the same way.

If the marginal totals are fixed, and the two variables are independent, it can be shown that the individual cell counts arise from the hypergeometric distribution, which is defined as follows:

Suppose we have an urn with $m$ black balls and $n$ red balls. From this urn, we draw a random sample (without replacement) of size $k$. If we let $W$ be the number of red balls drawn, then $W$ follows a hypergeometric distribution. The pmf is:

$$P(W = w) = \frac{\binom{n}{w}\binom{m}{k-w}}{\binom{n+m}{k}}$$

> **Note**
>
> What is the support of $W$?

To transfer this to the context of $2 \times 2$ tables, suppose we have fix the marginal counts of the table, and consider the count in the top-left corner to be a random variable following a hypergeometric distribution, with $r_{1.}$ red balls and $r_{2.}$ black balls. Consider drawing a sample of size $c_{1.}$ from these $r_{1.} + r_{2.}$ balls.

| | | |
|---|---|---|
| $W$ | - | $r_{1.}$ (red balls) |
| - | - | $r_{2.}$ (black balls) |
| $c_{1.}$ (sample size drawn) | $c_{2.}$ | |

Note that, assuming the marginal counts are fixed, knowledge of one of the four entries in the table is sufficient to compute all the counts in the table.

The test statistic is the observed count, $w$, in this cell. The $p$-value is calculated as

$$P(W \leq w)$$

In practice, instead of summing over all values, the $p$-value is obtained by simulating tables with the same marginals as the observed dataset, and estimating the above probability.

Using Fisher's test sidesteps the need for a large sample size (which is required for the $\chi^2$ approximation to hold); hence the "Exact" in the name of the test.

**Example 4.8** (Claritin and Nervousness)**.** Here is the code to perform the Fisher Exact Test for the Claritin data.

**R code**

```r
y <-  matrix(c(4, 2, 184, 260), nrow=2)
dimnames(y) <- list(c("claritin", "placebo"), c("nervous", "not nervous"))
claritin_tab <- as.table(y)
fisher.test(claritin_tab)
```

```
    Fisher's Exact Test for Count Data

data:  claritin_tab
p-value = 0.2412
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
  0.399349 31.473382
sample estimates:
odds ratio
  2.819568
```

**Python code**

```python
fe_output = stats.fisher_exact(claritin_tab)

print(f"The p-value for the test is {fe_output.pvalue:.4f}.")
```

```
The p-value for the test is 0.2412.
```

As the $p$-value is 0.2412, we again do not have sufficient evidence to reject the null hypothesis and conclude that there is a significant association.

By the way, we can check (in R) to see that the $\chi^2$-test would not have been appropriate:

```r
chisq.test(claritin_tab)$expected
```

```
Warning in chisq.test(claritin_tab): Chi-squared approximation may be incorrect

          nervous not nervous
claritin 2.506667    185.4933
placebo  3.493333    258.5067
```

### 4.4.3 $\chi^2$-Test for $r \times c$ Tables

Thus far, we have considered the situation concerning categorical variables where each has only two possible outcomes (2x2 table). However, it is common to want to check the association between two nominal variables where one of them, or both, has more than 2 outcomes.

**Example 4.9** (Political Association and Gender)**.** Let us return to the political association data, which we plotted in Example 4.2. The R code for applying the test is identical to before

```
chisq.test(political_tab)
```

```
    Pearson's Chi-squared test

data:  political_tab
X-squared = 30.07, df = 2, p-value = 2.954e-07
```

In this case, there is strong evidence to reject $H_0$. At 5% level, we would reject the null hypothesis and conclude there is an association between gender and political affiliation.

In general, we might have $r$ rows and $c$ columns. The null and alternative hypotheses are identical to the 2x2 case, and the test statistic is computed in the same way. However, under the null hypothesis, the test statistic follows a $\chi^2$ distribution with $(r-1)(c-1)$ degrees of freedom.

The $\chi^2$-test is based on a model of independence - the expected counts are derived under this assumption. As such, it is possible to derive residuals and study them, to see where the data deviates from this model.

We define the *standardised residuals* to be

$$r_{ij} = \frac{n_{ij} - \mu_{ij}}{\sqrt{\mu_{ij}(1-p_{i+})(1-p_{+j})}}$$

where

- $n_{ij}$ is the observed cell count in row $i$ and column $j$ (cell $ij$).
- $\mu_{ij}$ is the *expected* cell count in row $i$ and column $j$
- $p_{i+}$ is the marginal probability of row $i$
- $p_{+j}$ is the marginal probability of column $j$.

The residuals can be obtained from the test output. Under $H_0$, the residuals should be close to a standard Normal distribution. If the residual for a particular cell is very large (or small), we suspect that lack of fit (to the independence model) arises from that cell.

For the political association table, the standardised residuals (from R) are:

```
chisq.test(political_tab)$stdres
```

```
               Dem        Ind        Rep
female  4.5020535  0.6994517 -5.3159455
male   -4.5020535 -0.6994517  5.3159455
```

## 4.5 Measures of Association

This sections covers bivariate measures of association for categorical variables.

### 4.5.1 Odds Ratio

The most generally applicable measure of association, for 2x2 tables, is the Odds Ratio (OR). Suppose we have $X$ and $Y$ to be Bernoulli random variables with (population) success probabilities $p_1$ and $p_2$.

We define the odds of success for $X$ to be

$$\frac{p_1}{1-p_1}$$

Similarly, the odds of success for random variable $Y$ is $\frac{p_2}{1-p_2}$.

In order to measure the strength of their association, we use the *odds ratio*:

$$\frac{p_1/(1-p_1)}{p_2/(1-p_2)}$$

The odds ratio can take on any value from 0 to $\infty$.

- A value of 1 indicates no association between $X$ and $Y$. If $X$ and $Y$ were independent, this is precisely what we would observe.
- Deviations from 1 indicate stronger association between the variables.
- Note that, for odds ratios, deviations from 1 are not symmetric. For a given pair of variables, an association of 0.25 or 4 is the same - it is just a matter of which variable we put in the numerator odds.

Due to the above asymmetry, we often use the log-odds-ratio instead:

$$\log \frac{p_1/(1-p_1)}{p_2/(1-p_2)}$$

- Log-odds-ratios can take values from $-\infty$ to $\infty$.
- A value of 0 indicates no association between $X$ and $Y$.
- Deviations from 0 indicate stronger association between the variables, and deviations are now symmetric; a log-odds-ratio of -0.2 indicates the same *strength* as 0.2, just the opposite direction.

To obtain a confidence interval for the odds-ratio, we work with the log-odds ratio and then exponentiate the resulting interval. Here are the steps for a $2 \times 2$ table:

1. Suppose we label the observed counts in the table as $n_{11}, n_{12}, n_{21}, n_{22}$.
2. The *sample* odds ratio is
$$\widehat{OR} = \frac{n_{11} \times n_{22}}{n_{12} \times n_{21}}$$
3. For a large sample size, it can be shown that $\log \widehat{OR}$ follows a Normal distribution. Hence a 95% confidence interval can be obtained through
$$\log \frac{n_{11} \times n_{22}}{n_{12} \times n_{21}} \pm z_{0.025} \times ASE(\log \widehat{OR})$$

where the ASE (Asymptotic Standard Error) of the estimator is

$$\sqrt{\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}}$$

**Example 4.10** (Chest Pain and Gender Odds Ratio)**.** Let us compute the confidence interval for the odds ratio in the chest pain and gender example from earlier.

#### R code

```
library(DescTools)
OddsRatio(chest_tab,conf.level = .95)
```

```
odds ratio     lwr.ci     upr.ci
 1.3534040  0.8626023  2.1234612
```

#### Python code

```
import statsmodels.api as sm
chest_tab2 = sm.stats.Table2x2(chest_array)

print(chest_tab2.summary())
```

```
               Estimate   SE   LCB    UCB  p-value
--------------------------------------------------
Odds ratio        1.353        0.863 2.123   0.188
Log odds ratio    0.303 0.230 -0.148 0.753   0.188
Risk ratio        1.322        0.872 2.004   0.188
Log risk ratio    0.279 0.212 -0.137 0.695   0.188
--------------------------------------------------
```

### 4.5.2 For Ordinal Variables

When both variables are ordinal, it is often useful to compute the strength (or lack) of any monotone trend association. It allows us to assess if

> As the level of $X$ increases, responses on $Y$ tend to increase toward higher levels, or responses on $Y$ tend to decrease towards lower levels.

For instance, perhaps job satisfaction tends to increase as income does. In this section, we shall discuss a measure for ordinal variables, analogous to Pearson's correlation for quantitative variables, that describes the degree to which the relationship is monotone. It is based on the idea of a concordant or discordant pair of subjects.

**Definition 4.1.**

- A **pair of subjects** is *concordant* if the subject ranked higher on $X$ also ranks higher on $Y$.
- A **pair** is *discordant* if the subject ranking higher on $X$ ranks lower on $Y$.
- A **pair** is *tied* if the subjects have the same classification on $X$ and/or $Y$.

If we let

- $C$: number of concordant pairs in a dataset, and
- $D$: number of discordant pairs in a dataset.

Then if $C$ is much larger than $D$, we would have reason to believe that there is a strong positive association between the two variables. Here are two measures of association based on $C$ and $D$:

1. Goodman-Kruskal $\gamma$ is computed as

$$\gamma = \frac{C - D}{C + D}$$

2. Kendall $\tau_b$ is

$$\tau_b = \frac{C - D}{A}$$

   where $A$ is a normalising constant that results in a measure that works better with ties, and is less sensitive than $\gamma$ to the cut-points defining the categories. However, $\gamma$ has the advantage that it is more easily interpretable.

For both measures, values close to 0 indicate a very weak trend, while values close to 1 (or -1) indicate a strong positive (negative) association.

**Example 4.11** (Job Satisfaction by Income)**.** Consider the following table, obtained from Agresti (2012). The original data come from a nationwide survey conducted in the US in 1996.

### R code

```
x <- matrix(c(1, 3, 10, 6,
              2, 3, 10, 7,
              1, 6, 14, 12,
              0, 1,  9, 11), ncol=4, byrow=TRUE)
dimnames(x) <- list(c("<15,000", "15,000-25,000", "25,000-40,000", ">40,000"),
                    c("Very Dissat.", "Little Dissat.", "Mod. Sat.",
                      "Very Sat."))
us_svy_tab <- as.table(x)

output <- Desc(x, plotit = FALSE, verbose = 3)
output[[1]]$assocs
```

```
                      estimate  lwr.ci  upr.ci
Contingency Coeff.      0.2419       -       -
Cramer V                0.1439  0.0000  0.1693
Kendall Tau-b           0.1524 -0.0083  0.3130
Goodman Kruskal Gamma   0.2211 -0.0085  0.4507
```

```
Stuart Tau-c              0.1395 -0.0082  0.2871
Somers D C|R              0.1417 -0.0080  0.2915
Somers D R|C              0.1638 -0.0116  0.3392
Pearson Correlation       0.1772 -0.0241  0.3647
Spearman Correlation      0.1769 -0.0245  0.3645
Lambda C|R                0.0377  0.0000  0.2000
Lambda R|C                0.0159  0.0000  0.0693
Lambda sym                0.0259  0.0000  0.1056
Uncertainty Coeff. C|R    0.0311 -0.0076  0.0699
Uncertainty Coeff. R|C    0.0258 -0.0069  0.0585
Uncertainty Coeff. sym    0.0282 -0.0072  0.0637
Mutual Information        0.0508       -       -
```

**Python code**

```python
from scipy import stats

us_svy_tab = np.array([[1, 3, 10, 6],
                       [2, 3, 10, 7],
                       [1, 6, 14, 12],
                       [0, 1,  9, 11]])

dim1 = us_svy_tab.shape
x = []; y=[]
for i in range(0, dim1[0]):
    for j in range(0, dim1[1]):
        for k in range(0, us_svy_tab[i,j]):
            x.append(i)
            y.append(j)

kt_output = stats.kendalltau(x, y)
print(f"The estimate of tau-b is {kt_output.statistic:.4f}.")
```

```
The estimate of tau-b is 0.1524.
```

The output shows that both $\gamma = 0.22$ and $\tau_b = 0.15$ are close to significant. The lower confidence limit is close to being positive.

## 4.6 Further readings

In general, I have found that R packages seem to have a lot more measures of association for categorical variables. In Python, the measures are spread out across packages.

Above, we have only scratched the surface of what is available. If you are keen, do read up on

1. Somer's D (for association between nominal and ordinal)
2. Mutual Information (for association between all types of pairs of categorical variables)

3. Polychoric correlation (for association between two ordinal variables)

Also, take note of how log odds ratios, $\tau_b$ and $\gamma$ work. Firstly, values close to 0 reflect weak association. Values of $a$ and $-a$ indicate the same *strength*, but different direction of association. This allows the same intuition that Pearson's correlation does. When you are presented with new metrics, try to understand them by asking similar questions about them.

## 4.7 References

### 4.7.1 Website References

1. Documentation pages from statsmodels:
   - Mosaic plots
   - Contingency tables
2. Documentation pages from scipy:
   - Fisher test with scipy
   - $\chi^2$-test
3. Heart failure data
4. More information on Kendall $\tau_b$ statistic
5. The $\chi^2$ test we studied is a *test of independence*. It is a variant of the $\chi^2$ goodness-of-fit test, which is used to assess if data come from a particular distribution. In our case, the presumed distribution is one where the variables are independent. Read more about the goodness-of-fit test here.

## 4.8 Exercises

1. Use Python to create the spineplot for the heart failure data.
2. Write a function in Python that converts a contingency table into "long" format.
3. What is the interpretation of $\gamma$ implied by the text?
4. From the political association data, extract the standardised residuals (under independence), and create a QQ-plot.
5. Apply the Fisher Exact test to the political association data. How does it work in this case?

# 5 Robust Statistics

## 5.1 Introduction

Introductory statistics courses describe and discuss inferential methods based on the assumption that data is Normally distributed. However, we never truly know the distribution from which our data has arisen. The sample itself may deviate from Normality in various ways. To begin with, we might observe that the data has heavier tails than a Normal distribution. Second, the data could suggest that the originating distribution is heavily skewed, unlike the symmetric Normal.

A third way in which the Normal-based method could breakdown is when our dataset has extreme values, referred to as outliers. In such cases, many investigators will drop the anomalous points and proceed with the analysis on the remaining observations. This is not ideal for the following reasons:

1. The sharp decision to reject an observation is wasteful. We can do better by down-weighting the dubious observations.
2. It can be difficult to spot or detect outliers in multivariate data.

Continuing to use Normal based methods will result in confidence intervals and hypothesis tests that have low power. Instead, statisticians have developed a suite of methods that are **robust** to the assumption of Normality. These techniques may be sub-optimal when the data is truly Normal, but they quickly outperform the Normal-based method as soon as the distribution starts to deviate from Normality.

In this chapter, we shall introduce the concept of robustness and estimators of location and scale that have this property. Although we only touch on basic statistics in this topic, take note that robust techniques exist for regression and ANOVA as well. It is a vastly under-used technique.

## 5.2 Notation

For the rest of this topic, let us settle on some notation. Suppose we have an i.i.d sample $X_i$ from a continuous pdf $f$, where $i = 1, \dots, n$.

- We use $q_{f,p}$ to refer to the $p$-th quantile of $f$, i.e.
$$P(X \leq q_{f,p}) = p$$
- For standard Normal quantiles, we use $z_p$.
$$\Phi(z_p) = P(Z \leq z_p) = p$$
- We denote the order statistics from the sample with $X_{(i)}$. In other words,
$$X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$$
We denote the sample median as $X_{(1/2)}$.

## 5.3 Datasets

For this topic, we shall work with a couple of datasets that are clearly not Normal.

**Example 5.1** (Copper in Wholemeal Flour)**.** The dataset `chem` comes from the package `MASS`. The data was recorded as part of an analytical chemistry experiment – the amount of copper ($\mu g$) in wholemeal flour was measured for 24 samples.

```
library(MASS)
hist(chem, breaks = 20, xlab="Amount of copper")
```

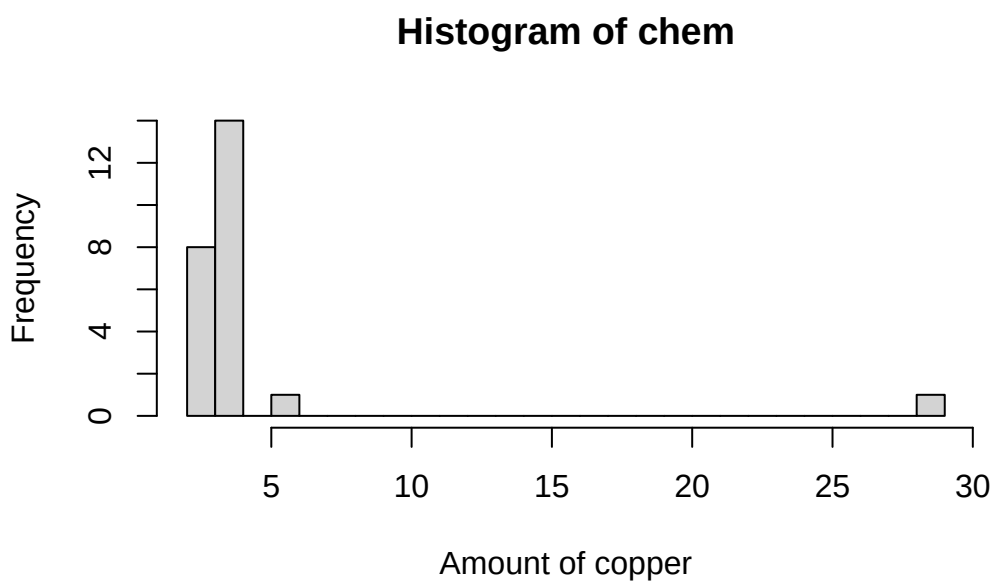



Figure 5.1: Copper measurements dataset

```
sort(chem)
```

```
 [1]  2.20  2.20  2.40  2.40  2.50  2.70  2.80  2.90  3.03  3.03  3.10  3.37
[13]  3.40  3.40  3.40  3.50  3.60  3.70  3.70  3.70  3.70  3.77  5.28 28.95
```

```
mean(chem)
```

```
[1] 4.280417
```

Although 22 out of the 24 points are less than 4, the mean is 4.28. This statistic is clearly being affected by the largest two values. This topic is about techniques that will work well even in the presence of such large anomalous values.

The second dataset is from a textbook on robust estimation techniques.

**Example 5.2** (Self-awareness Dataset)**.** For a second dataset, we use one on self-awareness from (Wilcox and R 2011), where participants in a psychometric study were timed how long they could keep a portion of an apparatus in contact with a specified target.

```
awareness <- c(77, 87, 88, 114, 151, 210, 219, 246, 253, 262, 296, 299, 306,
               376, 428, 515, 666, 1310, 2611)
hist(awareness, breaks=10, xlab="Timing")
```

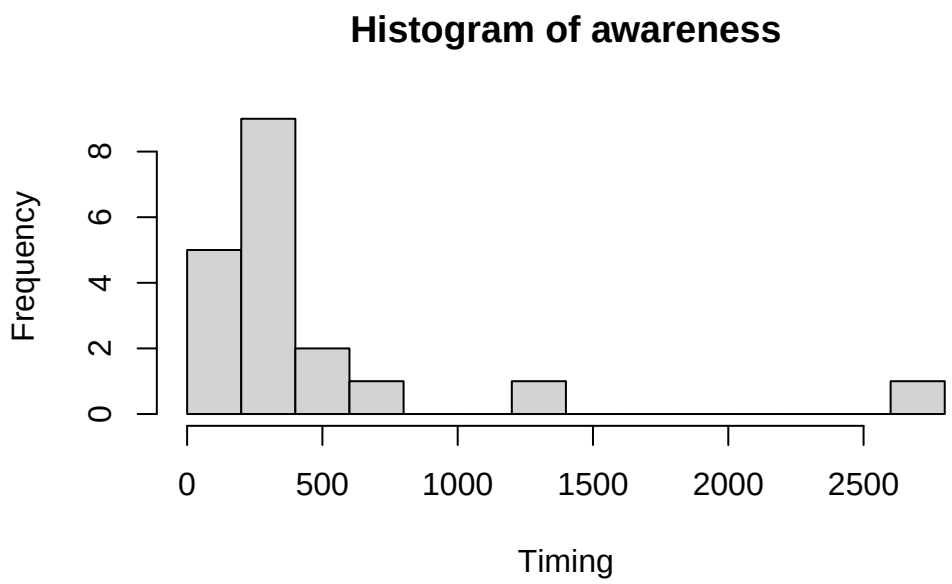



Figure 5.2: Self-awareness study timing

```
mean(awareness)
```

```
[1] 448.1053
```

Just like the data in Example 5.1, this data too is highly skewed to the right. The mean of the full dataset is larger than the 3rd quartile!

## 5.4 Assessing Robustness

The focus of this course is on the computational aspects of performing data analyses. However, this section and the next are a little theoretical. They only exist to provide a better overview of robust statistical techniques.

Section 5.4.1 introduces an approach for comparing two estimators in general. The remaining subsections (Section 5.5 onwards) briefly list ways in which robust statistics are evaluated.

### 5.4.1 Asymptotic Relative Efficiency

Suppose we wish to estimate a parameter $\theta$ of a distribution using a sample of size $n$. We have two candidate estimators $\hat{\theta}$ and $\tilde{\theta}$.

**Definition 5.1** (Asymptotic Relative Efficiency (ARE))**.** The asymptotic relative efficiency of $\tilde{\theta}$ relative to $\hat{\theta}$ is

$$ARE(\tilde{\theta}; \hat{\theta}) = \lim_{n \to \infty} \frac{\text{variance of } \hat{\theta}}{\text{variance of } \tilde{\theta}}$$

Usually, $\hat{\theta}$ is the optimal estimator according to some criteria. The intuitive interpretation is that when using $\hat{\theta}$, we only need $ARE$ times as many observations as when using $\tilde{\theta}$. Smaller values of ARE indicate that $\hat{\theta}$ is better than $\tilde{\theta}$.

Here are a couple of commonly used estimators and their AREs.

**Example 5.3** (Median versus Mean)**.** If our data is known to originate from a Normal distribution, due to its symmetry, we can use either the sample median *or* the sample mean to estimate $\mu$. Let $\hat{\theta} = \bar{X}$ and $\tilde{\theta} = X_{(1/2)}$.

Then it can be shown that

$$ARE(\tilde{\theta}; \hat{\theta}) = 2/\pi \approx 64\%$$

The sample median is *less efficient* than the sample mean, when the true distribution is Normal.

Here is an example of when robust statistics prove to be superior to non-robust ones.

**Example 5.4** (Contaminated Normal Variance Estimate)**.** Suppose first that we have $n$ observations $Y_i \sim N(\mu, \sigma^2)$, and we wish to estimate $\sigma^2$. Consider the two estimators:

1. $\hat{\sigma}^2 = s^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \bar{Y})^2$
2. $\tilde{\sigma}^2 = d^2 \pi / 2$, where
$$d = \frac{1}{n} \sum_i |Y_i - \bar{Y}|$$

In this case, when the underlying distribution truly is Normal, we have that

$$ARE(\tilde{\sigma}^2; \hat{\sigma}^2) = 87.6\%$$

However, now consider a situation where $Y_i \sim N(\mu, \sigma^2)$ with probability $1 - \epsilon$ and $Y_i \sim N(\mu, 9\sigma^2)$ with probability $\epsilon$. Let us refer to this as a *contaminated Normal* distribution.

As you can see from Figure 5.3, the two pdfs are almost indistinguishable by eye. However, the ARE values are very different:

| $\epsilon$ | ARE |
|---|---|
| 0 | 87.6% |
| 0.01 | 144% |

| $\epsilon$ | ARE |
|---|---|

The usual $s^2$ loses optimality very quickly; we can obtain more precise estimates using $\tilde{\sigma}^2$. This example was taken from section 5.5 of Venables and Ripley (2013).

There are three main approaches of assessing the robustness of an estimator. Let us cover the intuitive ideas here. In this section, $F$ refers to the cdf from which the sample was obtained.

## 5.5 Requirements of Robust Summaries

1. *Qualitative Robustness*: The first requirement of a robust statistic is that it if the underlying distribution $F$ changes slightly, then the estimate should not change too much.
2. *Infinitesimal Robustness*: The second requirement is tied to the concept of the *influence function* of an estimator. Roughly speaking, the influence function measures the relative extent that a small perturbation in $F$ has on the value of the estimate. In other words, it reflects the influence of adding one more observation to a large sample.
3. *Quantitative Robustness*: The final requirement is related to the contaminated distribution we touched on in Example 5.4. Consider

   $$F_{x,\epsilon} = (1-\epsilon)F + \epsilon\Delta_x$$

   where $\Delta_x$ is the degenerate probability distribution at $x$. The minimum value of $\epsilon$ for which the estimator goes to infinity as $x$ gets large, is referred to as the *breakdown point*. For the sample mean, the breakdown point is $\epsilon = 0$. For the sample median, the breakdown point is $\epsilon = 0.5$.

## 5.6 Measures of Location

The location parameter of a distribution is a value that characterises "a typical" observation, or the middle of the distribution. It is not always the mean of the distribution, but in the case of a symmetric distribution it will be.

### 5.6.1 M-estimators

Before we introduce robust estimators for the location, let us revisit the most commonly used one - the sample mean. Suppose we have observed $x_1, x_2, \dots, x_n$, a random sample from a $N(\mu, \sigma^2)$ distribution. As a reminder, here is how we derive the MLE for $\mu$.

The likelihood function is

$$L(\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i-\mu)^2/(2\sigma^2)}$$

The log-likelihood is

$$\log L = l(\mu, \sigma^2) = -n\log\sigma - \frac{n}{2}\log(2\pi) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2 \tag{5.1}$$

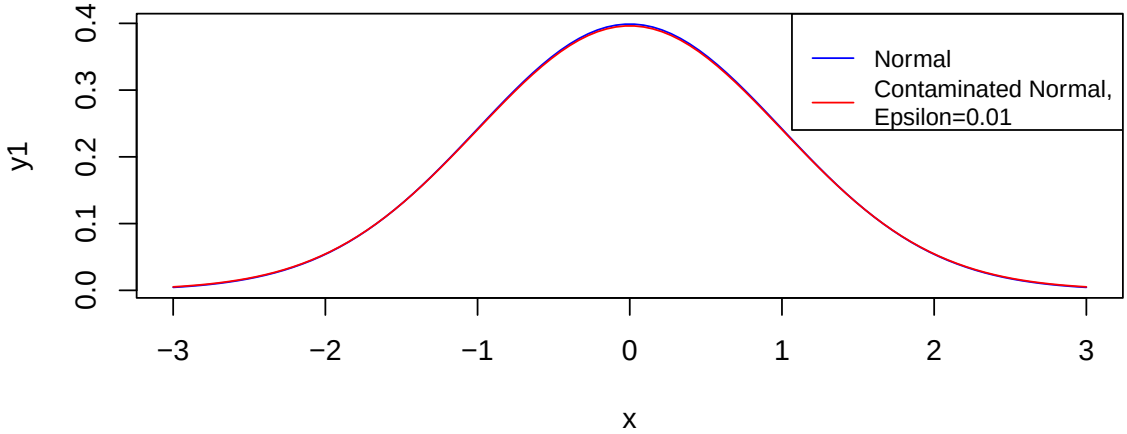



Figure 5.3: Contaminated Normal

Setting the partial derivative with respect to $\mu$ to be 0, we can solve for the MLE:

$$\begin{aligned} \frac{\partial l}{\partial \mu} &= 0 \\ \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i - \hat{\mu}) &= 0 \\ \hat{\mu} &= \bar{x} \end{aligned}$$

Observe that in Equation 5.1, we minimised the sum of squared errors, which arose from *minimising*

$$\sum_{i=1}^{n} -\log f(x_i - \mu)$$

where $f$ is the standard normal pdf. Instead of using $\log f$, Huber proposed using alternative functions (let's call the function $\rho$) to derive estimators (Huber 1992). The new estimator corresponds to

$$\arg\min_{\mu} \sum_{i=1}^{n} \rho(x_i - \mu) \tag{5.2}$$

The choice of $\rho$ confers certain properties on the resulting estimator. For instance, $\psi = \rho'$ is referred to as the *influence function*, which measures the relative change in a statistic as a new observation is added. To find the $\hat{\mu}$ that minimises Equation 5.2, it is equivalent to setting the derivative to zero and solving for $\hat{\mu}$:

$$\sum_{i=1}^{n} \psi(x_i - \mu) = 0$$

Note that, in general, the use of the sample mean corresponds to the use of $\rho(x) = x^2$. In that case, $\psi = 2x$ is unbounded, which results in high importance/weight placed on very large values. Instead, robust estimators should have a bounded $\psi$ function.

The approach outlined above - the use of $\rho$ and $\psi$ to define estimators, gave rise to a class of estimators known as M-estimators, since they are MLE-like. In the following sections, we shall introduce estimators corresponding to various choices of $\rho$. It is not always easy to identify the $\rho$ being used, but inspection of the form of $\psi$ leads to an understanding of how much emphasis the estimator places on large outlying values.

### 5.6.2 Trimmed mean

The $\gamma$-trimmed mean ($0 < \gamma \leq 0.5$) is the mean of a *distribution* after the distribution has been truncated at the $\gamma$ and $1-\gamma$ quantiles. Note that the truncated function has to be renormalised in order to be a pdf.

In formal terms, suppose that $X$ is a continuous random variable with pdf $f$. The usual mean is of course just $\mu = \int xf(x)dx$. The trimmed mean of the distribution is

$$\mu_t = \int_{q_{f,\gamma}}^{q_{f,1-\gamma}} x \frac{f(x)}{1-2\gamma} dx \tag{5.3}$$

Using the trimmed mean focuses on the middle portion of a distribution. The recommended value of $\gamma$ is (0, 0.2]. For a sample $X_1, X_2, \dots, X_n$, the estimate is computed (see Wilcox and R (2011) page 54) using the following algorithm:

1. Compute the value $g = \lfloor \gamma n \rfloor$, where $\lfloor x \rfloor$ refers to the floor function[1].
2. Drop the largest $g$ and smallest $g$ values from the sample.
3. Compute
$$\hat{\mu_t} = X_t = \frac{X_{(g+1)} + \cdots X_{(n-g)}}{n-2g}$$

It can be shown that the influence function for the trimmed mean is

$$\psi(x) = \begin{cases} x, & -c < x < c \\ 0, & \text{otherwise} \end{cases}$$

which indicates that, with this estimator, large outliers have **no** effect on the estimator.

### 5.6.3 Winsorised Mean

The Winsorised mean is similar to the trimmed mean in the sense that it modifies the tail of the distribution. However, it works by replacing extreme observations with fixed moderate values. The corresponding $\psi$ function is

$$\psi(x) = \begin{cases} -c, & x < -c \\ x, & |x| < c \\ c, & x > c \end{cases}$$

Just like in the trimmed mean case, we decide on the value $c$ by choosing a value $\gamma \in (0, 0.2]$. To calculate the Winsorised mean from a sample $X_1, X_2, \dots, X_n$, we use the following algorithm:

[1] The largest integer less than or equal to $x$.

1. Compute the value $g = \lfloor \gamma n \rfloor$.
2. Replace the smallest $g$ values in the sample with $X_{(g+1)}$ and the largest $g$ values with $X_{(n-g)}$.
3. Compute the arithmetic mean of the resulting $n$ values.

$$X_w = \frac{g \cdot X_{(g+1)} + X_{(g+1)} + \cdots + X_{(n-g)} + g \cdot X_{(n-g)}}{n}$$

**! Important**

Note that the trimmed mean and the Winsorised mean are no longer estimating the population distribution mean $\int x f(x) dx$. The three quantities coincide only if the population distribution is symmetric.
When this is not the case, it is important to be aware of what we are estimating. For instance, using the trimmed/winsorised mean is appropriate if we are interested in what a "typical" observation in the middle of the distribution looks like.

## 5.7 Measures of Scale

### 5.7.1 Sample Standard Deviation

Just as in Section 5.6.1, the MLE of the population variance $\sigma^2$ is not robust to outliers. It is given by

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Here are a few robust alternatives to this estimator. However, take note that, just like in the case of location estimators, the following estimators are not estimating the standard deviation. We can modify them so that if the underlying distribution truly is Normal, then they do estimate $\sigma$. However, if the distribution is not Normal, we should treat them as they are: robust measures of the spread of the distribution.

### 5.7.2 Median Absolute Deviation

For a random variable $X \sim f$, the median absolute deviation $w$ is defined by

$$P(|X - q_{f,0.5}| \leq w) = 0.5$$

We sometimes refer to $w$ as $MAD(X)$. In other words, it is the median of the distribution associated with $|X - q_{f,0.5}|$; it is the *median of absolute deviations from the median.*

If observations are truly from a Normal distribution, it can be shown that $MAD$ estimates $z_{0.75}\sigma$. Hence, in general, MAD is divided by $z_{0.75}$ so that it coincides with $\sigma$ *if the underlying distribution is Normal.*

**Proposition 5.1** (MAD for Normal)**.** *For $X \sim N(\mu, \sigma^2)$, the following property holds:*

$$\sigma \approx 1.4826 \times MAD(X)$$

*Proof.* Note that, since the distribution is symmetric, median$(X) = \mu$. Thus

$$\begin{aligned} MAD(X) &= \text{median}(|X - \text{median(X)}|) \\ &= \text{median}(|X - \mu|) \end{aligned}$$

Thus, the $MAD(X)$ is a value $q$ such that

$$P(|X - \mu| \leq q) = 0.5$$

Equivalently, we need $q$ such that

$$P\left(\left|\frac{X-\mu}{\sigma}\right| \leq q/\sigma\right) = P(|Z| \leq q/\sigma) = 0.5$$

Remember that we can retrieve values for the standard Normal cdf easily from R or Python:

$$\begin{aligned} P(-q/\sigma \leq Z \leq q/\sigma) &= 0.5 \\ 1 - 2 \times \Phi(-q/\sigma) &= 0.5 \\ -q/\sigma &= -0.6745 \\ q &= 0.6745\sigma \end{aligned}$$

Thus $MAD(X) = 0.6745\sigma$. The implication is that we can estimate $\sigma$ in a standard Normal with

$$\hat{\sigma} = \frac{1}{0.6745} MAD(X) \approx \frac{1}{0.6745} MAD(X)$$

□

### 5.7.3 Interquartile Range

The general definition of $IQR(X)$ is

$$q_{f,0.75} - q_{f,0.25}$$

It is a linear combination of quantiles. Again, we can modify the IQR so that, if the underlying distribution is Normal, we are estimating the standard deviation $\sigma$.

**Proposition 5.2** (IQR for Normal)**.** *For $X \sim N(\mu, \sigma^2)$, the following property holds:*

$$\sigma \approx \frac{IQR(X)}{1.35}$$

*Proof.* For $X \sim N(\mu, \sigma^2)$, let $q_{0.25}$ and $q_{0.75}$ represent the 1st and 3rd quartiles of the distribution.

$$\begin{aligned} P(X \leq q_{0.25}) &= 0.25 \\ P\left(\frac{X-\mu}{\sigma} \leq \frac{q_{0.25}-\mu}{\sigma}\right) &= 0.25 \\ P\left(Z \leq \frac{q_{0.25}-\mu}{\sigma}\right) &= 0.25 \end{aligned}$$

Thus (from R or Python[2]) we know that

$$\begin{aligned} \frac{q_{0.25} - \mu}{\sigma} = z_{0.25} &= -0.6745 \\ \therefore q_{0.25} &= \mu - 0.6745\sigma \end{aligned}$$

Similarly, we can derive that $q_{0.75} = \mu + 0.6745\sigma$. Now we can derive that

$$IQR(X) = q_{0.75} - q_{0.25} \approx 1.35\sigma$$

The implication is that, if we have sample data from standard Normal, we can estimate $\sigma$ from the IQR using:

$$\hat{\sigma} = \frac{IQR(\{X_1, \dots X_n\})}{1.35}$$

□

## 5.8 Examples

**Example 5.5** (Location Estimates: Copper Dataset)**.** To begin, let us apply the three estimators of location to the chemical dataset.

### R code

```
mean(chem)
## [1] 4.280417

mean(chem, trim = 0.1) # using gamma = 0.1
## [1] 3.205

library(DescTools)
vals <- quantile(chem, probs=c(0.1, 0.9))
win_sample <- Winsorize(chem, vals) # gamma = 0.1
mean(win_sample)
## [1] 3.182375
```

### Python code

```
import pandas as pd
import numpy as np
from scipy import stats

chem = pd.read_csv("data/mass_chem.csv")

chem.chem.mean()
```

[2] In R: `qnorm(0.25)`

```
## np.float64(4.2804166666666665)

stats.trim_mean(chem, proportiontocut=0.1)
## array([3.205])

stats.mstats.winsorize(chem.chem, limits=0.1).mean()
## np.float64(3.185)
```

As we observe, the robust estimates are less affected by the extreme and isolate value 28.95. They are more indicative of the general set of observations.

**Example 5.6** (Scale Estimates: Copper Dataset)**.** Now we turn the scale estimators for the self-awareness dataset.

#### R code

```
sd(awareness)
## [1] 594.6295

mad(awareness, constant=1)
## [1] 114

IQR(awareness)
## [1] 221.5
```

#### Python code

```
awareness = np.array([77, 87, 88, 114, 151, 210, 219, 246, 253, 262, 296,
                      299, 306, 376, 428, 515, 666, 1310, 2611])

awareness.std()
## np.float64(578.7698292373723)

stats.median_abs_deviation(awareness)
## np.float64(114.0)

stats.iqr(awareness)
## np.float64(221.5)
```

## 5.9 Summary

In this topic, we have introduced the concept of robust statistics. Understanding how these methods work requires a large amount of theoretical derivations and set-up. However, although we have not gone into much depth in the notes, we shall investigate the value of these methods in our tutorials.

## 5.10 References

### 5.10.1 Website References

1. Wikipedia on Robust Statistics This page contains visualisations of the $\psi$ curves for the estimators mentioned above, along with others.
2. Scipy stats: This page contains information on the `median_abs_deviation()`, `trim_mean()`, and `mstats.winsorize()` methods in Python.

## 5.11 Exercises

1. List two ways in which data could possibly deviate from Normality.
2. Compute the trimmed mean for the `G3` variable in the student performance dataset, using $\gamma = 0.1$.
3. Winsorise the `chem` dataset, with $\gamma = 0.1$, and create QQ-plots for the original and new dataset.
4. Identify the error(s) in the following R implementation of MAD, and create a Python function with the correct algorithm.

```
mad2 <- function(x) {
  deviations <- abs(x - mean(x))
  median(deviations)
}
```

5. Gini's mean difference is defined to be the mean absolute difference between any two distinct elements of a vector. Write a function that implements it, and compare it with robust measures of spread that were discussed in this chapter.

# 6 Introduction to SAS

## 6.1 Introduction

SAS (Statistical Analysis System) is a software that was originally created in the 1960s. Today, it is widely used by statisticians working in biostatistics and the pharmaceutical industries. Unlike Python and R, it is a proprietary software. The full license is quite expensive for a low-usage case such as ours. Thankfully, there is a free web-based version that we can use for our course.

## 6.2 Registering for a SAS Studio Account

The first step is to create a SAS profile: Use your *NUS email address* to register and create your SAS profile using this link.

Once you have verified your account using the email that would be sent to you, the following link should take you to the login page shown in Figure 6.1.

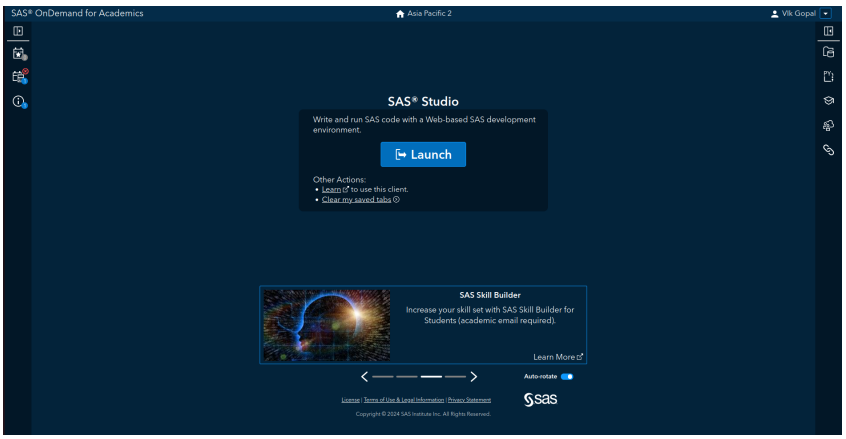



Figure 6.1: SAS: Login page

Subsequently logging in should take you to the landing page, where you can begin writing SAS code and using SAS. This interface can be seen in Figure 6.2.

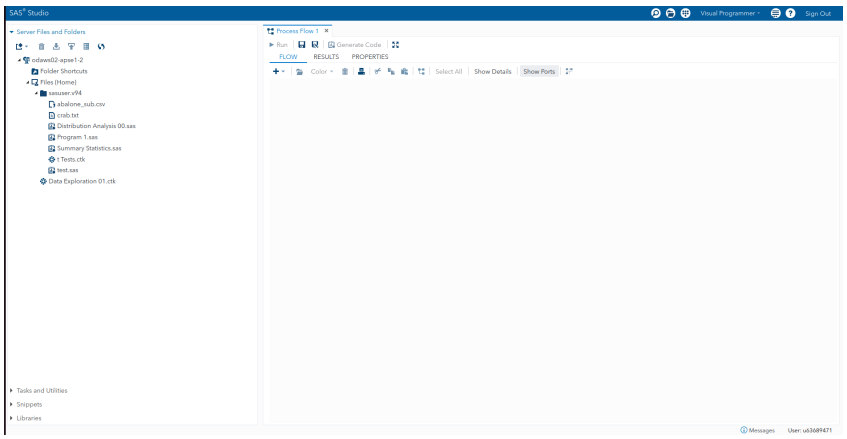



Figure 6.2: SAS: Studio interface

## 6.3 An Overview of SAS Language

The SAS language is not a fully-fledged programming language like Python is, or even R. For the most part, we are going to capitalise on the point-and-click interface of SAS Studio in our course. However, even so, it is good to understand a little about the language so that we can modify the options for different procedures as necessary.

A SAS program is a sequence of statements executed in order. Keep in mind that:

> Every SAS statement ends with a semicolon.

SAS programs are constructed from two basic building blocks: DATA steps and PROC steps. A typical program starts with a DATA step to create a SAS data set and then passes the data to a PROC step for processing.

**Example 6.1** (Creating and Printing a Dataset)**.** Here is a simple program that converts miles to kilometres in a DATA step and then prints the results with a PROC step:

```
DATA distance;
    Miles = 26.22;
    Kilometer = 1.61 * Miles;

PROC PRINT DATA=distance;
RUN;
```

To run the above program, click on the "Running Man" icon in SAS studio. You should obtain the output shown in Figure 6.3.

This dataset has only one observation (row).

*Data* steps start with the DATA keyword. This is followed by the name for the dataset. *Procedures* start with PROC followed by the name of the particular procedure (e.g. PRINT, SORT or PLOT) you wish to run on the dataset. Most SAS procedures have only a handful of possible statements. A step ends when SAS encounters a new step (marked by a DATA or PROC statement) or a RUN statement. RUN statements are not part of a DATA or PROC step; they are global statements.

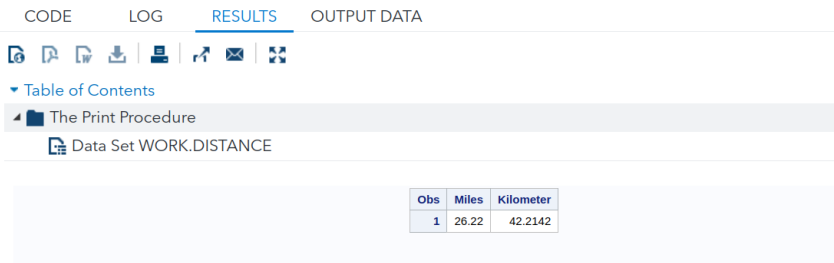



Figure 6.3: SAS: Simple data creation

**Example 6.2** (Creating a Dataset Inline)**.** The following program explicitly creates a dataset within the DATA step.

```
/*CREATING DATA MANUALLY:; */

DATA ex_1;
INPUT subject gender $ CA1 CA2 HW $;
DATALINES;
10 m 80 84 a
7 m 85 89 a
4 f 90 86 b
20 m 82 85 b
25 f 94 94 a
14 f 88 84 c
;

PROC MEANS DATA=ex_1;
VAR CA1 CA2;
RUN;
```

The output for the above code is shown in Figure 6.4 and Figure 6.5.

In the statements above, the $'s in the INPUT statement inform SAS that the preceding variables (gender and HW) are character. Note how the semi-colon for the DATALINES appears *after* all the data has been listed.

PROC MEANS creates basic summary statistics for the variables listed.

To review, there are only 2 types of steps in SAS programs:

## 6.4 Basic Rules for SAS Programs

### 6.4.1 For SAS statements

- All SAS statements (except those containing data) must end with a semicolon (;).
- SAS statements typically begin with a SAS keyword. (DATA, PROC).
- SAS statements are not case sensitive, that is, they can be entered in lowercase, uppercase, or a mixture of the two.
  - Example : SAS keywords (DATA, PROC) are not case sensitive

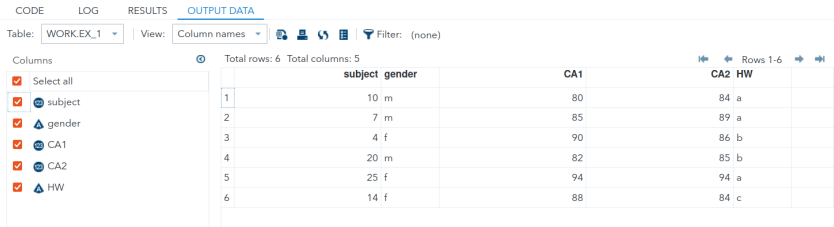



Figure 6.4: SAS: Inline data created

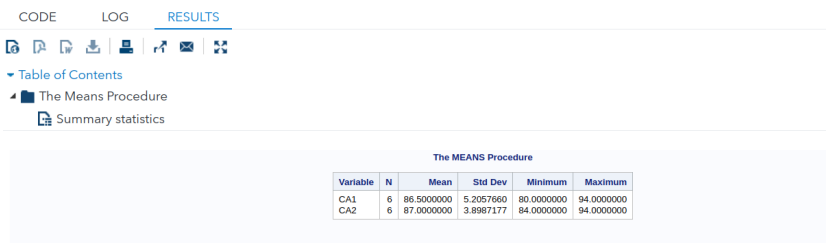



Figure 6.5: SAs: PROC MEANS output

**DATA steps**

- begin with DATA statements.
- read and modify data.
- create a SAS dataset.

**PROC steps**

- begin with PROC statements.
- perform specific analysis or function.
- produce reports or results.

- A delimited comment begins with a forward slash-asterisk (/) *and ends with an asterisk-forward slash (/)*. All text within the delimiters is ignored by SAS.

### 6.4.2 For SAS names

- All names must contain between 1 and 32 characters.
- The first character appearing in a name must be a letter (A, B, ... Z, a, b, ..., z) or an underscore ( ). Subsequent characters must be letters, numbers, or underscores. That is, no other characters, such as $, %, or & are permitted.
- Blanks also cannot appear in SAS names.
- SAS names are not case sensitive, that is, they can be entered in lowercase, uppercase, or a mixture of the two. (SAS is only case sensitive within quotation marks.)

### 6.4.3 For SAS variables

- If the variable in the INPUT statement is followed by a dollar sign ($), SAS assumes this is a character variable. Otherwise, the variable is considered as a numeric variable.

## 6.5 Reading Data into SAS

In this topic, we shall introduce a new dataset, also from the UCI Machine Learning repository.

**Example 6.3** (Bike Rentals)**.** The dataset was collected by the authors in Fanaee-T and Gama (2013). It contains information on bike-sharing rentals in Washington D.C. USA for the years 2011 and 2012, along with measurements of weather. The original dataset contained hourly and daily aggregated data. For our class, we use a re-coded version of the daily data. Our dataset can be found on Canvas as `bike2.csv`.

Here is the data dictionary:

| Field | Description |
|---|---|
| instant | Record index |
| dteday | Date |
| season | spring, summer, fall, winter) |
| yr | Year (0: 2011, 1: 2012) |
| mnth | Abbreviated month |
| holiday | Whether the day is a holiday or not |
| weekday | Abbreviated day of the week |
| workingday | yes: If day is neither weekend nor holiday is 1, no: Otherwise |
| weathersit | Weather situation: clear: Clear, Few clouds, Partly cloudy, Partly cloudy; mist: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist; light_precip: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds; heavy_precip: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog |
| temp | Normalized temperature in Celsius. Divided by 41 (max) |
| atemp | Normalized perceived temperature in Celsius. Divided by 50 (max) |
| hum | Normalized humidity. Divided by 100 (max) |
| windspeed | Normalized wind speed. Divided by 67 (max) |
| casual | Count of casual rental bike users |
| registered | Count of registered rental bike users |
| cnt | Count of total rental bikes including both casual and registered |

Our first step will be to load the dataset into SAS Studio.

## 6.6 Uploading and Using Datasets

To use our own datasets on SAS Studio, we have to execute the following steps:

1. Create a new library. In SAS, a library is a collection of datasets. If you already have a library created, you can simply import datasets into it. The default library on SAS is called `WORK`. However, the datasets in `WORK` will be purged every time you sign out. Hence it is better to create a new, dedicated library for this course.
2. Import your dataset (csv, xlsx, etc.) into the library.
3. After this, the data will be available for use with the reference name `<library-name>.<dataset-name>`.

From the "Libraries" menu on the left of SAS studio, click on the "New library" icon (the one circled in red in Figure 6.6), and create a new library called "ST2137". You can use the default suggested path for the library.

Now expand the menu for "Server Files and Folders" and upload `bike2.csv` file to SAS, using the circled icon in Figure 6.7.

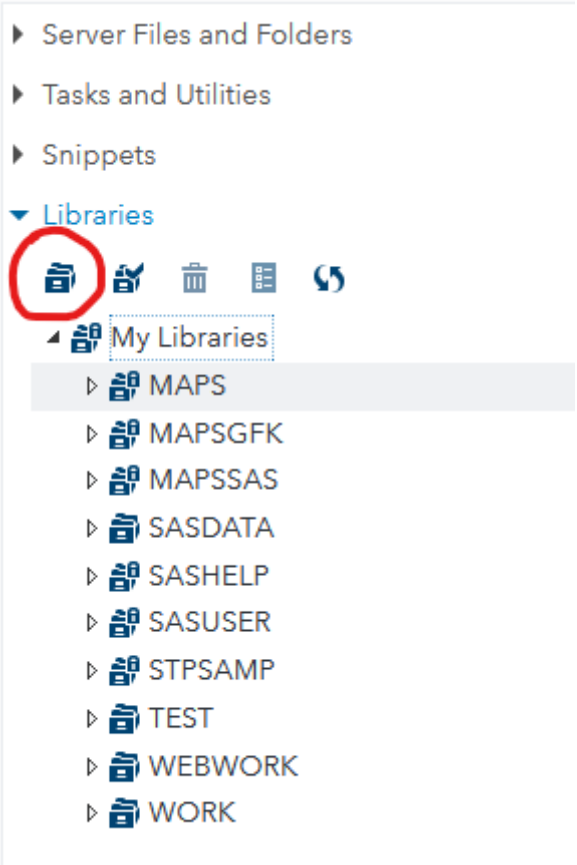



Figure 6.6: SAS: New library creation icon

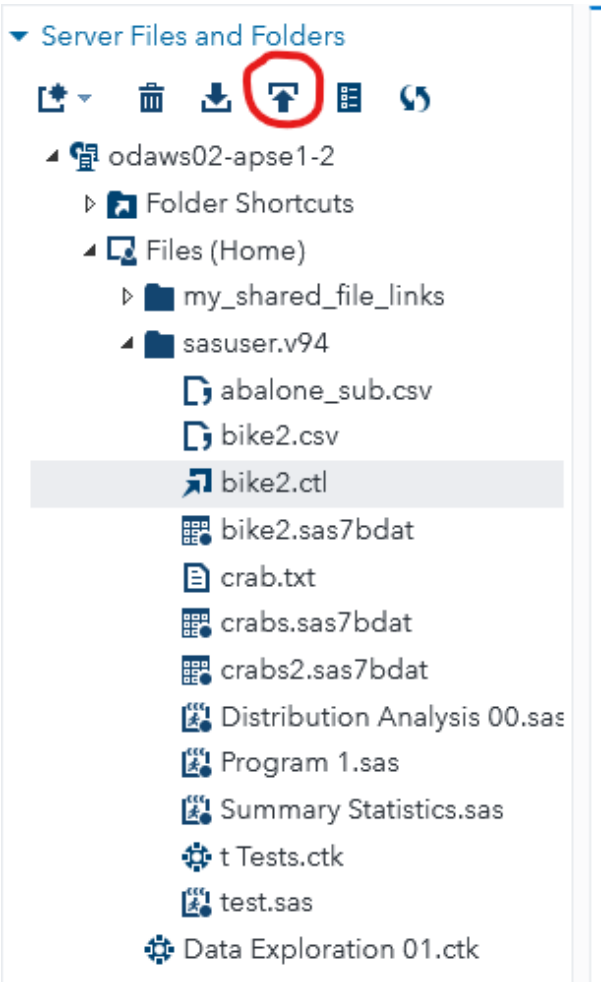



Figure 6.7: SAS: Upload data icon

Finally, right-click on the top of the main Studio area (where we write code) and select "New Import Data". Select the `bike2.csv` that has just been uploaded, and modify the OUTPUT DATA settings to Library `ST2137` and Data set name `BIKE2`. Click on the running man, and your dataset is now ready for use in SAS studio!

## 6.7 Summarising Numerical Data

Most of the SAS routines we are going to work with can be found in the "Tasks and Utilities" section (see highlighted tasks in Figure 6.8).

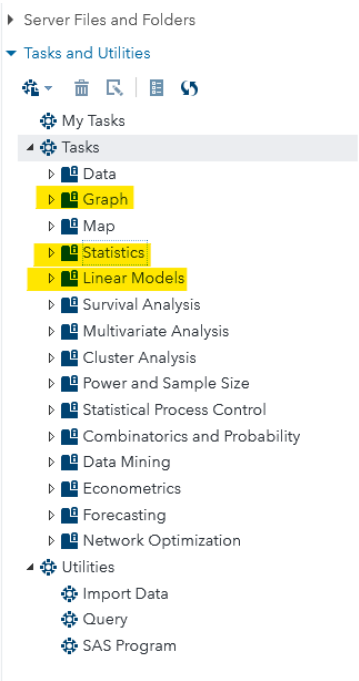



Figure 6.8: SAS: Common ST2137 procedures

### 6.7.1 Numerical Summaries

**Example 6.4** (5-number Summaries)**.** We expect that the total count of users will vary by the seasons. Hence, we begin by computing five-number summaries for each season.

Under Tasks, go to Statistcs > Summary Statistics. Select `cnt` as the analysis variable, and `season` as the classification variable. Under the options tab, select the lower and upper quartiles, along with comparative boxplots. The output should look like Figure 6.9:

We observe that the median count is highest for fall, followed by summer, winter and lastly spring. The spreads, as measured by IQR, are similar across the seasons: approximately 2000 users. In the middle 50%, the count distribution for spring is the most right-skewed.

### 6.7.2 Scatter Plots

**Example 6.5** (Casual vs Registered Scatterplot)**.** To create a scatterplot in SAS, go to Tasks > Graphs > Scatter Plot.

| Analysis Variable : cnt | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| season | N Obs | Mean | Std Dev | Minimum | Maximum | Median | N | Lower Quartile | Upper Quartile |
| fall | 188 | 5644.30 | 1459.80 | 1115.00 | 8714.00 | 5353.50 | 188 | 4583.00 | 6941.50 |
| spring | 181 | 2604.13 | 1399.94 | 431.0000000 | 7836.00 | 2209.00 | 181 | 1538.00 | 3456.00 |
| summer | 184 | 4992.33 | 1695.98 | 795.0000000 | 8362.00 | 4941.50 | 184 | 3996.00 | 6384.00 |
| winter | 178 | 4728.16 | 1699.62 | 22.0000000 | 8555.00 | 4634.50 | 178 | 3614.00 | 5629.00 |

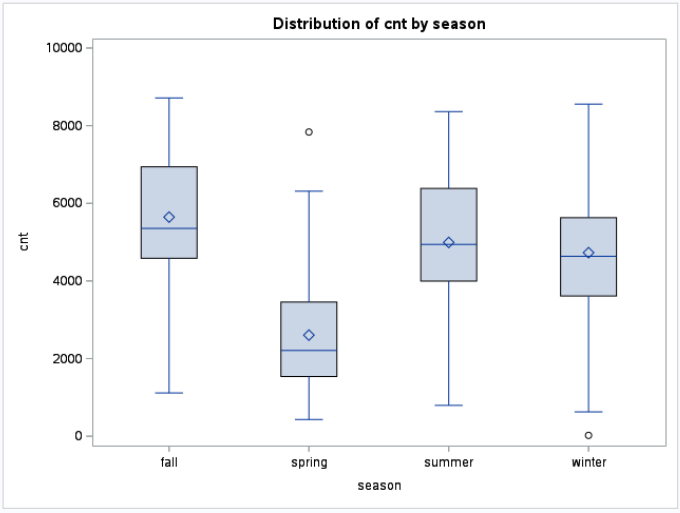



Figure 6.9: SAS: 5-number-summaries and boxplots for bike rental data

Specify `casual` on the x-axis, `registered` on the y-axis, and `workingday` as the Group. You should observe the plot created Figure 6.10:

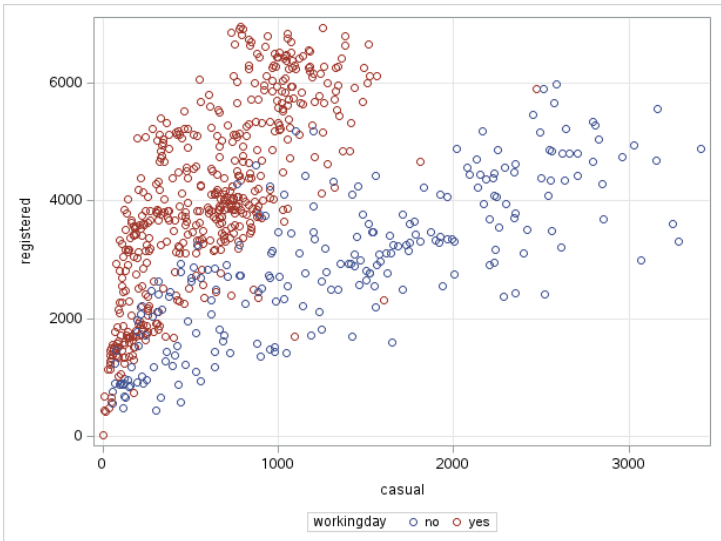



Figure 6.10: SAS: Scatterplot for bike rental data

We can see that there seem to be two different relationships between the counts of casual and registered users. The two relationships correspond to whether it as a working day or not.

### 6.7.3 Histograms

**Example 6.6** (Casual Users Distribution)**.** Now suppose we focus on casual users, and study the distribution of counts by whether a day is a working day or not. To create a histogram,

go to Tasks > Graph > Histogram. Select `casual` as the analysis variable, and `workingday` as the group variable.

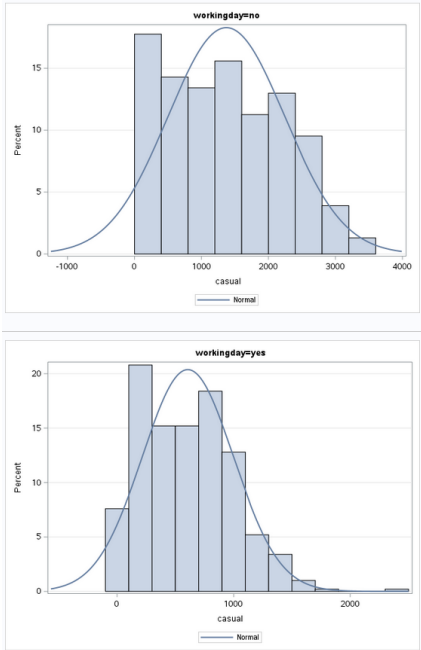



Figure 6.11: SAS: Histograms for bike rental data

From Figure 6.11, we can see that the distribution is right-skewed in both cases. However, the range of counts for non-working days extends further, to about 3500.

### 6.7.4 Boxplots

**Example 6.7** (Boxplots for Casual Users, by Season)**.** In Example 6.4, we observed that total counts vary by season, and in Example 6.6, we observed that working days seem to have fewer casual users. Let us investigate if this difference is related to season.

To create boxplots, go to Tasks > Box Plot. Select `casual` as the analysis variable, `season` as the category and `workingday` as the subcategory. You should obtain the plot in Figure 6.12.

To order the seasons according to the calendar, I had to add this line to the code:

```
proc sgplot data=ST2137.BIKE2;
    vbox casual / category=season group=workingday grouporder=ascending;
    xaxis values=('spring' 'summer' 'fall' 'winter');
    yaxis grid;
run;
```

There is little further insight compared to the previous two examples. However, now try the same plots, but on the log scale (modify the APPEARANCE tab and re-run). You should now obtain Figure 6.13.

Now, we can observe that the difference within each season is constant across seasons. Because the difference in logarithms is constant, it means that, on the original scale, it is a constant multiplicative factor that increases counts from workingday to non-working day.

We have arrived at a more succint representation of the relationship by using the log transform.

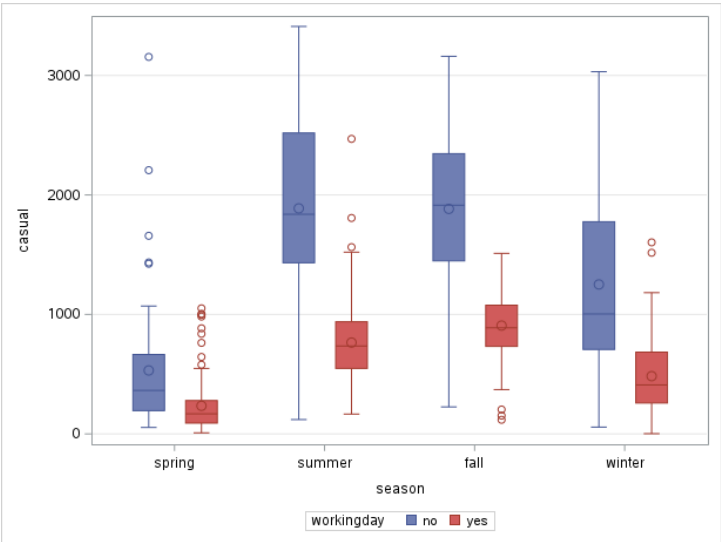



Figure 6.12: SAS: Boxplots for bike rentals, by season

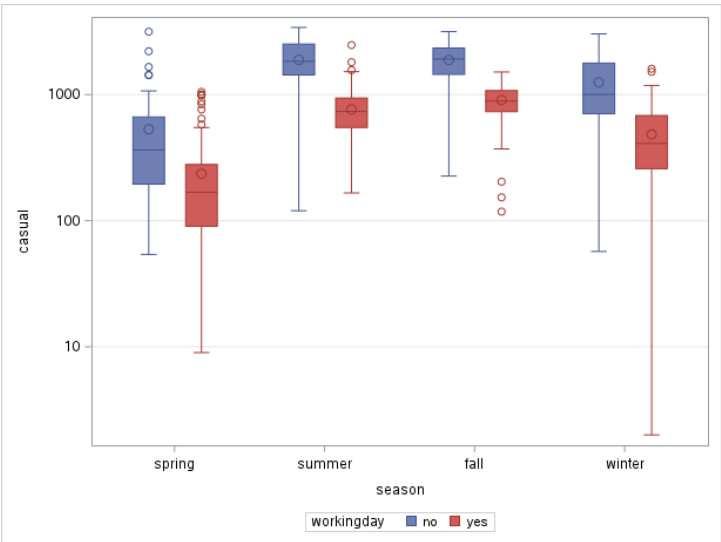
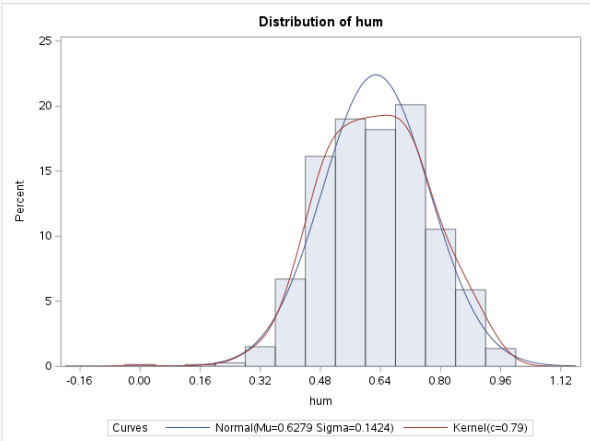



Figure 6.13: SAS: Boxplots for bike rentals on log-scale, by season

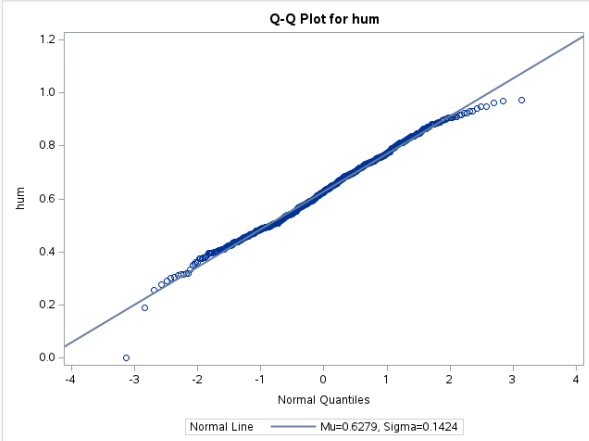
### 6.7.5 QQ-plots

**Example 6.8** (Normality Check for Humidity)**.** To create QQ-plots, we go to Tasks > Statistics > Distribution Analysis.

Select `hum` for the analysis variable. Under options, add the normal curve, the kernel density estimate, and the Normal quantile-quantile plot. You should obtain the following two charts:




(a) SAS: Histogram for humidity




(a) sAS: QQ-plot for humidity

The plot shows that humidity values are quite close to a Normal distribution, apart from a single observation on the left.

## 6.8 Categorical Data

We now turn to categorical data methods with SAS. We return to the dataset on student performance that we used in the topic on summarising data. Upload and store `student-mat.csv` as ST2137.STUD_PERF on the SAS Studio website.

**Example 6.9** ($\chi^2$ Test for Independence)**.** For a test of independence of `address` and `paid`, go to Tasks > Table Analysis, and select:

- `address` as the column variable
- `paid` as the row variable.
- Under OPTIONS, check the "Chi-square statistics" box.

The output in Figure 6.16 and Figure 6.17 should enable you to perform the test.

For measures of association, we only need to select the option for "Measures of Association" to generate the Kendall $\tau_b$ that we covered earlier in Chapter 4.

**Example 6.10** (Kendall $\tau$ for `Walc` and `Dalc`)**.** Once we load the data ST2137.STUD_PERF, we go to Tasks > Table Analysis. After selecting the two variables, we check the appropriate box to obtain Figure 6.19.

You may observe that the particular associations computed and returned are similar to those by the `Desc` R package that we used in Example 4.10.

| Frequency<br>Expected<br>Col Pct | Table of paid by address | | |
|---|---|---|---|
| | address | | |
| paid | R | U | Total |
| no | 52<br>47.676<br>59.09 | 162<br>166.32<br>52.77 | 214 |
| yes | 36<br>40.324<br>40.91 | 145<br>140.68<br>47.23 | 181 |
| Total | 88 | 307 | 395 |

Figure 6.16: SAS: Observed & expected counts, student performance data

Statistics for Table of paid by address

| Statistic | DF | Value | Prob |
|---|---|---|---|
| Chi-Square | 1 | 1.1012 | 0.2940 |
| Likelihood Ratio Chi-Square | 1 | 1.1070 | 0.2927 |
| Continuity Adj. Chi-Square | 1 | 0.8612 | 0.3534 |
| Mantel-Haenszel Chi-Square | 1 | 1.0984 | 0.2946 |
| Phi Coefficient | | 0.0528 | |
| Contingency Coefficient | | 0.0527 | |
| Cramer's V | | 0.0528 | |

Figure 6.17: SAS: Test statistics and p-values for student performance data

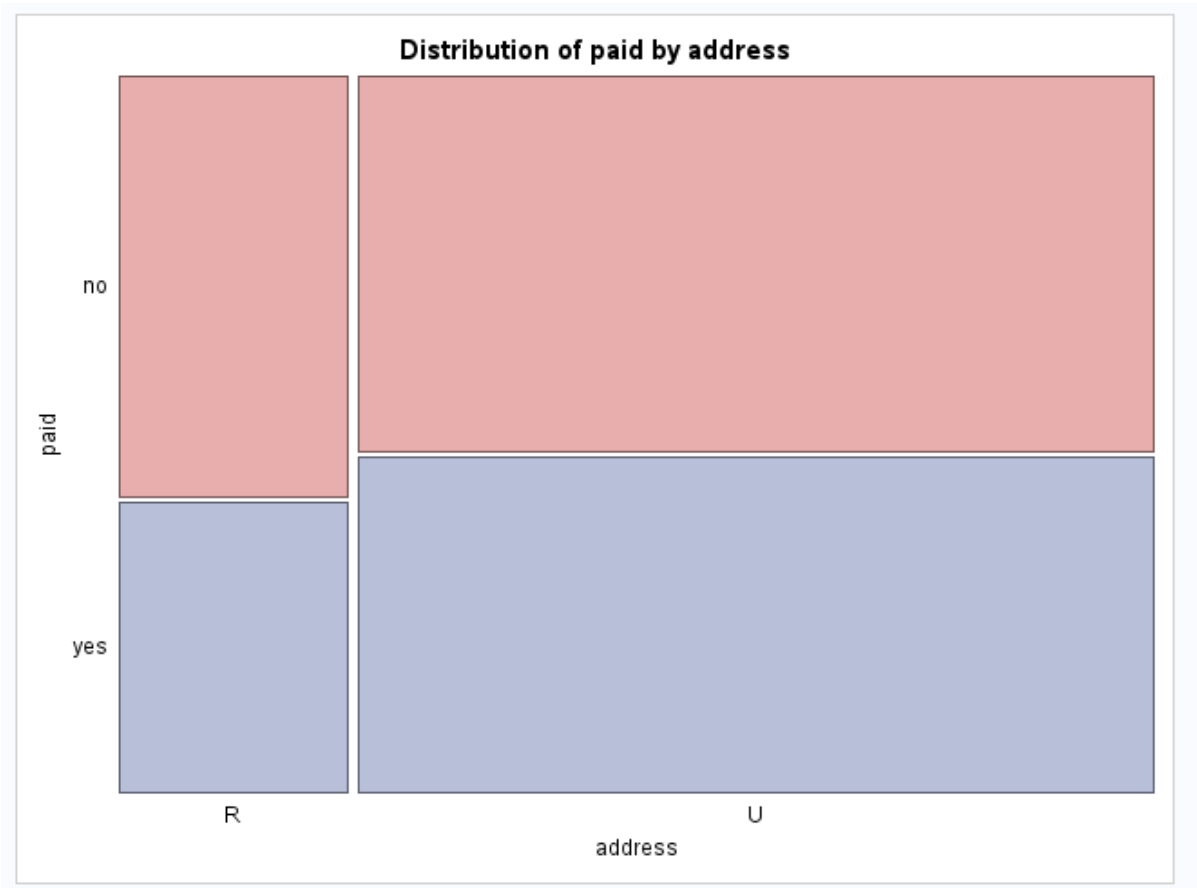



Figure 6.18: SAS: Mosaic plot for student performance data

| Statistic | Value | ASE |
|---|---|---|
| Gamma | 0.8323 | 0.0269 |
| Kendall's Tau-b | 0.5796 | 0.0281 |
| Stuart's Tau-c | 0.4288 | 0.0290 |
| Somers' D C\|R | 0.7293 | 0.0300 |
| Somers' D R\|C | 0.4605 | 0.0297 |
| Pearson Correlation | 0.6475 | 0.0308 |
| Spearman Correlation | 0.6399 | 0.0306 |
| Lambda Asymmetric C\|R | 0.2131 | 0.0267 |
| Lambda Asymmetric R\|C | 0.1008 | 0.0563 |
| Lambda Symmetric | 0.1763 | 0.0259 |
| Uncertainty Coefficient C\|R | 0.2009 | 0.0210 |
| Uncertainty Coefficient R\|C | 0.3227 | 0.0277 |
| Uncertainty Coefficient Symmetric | 0.2476 | 0.0237 |

Figure 6.19: SAS: Categorical variables' association, Walc by Dalc

## 6.9 References

## 6.10 Website References

1. SAS account sign-up Use this link to sign up for a SAS account.
2. SAS Studio link Once you have activated your account, use this link to login to your SAS studio online.
3. SAS Studio Help This link contains help on SAS studio features and commands.

# 7 Two-sample Hypothesis Tests

## 7.1 Introduction

In this topic, we introduce the routines for a common class of hypothesis tests: the scenario of comparing the location parameter of two groups. This technique is commonly used in A/B testing to assess if an intervention has resulted in a significant difference between two groups.

Hypothesis tests are routinely abused in many ways by investigators. Such tests typically require strong assumptions to hold, and can result in false positives or negatives. As such, personally, I prefer to use confidence intervals to make an assessment of the significance of a result. However, in this topic, we introduce how the $p$-values can be obtained for these hypothesis tests.

## 7.2 Procedure for Significance Tests

As a recap, here is the general approach for conducting a hypothesis test:

### 7.2.1 Step 1: Assumptions

In this step, we verify that the assumptions required for the test are valid. In some tests, this step is carried out at the end of the others, but it is always essential to perform. Some tests are very sensitive to the assumptions - this is the main reason that the class of robust statistics was invented.

### 7.2.2 Step 2: State the hypotheses and significance level

The purpose of hypothesis testing is to make an inferential statement about the population from which the data arose. This inferential statement is what we refer to as the hypothesis regarding the population.

> **Note**
>
> A hypothesis is a statement about *population*, usually claiming that a parameter takes a particular numerical value or falls in a certain range of values.

The hypotheses will be stated as a pair: The first hypothesis is the null hypothesis $H_0$ and the second is the alternative hypothesis $H_1$. Both statements will involve the **population parameter** (not the data summary) of interest. For example, if we have a sample of observations from two groups $A$ and $B$, and we wish to assess if the mean of the populations is different, the hypotheses would be

$$
\begin{aligned}
H_0: &\quad \mu_A = \mu_B \\
H_1: &\quad \mu_A \neq \mu_B
\end{aligned}
$$

$H_0$ is usually a statement that indicates "no difference", and $H_1$ is *usually* the complement of $H_0$.

At this stage, it is also crucial to state the significance level of the test. The significance level corresponds to the Type I error of the test - the probability of rejecting $H_0$ when in fact it was true. This level is usually denoted as $\alpha$, and is usually taken to be 5%, but there is no reason to adopt this blindly. Think of the choice of 5% as corresponding to accepting an error rate of 1 in 20 - that's how it was originally decided upon by Fisher. Where possible, the significance level should be chosen to be appropriate for the problem at hand.

> ⚠ Warning
>
> It is important to state the significance level at this stage, because if it is chosen *after inspecting the data*, the test is no longer valid. This is because, after knowing the $p$-value, one could always choose the significance level such that it yields the *desired* decision (reject or not).

It is also possible to test *one-sided* alternatives. In such scenarios, the hypotheses would be of the form:

$$
\begin{aligned}
H_0: &\quad \mu_A = \mu_B \\
H_1: &\quad \mu_A > \mu_B
\end{aligned}
$$

Such a test is known as a one-tailed test.

### 7.2.3 Step 3: Compute the test statistic

The test statistic is usually a measure of how far the observed data deviates from the scenario defined by $H_0$. Usually, the larger it is, the more evidence we have against $H_0$.

The construction of a hypothesis test involves the derivation of the exact or approximate distribution of the test statistic under $H_0$. Deviations from the assumption could render this distribution incorrect.

### 7.2.4 Step 4: Compute the $p$-value

The $p$-value quantifies the chance of observing such a test statistic, or one that is more extreme in the direction of $H_1$, under $H_0$. The distribution of the test statistic under $H_0$ is used to compute this value between 0 and 1. A value closer to 0 indicates stronger evidence against $H_0$.

In the case of a one-tailed test, try to understand the behaviour of the test statistic, and identify the *signal* that would yield more evidence supporting the alternative hypothesis.

### 7.2.5 Step 5: State your conclusion

This is the binary decision stage. If the $p$-value is less than the stated significance level, we conclude that we reject $H_0$. Otherwise, we say that we do not reject $H_0$. It is conventional to use this terminology (instead of saying "accept $H_1$") since our $p$-value is obtained with respect to $H_0$.

## 7.3 Confidence Intervals

Confidence intervals are an alternative method of inference for population parameters. Instead of yielding a binary reject/do-not-reject result, they return a confidence interval that contains the plausible values for the population parameter. Many confidence intervals are derived by inverting hypothesis tests, and almost all confidence intervals are of the form

$$\text{Sample estimate } \pm \text{ margin of error}$$

For instance, if we observe $x_1, \ldots, x_n$ from a Normal distribution, and wish to estimate the mean of the distribution, the 95% confidence interval based on the the $t$ distribution is

$$\bar{x} \pm t_{0.025,n-1} \times \frac{s}{\sqrt{n}}$$

where

- $s$ is the sample standard deviation, and
- $t_{0.025,n-1}$ is the 0.025-quantile from the $t$ distribution with $n-1$ degrees of freedom.

The formulas for many confidence intervals rely on asymptotic Normality of the estimator. However, this is an assumption that can be overcome with the technique of bootstrapping. We shall touch on this in the final topic of our course, in Section 10.6.2.

Bootstrapping can also be used to sidestep the distributional assumptions in hypothesis tests, but I still much prefer confidence intervals to tests because they yield an interval; they provide much more information than a binary outcome.

## 7.4 Parametric Tests

Parametric tests are hypothesis tests that assume some form of distribution for the sample (or population) to follow. An example of such a test is the $t$-test, which assumes that the data originate from a Normal distribution.

Conversely, nonparametric tests are hypothesis tests that do not assume any form of distribution for the sample. It seems we should always use non-parametric tests since distributional assumptions would not be violated, right? Unfortunately, since nonparametric tests are so general, they do not have a high discriminative ability - we say that they have low power. In other words, if a dataset truly comes from a Normal distribution, using the $t$-test would be able to detect smaller differences between the groups better than a non-parametric test.

In this section, we cover parametric tests for comparing the difference in mean between **two** groups.

### 7.4.1 Independent Samples Test

In an independent samples $t$-test, observations in one group yield *no information* about the observations in the other group. Independent samples can arise in a few ways:

- In an experimental study, study units could be assigned randomly to different treatments, thus forming the two groups.
- In an observational study, we could draw a random sample from the population, and then record an explanatory categorical variable on each unit, such as the gender or senior-citizen status.
- In an observational study, we could draw a random sample from a group (say smokers), and then a random sample from another group (say non-smokers). This would result in a situation where the independent 2-sample $t$-test is appropriate.

#### Formal Set-up

Formally speaking, this is how the independent 2-sample t-test works:

Suppose that $X_1, X_2, \dots, X_{n_1}$ are independent observations from group 1, and $Y_1, \dots Y_{n_2}$ are independent observations from group 2. It is assumed that

$$X_i \sim N(\mu_1, \sigma^2),\ i = 1, \dots, n_1 \tag{7.1}$$
$$Y_j \sim N(\mu_2, \sigma^2),\ j = 1, \dots, n_2 \tag{7.2}$$

The null and alternative hypotheses would be

$$\begin{aligned} H_0 : &\quad \mu_1 = \mu_2 \\ H_1 : &\quad \mu_1 \neq \mu_2 \end{aligned}$$

The test statistic for this test is:

$$T_1 = \frac{(\bar{X} - \bar{Y}) - 0}{s_p \sqrt{1/n_1 + 1/n_2}}$$

where

$$s_p^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$$

Under $H_0$, the test statistic $T_1 \sim t_{n_1+n_2-2}$. When we use a software to apply the test above, it will typically also return a confidence interval, computed as

$$(\bar{X} - \bar{Y}) \pm t_{n_1+n_2-2,1-\alpha/2} \times s_p \sqrt{1/n_1 + 1/n_2}$$

The set-up above corresponds to the case where the variance within each group is assumed to be the same. We use information from both groups to estimate the common variance. If we

find evidence in the data to the contrary, we used the *unpooled* variance in the denominator of $T_1$:

$$T_{1,unpooled} = \frac{(\bar{X} - \bar{Y}) - 0}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}$$

where $s_1^2$ and $s_2^2$ are the sample variance from groups 1 and 2. The test statistic still follows a $t$ distribution, but the degrees of freedom are approximated. This approximation is known as the Satterthwaite approximation.

**Example 7.1** (Abalone Measurements)**.** The dataset on abalone measurements from the UCI machine learning repository contains measurements of physical characteristics, along with the gender status. We derive a sample of 50 measurements of male and female abalone records for use here. Our goal is to study if there is a significant difference between the viscera weight[1] between males and females. The derived dataset can be found on Canvas.

### R code

```
abl <- read.csv("data/abalone_sub.csv")
x <- abl$viscera[abl$gender == "M"]
y <- abl$viscera[abl$gender == "F"]

t.test(x, y, var.equal=TRUE)
```

```
    Two Sample t-test

data:  x and y
t = 0.91008, df = 98, p-value = 0.365
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.02336287  0.06294287
sample estimates:
mean of x mean of y
  0.30220   0.28241
```

### Python code

```
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm

abl = pd.read_csv("data/abalone_sub.csv")
#abl.head()
```

[1]Viscera is the gut weight after bleeding out the abalone (in grams).

```
#abalone_df.describe()

x = abl.viscera[abl.gender == "M"]
y = abl.viscera[abl.gender == "F"]

t_out = stats.ttest_ind(x, y)
ci_95 = t_out.confidence_interval()

print(f"""
* The p-value for the test is {t_out.pvalue:.3f}.
* The actual value of the test statistic is {t_out.statistic:.3f}.
* The upper and lower limits of the CI are ({ci_95[0]:.3f}, {ci_95[1]:.3f}).
""")
```

```
* The p-value for the test is 0.365.
* The actual value of the test statistic is 0.910.
* The upper and lower limits of the CI are (-0.023, 0.063).
```

### SAS Output

When we run the independent samples t-test on SAS, we should observe the output in Figure 7.1.

Variable: viscera

| gender | Method | N | Mean | Std Dev | Std Err | Minimum | Maximum |
|---|---|---|---|---|---|---|---|
| F | | 50 | 0.2824 | 0.1087 | 0.0154 | 0.0950 | 0.5750 |
| M | | 50 | 0.3022 | 0.1087 | 0.0154 | 0.0400 | 0.6380 |
| Diff (1-2) | Pooled | | -0.0198 | 0.1087 | 0.0217 | | |
| Diff (1-2) | Satterthwaite | | -0.0198 | | 0.0217 | | |

| gender | Method | Mean | 95% CL Mean | | Std Dev | 95% CL Std Dev | |
|---|---|---|---|---|---|---|---|
| F | | 0.2824 | 0.2515 | 0.3133 | 0.1087 | 0.0908 | 0.1355 |
| M | | 0.3022 | 0.2713 | 0.3331 | 0.1087 | 0.0908 | 0.1355 |
| Diff (1-2) | Pooled | -0.0198 | -0.0629 | 0.0234 | 0.1087 | 0.0954 | 0.1264 |
| Diff (1-2) | Satterthwaite | -0.0198 | -0.0629 | 0.0234 | | | |

| Method | Variances | DF | t Value | Pr > \|t\| |
|---|---|---|---|---|
| Pooled | Equal | 98 | -0.91 | 0.3650 |
| Satterthwaite | Unequal | 98 | -0.91 | 0.3650 |

| Equality of Variances | | | | |
|---|---|---|---|---|
| Method | Num DF | Den DF | F Value | Pr > F |
| Folded F | 49 | 49 | 1.00 | 0.9980 |

Figure 7.1: SAS: Independent samples t-test, Abalone data

The assumptions of the test include Normality and equal variance. Let us assess the equality of variance first. While there are many hypothesis tests specifically for assessing if variances

are equal (e.g. Levene, Bartlett), in our class, I advocate a simple rule of thumb. If the larger s.d is more than twice the smaller one, than we should not use the equal variance form of the test. This rule of thumb is widely used in practice (see Section 7.7.1).

### R code

```
aggregate(viscera ~ gender, data=abl, sd)
```

```
  gender   viscera
1      F 0.1087070
2      M 0.1087461
```

### Python code

```
abl.groupby('gender').describe()
```

```
       viscera
         count     mean       std    min       25%    50%       75%    max
gender
F         50.0  0.28241  0.108707  0.095  0.201250  0.275  0.365125  0.575
M         50.0  0.30220  0.108746  0.040  0.253125  0.310  0.348750  0.638
```

To assess the Normality assumption, we make also histograms and qq-plots (see Figure 7.2, Figure 7.3, and
Figure 7.4).

> **Note**
>
> Only the R code is shown here.

```
library(lattice)
histogram(~viscera | gender, data=abl, type="count")
qqnorm(y, main="Female Abalones");  qqline(y)
qqnorm(x, main="Male Abalones");  qqline(x)
```

We would conclude that there is no significant difference between the mean viscera weight of males and females.

Apart from qq-plots, it is worthwhile to touch on additional methods that are used to assess how much a dataset deviates from Normality.

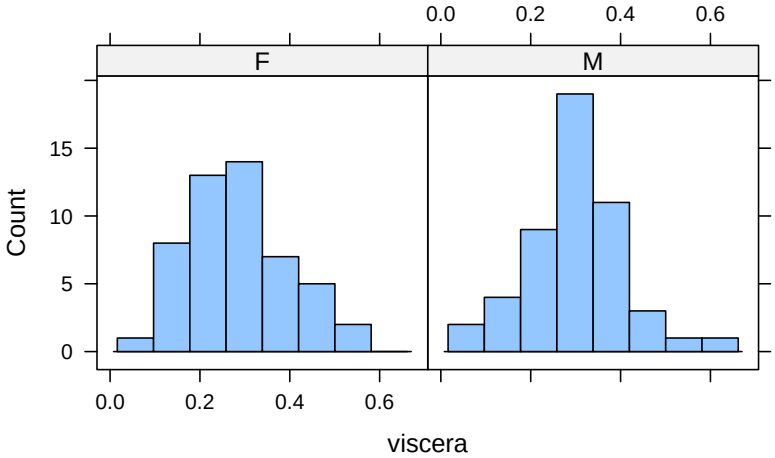



Figure 7.2: R: Histograms for males and females

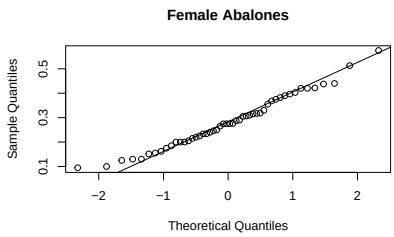



Figure 7.3: R: QQ-plot for females

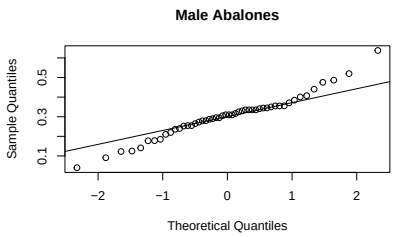



Figure 7.4: R: QQ-plot for males

### 7.4.2 More on Assessing Normality

#### Skewness

The Normal distribution is symmetric about it's mean. Hence if we observe asymmetry in our histogram, we might suspect deviation from Normality. To quantify this asymmetry, we use *skewness*. One method of estimating the skewness of a distribution from data is:

$$g_1 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^3}{[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2]^{3/2}}$$

This is the method-of-moments estimator for the distribution skewness parameter. A value close to 0 indicates low skewness (i.e. high symmetry). Positive values correspond to right-skew and negative values to left-skew.

#### Kurtosis

Kurtosis measures the thickness of the tails of a distribution. Positive kurtosis implies that the tails are "fatter" than those of a Normal. Negative values indicate that the tails are "thinner" than those of a Normal.

The method of moments estimator is

$$g_2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2]^2} - 3$$

#### Hypothesis tests for Normality

The Shapiro-Wilk test and the Kolmogorov-Smirnov test are formal hypothesis tests with the following hypotheses:

$$\begin{aligned} H_0 &: \quad \text{Data follows a Normal distribution} \\ H_1 &: \quad \text{Data does not follow a Normal distribution} \end{aligned}$$

You can read more about them in the references, but take note that applying multiple tests leads to a higher Type I error. Moreover, a large small sample size will almost always reject $H_0$ because small deviations are being classed as significant. I advocate a more graphical approach in assessing Normality, especially since the solution to Non-normality (the bootstrap) is readily accessible today.

**Example 7.2** (Abalone Measurements)**.** Let us apply the above computations to the abalone measurements.

#### R code

```r
library(DescTools)
aggregate(viscera ~ gender, data=abl, Skew, method=1)
##   gender   viscera
## 1      F 0.4060918
## 2      M 0.2482997

aggregate(viscera ~ gender, data=abl, Kurt, method=1)
##   gender    viscera
## 1      F -0.2431501
## 2      M  1.1660593

# Shapiro-Wilk Test only for males:
shapiro.test(x)
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.96779, p-value = 0.1878
```

### Python code

```python
abl.groupby("gender").skew()
##          viscera
## gender
## F       0.418761
## M       0.256046

for i,df in abl.groupby('gender'):
    print(f"{df.gender.iloc[0]}: {df.viscera.kurt():.4f}")
## F: -0.1390
## M: 1.4220

stats.shapiro(x)
## ShapiroResult(statistic=np.float64(0.9677872659314101), pvalue=np.float64(0.187849079365071
```

### SAS output

Although the skewness seems large for the female group, the Normality tests do not reject the null hypothesis (see Figure 7.5).

Variable: viscera
gender = F

| Tests for Normality | | | | |
|---|---|---|---|---|
| Test | Statistic | | p Value | |
| Shapiro-Wilk | W | 0.977835 | Pr < W | 0.4647 |
| Kolmogorov-Smirnov | D | 0.069947 | Pr > D | >0.1500 |
| Cramer-von Mises | W-Sq | 0.040268 | Pr > W-Sq | >0.2500 |
| Anderson-Darling | A-Sq | 0.276907 | Pr > A-Sq | >0.2500 |

Variable: viscera
gender = M

| Tests for Normality | | | | |
|---|---|---|---|---|
| Test | Statistic | | p Value | |
| Shapiro-Wilk | W | 0.967787 | Pr < W | 0.1878 |
| Kolmogorov-Smirnov | D | 0.13202 | Pr > D | 0.0279 |
| Cramer-von Mises | W-Sq | 0.123354 | Pr > W-Sq | 0.0532 |
| Anderson-Darling | A-Sq | 0.673459 | Pr > A-Sq | 0.0780 |

Figure 7.5: SAS: Normality tests

### 7.4.3 Paired Sample Test

The data in a paired sample test also arises from two groups, but the two groups are not independent. A very common scenario that gives rise to this test is when the same subject receives both treatments. His/her measurement under each treatment gives rise to a measurement in each group. However, the measurements are no longer independent.

**Example 7.3** (Reaction time of drivers)**.** Consider a study on 32 drivers sampled from a driving school. Each driver is put in a simulation of a driving situation, where a target flashes red and green at random periods. Whenever the driver sees red, he/she has to press a brake button.

For each driver, the study is carried out twice - at one of the repetitions, the individual carries on a phone conversation while at the other, the driver listens to the radio.

Each measurement falls under one of two groups - "phone" or "radio", but the measurements for driver $i$ are clearly related. Some people might just have a slower/faster baseline reaction time!

This is a situation where a paired sample test is appropriate, not an independent sample test.

#### Formal Set-up

Suppose that we observe $X_1, \dots, X_n$ independent observations from group 1 and $Y_1, \dots, Y_n$ independent observations from group 2. However the pair $(X_i, Y_i)$ are correlated. It is assumed that

$$X_i \sim N(\mu_1, \sigma_1^2),\ i = 1, \dots, n \tag{7.3}$$
$$Y_j \sim N(\mu_2, \sigma_2^2),\ j = 1, \dots, n \tag{7.4}$$

We let $D_i = X_i - Y_i$ for $i = 1, \dots, n$. It follows that

$$D_i \sim N(\mu_1 - \mu_2,\ \sigma_1^2 + \sigma_2^2 - 2cov(X_i, Y_i))$$

The null and alternative hypotheses are stated in terms of the distribution of $D_i$:

$$\begin{aligned} H_0 :&\quad \mu_D = 0 \\ H_1 :&\quad \mu_D \neq 0 \end{aligned}$$

The test statistic for this test is:

$$T_2 = \frac{\bar{D} - 0}{s/\sqrt{n}}$$

where

$$s^2 = \frac{\sum_{i=1}^{n}(D_i - \bar{D})^2}{(n-1)}$$

Under $H - 0$, the test statistic $T_2 \sim t_{n-1}$. When we use a software to apply the test above, it will typically also return a confidence interval, computed as

$$\bar{D} \pm t_{n-1,1-\alpha/2} \times s/\sqrt{n}$$

**Example 7.4** (Heart Rate Before/After Treadmill)**.** The following dataset comes from the textbook (Rosner 2015), where an individual recorded his heart rate before using a treadmill (baseline) and 5 minutes after use, for 12 days in 2006.

### R code

To run the paired test, we use the same function `t.test` as earlier, but we set the argument `paired`.

```
hr_df <- read.csv("data/health_promo_hr.csv")
before <- hr_df$baseline
after <- hr_df$after5
t.test(before, after, paired=TRUE)
```

```
    Paired t-test

data:  before and after
t = -21.714, df = 11, p-value = 2.209e-10
alternative hypothesis: true mean difference is not equal to 0
```

```
95 percent confidence interval:
 -16.77286 -13.68548
sample estimates:
mean difference
      -15.22917
```

### Python code

```python
hr_df = pd.read_csv("data/health_promo_hr.csv")
#hr_df.head()

paired_out = stats.ttest_rel(hr_df.baseline, hr_df.after5)
print(f"""
Test statistic: {paired_out.statistic:.3f}.
p-val: {paired_out.pvalue:.3f}.""")
```

```
Test statistic: -21.714.
p-val: 0.000.
```

### SAS Output

**Difference: baseline - after5**

| N | Mean | Std Dev | Std Err | Minimum | Maximum |
|---|---|---|---|---|---|
| 12 | -15.2292 | 2.4296 | 0.7014 | -18.5000 | -10.7500 |

| Mean | 95% CL Mean | | Std Dev | 95% CL Std Dev | |
|---|---|---|---|---|---|
| -15.2292 | -16.7729 | -13.6855 | 2.4296 | 1.7211 | 4.1252 |

| DF | t Value | Pr > \|t\| |
|---|---|---|
| 11 | -21.71 | <.0001 |

Figure 7.6: SAS: HR paired t-test

It is imperative to also make the checks for Normality. If you were to make them, you would realise that the sample size is rather small - it is difficult to make the case for Normality here.

A couple of interesting plots from SAS are shown in Figure 7.7 and Figure 7.8.

When we inspect the paired plot, we are looking for a similar gradient for each line, or at least similar in sign. If instead, we observed a combination of positive and negative gradients, then we would be less confident that there is a difference in means between the groups.

For the agreement plot, if we were to observe the points scattered around the line $y = x$, then we would be more inclined to believe that the mean difference is indeed 0.

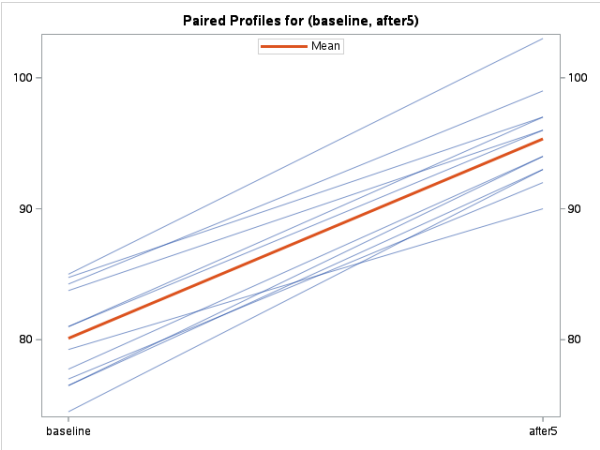



Figure 7.7: SAS: Paired profiles

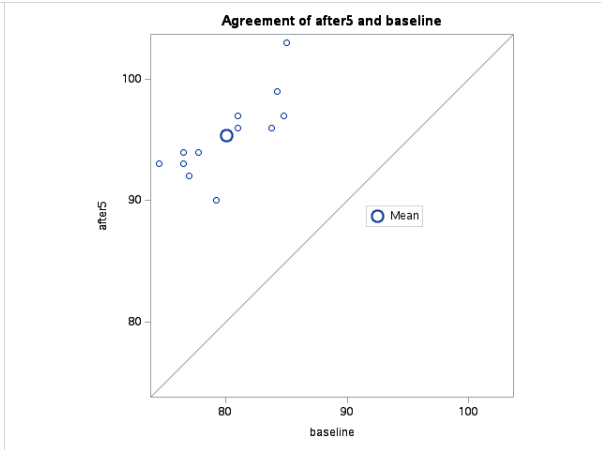



Figure 7.8: SAS: Agreement plot

## 7.5 Non-parametric Tests

If the distributional assumptions of the $t$-test are not met, we can take action in several ways. Historically, one method was to transform the data (if it was skewed) to make the histogram symmetric and thus closer to a Normal. But this was not ideal - it did not help in cases where the data was symmetric but had fatter tails than the Normal. As time progressed, statisticians invented tools to overcome the distributional assumptions. One sub-field was robust statistics, which keep the assumptions to a minimum, e.g. only requiring the underlying distribution to be symmetric. Another sub-field was the area of non-parametric statistics, where almost **no** distributional assumptions are made about the data.

In this section, we cover the non-parametric analogues for independent and paired 2-sample tests.

### 7.5.1 Independent Samples Test

The non-parametric analogue of the independent 2-sample test is the Wilcoxon Rank Sum (WRS) test (equivalent to the Mann-Whitney test).

#### Formal Set-up

Suppose that we observe $X_1, \dots, X_{n_1}$ independent observations from group 1 (with distribution $F$) and $Y_1, \dots, Y_{n_2}$ independent observations from group 2 (with distribution $G$).

The hypotheses associated with this test are:

$$\begin{aligned} H_0 &: F = G \\ H_1 &: F(x) = G(x - \Delta), \Delta \neq 0 \end{aligned}$$

In other words, the alternative hypothesis is that the distribution of group 1 is a location shift of the distribution of group 2.

The WRS test begins by pooling the $n_1 + n_2$ data points and ranking them. The smallest observation is awarded rank 1, and the largest observation is awarded rank $n_1 + n_2$, assuming there are no tied values. If there are tied values, the observations with the same value receive an averaged rank.

Compute $R_1$, the sum of the ranks in group 1. If this sum is large, it means that most of the values in group 1 were larger than those in group 2. Note that the average rank in the combined sample is

$$\frac{n_1 + n_2 + 1}{2}$$

Under $H_0$, the expected rank sum of group 1 is

$$E(R_1) = n_1 \times \frac{n_1 + n_2 + 1}{2}$$

The test statistic is a comparison of $R_1$ with the above expected value:

$$W_1 = \begin{cases} \frac{\left|R_1 - \frac{n_1(n_1+n_2+1)}{2}\right| - \frac{1}{2}}{\sqrt{n_1 n_2 (n_1+n_2+1)/12}}, & R_1 \neq \frac{n_1(n_1+n_2+1)}{2} \text{ and no ties} \\ \frac{\left|R_1 - \frac{n_1(n_1+n_2+1)}{2}\right| - \frac{1}{2}}{\sqrt{n_1 n_2 \left(n_1+n_2+1 - \frac{\sum_{i=1}^{g} t_i(t_i^2-1)}{(n_1+n_2)(n_1+n_2-1)}\right)/12}}, & R_1 \neq \frac{n_1(n_1+n_2+1)}{2} \text{ and ties present} \\ 0, & R_1 = \frac{n_1(n_1+n_2+1)}{2} \end{cases}$$

where $g$ refers to the number of groups with ties, and $t_i$ refers to the number of tied values in each group.

The test above should only be used if both $n_1$ and $n_2$ are at least 10, and if the observations (not the ranks) come from an underlying continuous distribution. If these assumptions hold, then the test statistic $W_1$ follows a $N(0, 1)$ distribution.

**Example 7.5** (Abalone Measurements)**. R code**

We can perform the WRS test in R:

```
wilcox.test(x, y)
```

```
    Wilcoxon rank sum test with continuity correction

data:  x and y
W = 1415.5, p-value = 0.2553
alternative hypothesis: true location shift is not equal to 0
```

**Python code**

As mentioned, the Mann-Whitney test will return the same $p$-value as the WRS test.

```python
wrs_out = stats.mannwhitneyu(x, y)

print(f"""Test statistic: {wrs_out.statistic:.3f}.
p-val: {wrs_out.pvalue:.3f}.""")
```

```
Test statistic: 1415.500.
p-val: 0.255.
```

### SAS Output

The SAS output can be seen in Figure 7.9.

| Wilcoxon Scores (Rank Sums) for Variable viscera Classified by Variable gender | | | | | |
|---|---|---|---|---|---|
| gender | N | Sum of Scores | Expected Under H0 | Std Dev Under H0 | Mean Score |
| F | 50 | 2359.50 | 2525.0 | 145.039615 | 47.190 |
| M | 50 | 2690.50 | 2525.0 | 145.039615 | 53.810 |
| Average scores were used for ties. | | | | | |

| Wilcoxon Two-Sample Test | | | | | |
|---|---|---|---|---|---|
| | | | | t Approximation | |
| Statistic | Z | Pr < Z | Pr > \|Z\| | Pr < Z | Pr > \|Z\| |
| 2359.500 | -1.1376 | 0.1276 | 0.2553 | 0.1290 | 0.2580 |
| Z includes a continuity correction of 0.5. | | | | | |

| Kruskal-Wallis Test | | |
|---|---|---|
| Chi-Square | DF | Pr > ChiSq |
| 1.3020 | 1 | 0.2538 |

Figure 7.9: SAS: Wilcoxon rank sum output

Since we know the number of observations in each group to be more than 10, the approximation holds. Comparing with Example 7.1, observe that we have a similar conclusion.

Notice that the test statistic appears different in SAS. However, it is simply a matter of a location shift of the test statistic. R and Python subtract the smallest possible sum of ranks from group 1, but SAS does not. In our example, the group size is 50. Hence we can recover the SAS test statistic from the R test statistic with

$$1415.5 + \frac{50(50+1)}{2} = 2690.5$$

The $p$-value is identical in all three software.

### 7.5.2 Paired Samples Test

The analogue of the paired sample $t$-test is known as the Wilcoxon Sign Test (WST).

#### Formal Set-up

Again, suppose that we observe $X_1, \dots, X_n$ observations from group 1 and $Y_1, \dots, Y_n$ observations from group 2. Groups 1 and 2 are paired (or correlated) in some way.

Once again, we compute $D_i = X_i - Y_i$. The null hypothesis is that

$$\begin{aligned} H_0 &: \text{ median of } D_i \text{ is } 0. \\ H_1 &: \text{ median of } D_i \text{ is not } 0. \end{aligned}$$

We begin by ranking the $|D_i|$. Ignoring pairs for which $D_i = 0$, we rank the remaining observations from 1 for the pair with the smallest absolute value, up to $n$ for the pair with the largest absolute value (assuming no ties).

We then compute $R_2$, the sum of ranks for the positive $D_i$. If this sum is large, we expect that the pairs with $X_i > Y_i$ have a larger difference (in absolute values) than those with $X_i < Y_i$. Under $H_0$, it can be shown that

$$E(R_2) = m(m+1)/4$$

where $m$ is the number of of non-zero differences.

Thus the test statistic is a comparison of $R_2$ with the above expected value:

$$W_2 = \begin{cases} \frac{\left|R_2 - \frac{n(n+1)}{4}\right| - \frac{1}{2}}{\sqrt{n(n+1)(2n+1)/24}}, & R_2 \neq \frac{n(n+1)}{4} \text{ and no ties} \\ \frac{\left|R_2 - \frac{n(n+1)}{4}\right| - \frac{1}{2}}{\sqrt{n(n+1)(2n+1)/(24 - \sum_{i=1}^{g}(t_i^3 - t_i)/48)}}, & R_2 \neq \frac{n(n+1)}{4} \text{ and ties present} \\ 0, & R_2 = \frac{n(n+1)}{4} \end{cases}$$

where $g$ denotes the number of tied groups, and $t_i$ refers to the number of differences with the same absolute value in the $i$-th tied group.

If the number of non-zero $D_i$'s is at least 16, then the test statistic $W_2$ follows a $N(0,1)$ distribution approximately.

**Example 7.6** (Heart Rate Before/After Treadmill)**. R code**

We can perform the Wilcoxon Signed Rank test in R:

```
wilcox.test(before, after, paired = TRUE, exact = FALSE)
```

```
    Wilcoxon signed rank test with continuity correction

data:  before and after
V = 0, p-value = 0.002507
alternative hypothesis: true location shift is not equal to 0
```

### Python code

```python
wsr_out = stats.wilcoxon(hr_df.baseline, hr_df.after5,
                         correction=True, method='approx')
print(f"""Test statistic: {wsr_out.statistic:.3f}.
p-val: {wsr_out.pvalue:.3f}.""")
```

```
Test statistic: 0.000.
p-val: 0.003.
```

### SAS Output

**Variable: _Difference_ (Difference: baseline - after5)**

| Tests for Location: Mu0=0 | | | | |
|---|---|---|---|---|
| **Test** | **Statistic** | | **p Value** | |
| Student's t | t | -21.7136 | Pr > \|t\| | <.0001 |
| Sign | M | -6 | Pr >= \|M\| | 0.0005 |
| Signed Rank | S | -39 | Pr >= \|S\| | 0.0005 |

Figure 7.10: SAS: Signed rank test output

In this problem, we do not have 16 non-zero $D_i$'s. Hence, we should in fact be using the "exact" version of the test (R and Python). However, the exact version of the test cannot be used when there are ties.

SAS does indeed use the "exact" version of the test, so that accounts for the difference in $p$-values. The test statistic in SAS is computed by subtracting $E(R_2)$, but in R and Python this subtraction is not done. To get from test statistic in R (0) to the one in SAS (-39):

$$0 - \frac{12(12+1)}{4} = -39$$

In cases like these, we can turn to the bootstrap, or we can use a permutation test. We shall revisit these in our final topic Section 10.1.

## 7.6 Summary

While the test statistics are shown in detail, we do not need the full details for our class. I should also point out that the definitions of the test statistic in Section 7.5.1 and Section 7.5.2 were both taken from Rosner (2015). However, take note that different software have slightly different implementations, for instance in how they deal with ties. As always, my advice is to read the documentation as much as possible, and then to use the output of the tests as a guide to your decision-making (instead of treating them as the absolute truth).

## 7.7 References

### 7.7.1 Website References

1. UCI full abalone dataset: The dataset in the examples above, starting from Example 7.1, consists of samples from the full dataset.
2. Inference recap from Penn State:
   - Hypothesis testing recap
   - Confidence intervals recap
3. Tests for Normality More information on the Kolmogorov-Smirnov and Shapiro-Wilks Tests for Normality.
4. Overview of $t$-tests This page includes the rule of thumb about deciding when to use the equal variance test, and when to use the unequal variances version.
5. SAS vs. R/Python This link provides an explanation why the WRS test statistic for SAS is different from R in Example 7.5.

## 7.8 Exercises

1. List the different methods of assessing Normality.
2. Compute skewness for all variables in the concrete data set.
3. Compute $R_1$, the test statistic for the Wilcoxon Rank Sum Test, using R and Python.
4. Compute kurtosis for the `chem` data from chapter 5. Remove the outliers and recompute.
5. Perform the 2-sample t-test for `G3` scores, comparing scores for `Medu=1` to `Medu=4`.

# 8 ANOVA

## 8.1 Introduction

In the previous topic, we learned how to run two-sample $t$-tests. The objective of these procedures is to compare the means from two groups. Frequently, however, the means of more than two groups need to be compared.

In this topic, we introduce the *one-way analysis of variance* (ANOVA), which generalises the $t$-test methodology to more than 2 groups. Hypothesis tests in the ANOVA framework require the assumption of Normality. When this does not hold, we turn to the Kruskal-Wallis test - the non-parametric version, to compare distributions between groups.

While the $F$-test in ANOVA provides a determination of whether or not the group means are different, in practice, we would always want to follow up with specific comparisons between groups as well. This topic covers how we can construct confidence intervals in those cases as well.

**Example 8.1** (Effect of Antibiotics)**.** The following example was taken from Ekstrøm and Sørensen (2015). An experiment with dung from heifers[1] was carried out in order to explore the influence of antibiotics on the decomposition of dung organic material. As part of the experiment, 36 heifers were randomly assigned into six groups.

Antibiotics of different types were added to the feed for heifers in five of the groups. The remaining group served as a control group. For each heifer, a bag of dung was dug into the soil, and after 8 weeks the amount of organic material was measured for each bag.

Figure 8.1 contains a boxplot of the data from each group.

Compared to the control group, it does appear that the median organic weight of the dung from the other heifer groups is higher. The following table displays the mean, standard deviation, and count from each group:

```
      type  mean    sd count
1  Control 2.603 0.119     6
2  Alfacyp 2.895 0.117     6
3 Enroflox 2.710 0.162     6
4 Fenbenda 2.833 0.124     6
5 Ivermect 3.002 0.109     6
6 Spiramyc 2.855 0.054     4
```

Observe that the Spiramycin group only yielded 4 readings instead of 6. Our goal in this topic is to apply a technique for assessing if group means are statistically different from one another. Here are the specific analyses that we shall carry out:

[1] A heifer is a young, female cow that has not had her first calf yet.

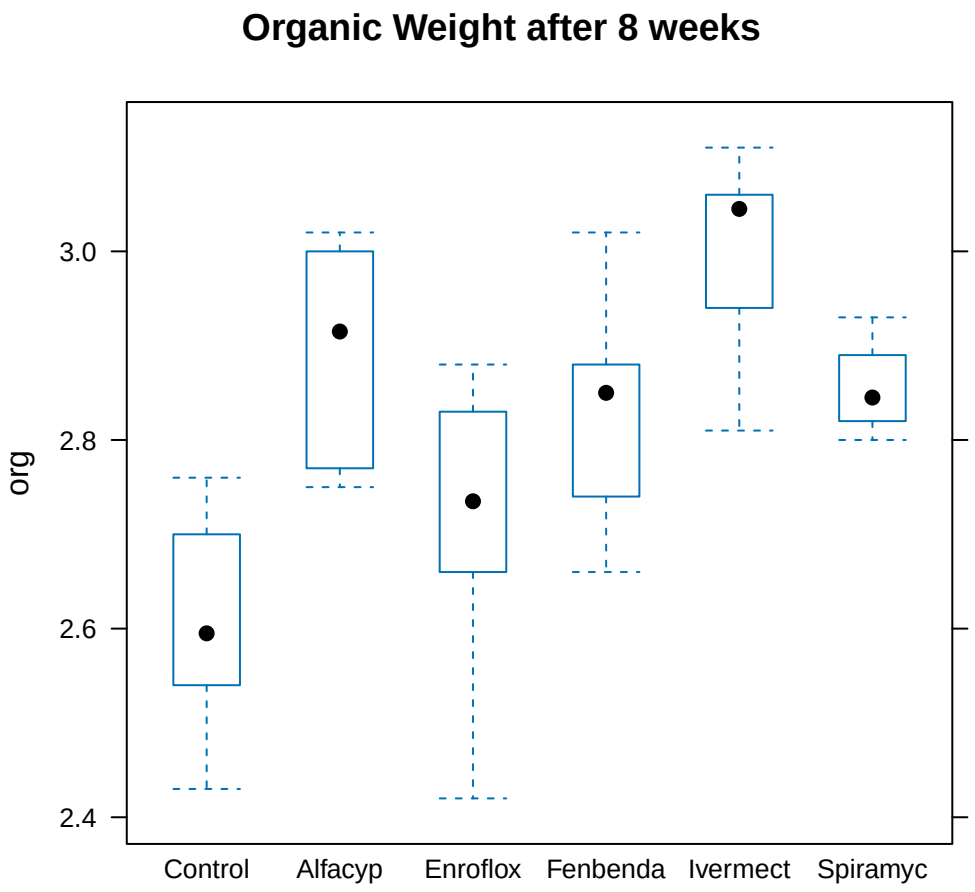

Organic Weight after 8 weeks
org
3.0
2.8
2.6
2.4
Control
Alfacyp
Enroflox
Fenbenda
Ivermect
Spiramyc

Figure 8.1

1. Is there any significant difference, at 5% level, between the mean decomposition level of the groups?
2. At 5% level, is the mean level for Enrofloxacin different from the control group?
3. Pharmacologically speaking, Ivermectin and Fenbendazole are similar to each other. Let us call this sub-group (A). They work differently than Enrofloxacin. At 5% level, is there a significant difference between the mean from sub-group A and Enrofloxacin?

## 8.2 One-Way Analysis of Variance

### 8.2.1 Formal Set-up

Suppose there are $k$ groups with $n_i$ observations in the $i$-th group. The $j$-th observation in the $i$-th group will be denoted by $Y_{ij}$. In the One-Way ANOVA, we assume the following model:

$$Y_{ij} = \mu + \alpha_i + e_{ij},\ i = 1, \dots, k,\ j = 1, \dots, n_i \tag{8.1}$$

- $\mu$ is a constant, representing the underlying mean of all groups taken together.
- $\alpha_i$ is a constant specific to the $i$-th group. It represents the difference between the mean of the $i$-th group and the overall mean.
- $e_{ij}$ represents random error about the mean $\mu + \alpha_i$ for an individual observation from the $i$-th group.

In terms of distributions, we assume that the $e_{ij}$ are i.i.d from a Normal distribution with mean 0 and variance $\sigma^2$. This leads to the model for each observation:

$$Y_{ij} \sim N(\mu + \alpha_i,\ \sigma^2) \tag{8.2}$$

It is not possible to estimate both $\mu$ and all the $k$ different $\alpha_i$'s, since we only have $k$ observed mean values for the $k$ groups. For identifiability purposes, we need to constrain the parameters. There are two common constraints used, and note that different software have different defaults:

1. Setting $\sum_{i=1}^{k} \alpha_i = 0$, or
2. Setting $\alpha_1 = 0$.

Continuing on from Equation 8.2, let us denote the mean for the $i$-th group as $\overline{Y_i}$, and the overall mean of all observations as $\overline{\overline{Y}}$. We can then write the deviation of an individual observation from the overall mean as:

$$Y_{ij} - \overline{\overline{Y}} = \underbrace{(Y_{ij} - \overline{Y_i})}_{\text{within}} + \underbrace{(\overline{Y_i} - \overline{\overline{Y}})}_{\text{between}} \tag{8.3}$$

The first term on the right of the above equation is the source of *within-group variability*. The second term on the right gives rise to *between-group variability*. The intuition behind the ANOVA procedure is that if the between-group variability is large and the within-group variability is small, then we have evidence that the group means are different.

If we square both sides of Equation 8.3 and sum over all observations, we arrive at the following equation; the essence of ANOVA:

$$\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(Y_{ij}-\overline{\overline{Y}}\right)^2=\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(Y_{ij}-\overline{Y_i}\right)^2+\sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(\overline{Y_i}-\overline{\overline{Y}}\right)^2$$

The squared sums above are referred to as:

$$SS_T = SS_W + SS_B$$

- $SS_T$: Sum of Squares Total,
- $SS_W$: Sum of Squares Within, and
- $SS_B$: Sum of Squares Between.

In addition the following definitions are important for understanding the ANOVA output:

1. The Between Mean Square:
$$MS_B = \frac{SS_B}{k-1}$$
2. The Within Mean Square:
$$MS_W = \frac{SS_W}{n-k}$$

The mean squares are estimates of the variability between and within groups. The ratio of these quantities is the test statistic.

### 8.2.2 $F$-Test in One-Way ANOVA

The null and alternative hypotheses are:

$$\begin{aligned} H_0 &: \alpha_i = 0 \text{ for all } i \\ H_1 &: \alpha_i \neq 0 \text{ for at least one } i \end{aligned}$$

The test statistic is given by

$$F = \frac{MS_B}{MS_W}$$

Under $H_0$, the statistic $F$ follows an $F$ distribution with $k-1$ and $n-k$ degrees of freedom.

### 8.2.3 Assumptions

These are the assumptions that will need to be validated.

1. The observations are independent of each other. This is usually a characteristic of the design of the experiment, and is not something we can always check from the data.
2. The errors are Normally distributed. Residuals can be calculated as follows:
$$Y_{ij} - \overline{Y_i}$$
The distribution of these residuals should be checked for Normality.

3. The variance within each group is the same. In ANOVA, the $MS_W$ is a pooled estimate (across the groups) that is used; in order for this to be valid, the variance within each group should be identical. As in the 2-sample situation, we shall avoid separate hypotheses tests and proceed with the rule-of-thumb that if the ratio of the largest to smallest standard deviation is less than 2, we can proceed with the analysis.

**Example 8.2** (F-test)**.** We being by applying the overall $F$-test to the heifers data, to assess if there is any significant difference between the means.

### R code

```
#R
heifers <- read.csv("data/antibio.csv")
u_levels <- sort(unique(heifers$type))
heifers$type <- factor(heifers$type,
                       levels=u_levels[c(2, 1, 3, 4, 5, 6)])
heifers_lm <- lm(org ~ type, data=heifers)
anova(heifers_lm)
```

```
Analysis of Variance Table

Response: org
          Df  Sum Sq  Mean Sq F value    Pr(>F)
type       5 0.59082 0.118165  7.9726 8.953e-05 ***
Residuals 28 0.41500 0.014821
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Python code

```
#Python
import pandas as pd
import numpy as np
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols

heifers = pd.read_csv("data/antibio.csv")
heifer_lm = ols('org ~ type', data=heifers).fit()
anova_tab = sm.stats.anova_lm(heifer_lm, type=3,)
print(anova_tab)
```

```
            df    sum_sq   mean_sq         F   PR(>F)
type       5.0  0.590824  0.118165  7.972558  0.00009
Residual  28.0  0.415000  0.014821       NaN      NaN
```

### SAS output

Dependent Variable: org

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 5 | 0.59082353 | 0.11816471 | 7.97 | <.0001 |
| Error | 28 | 0.41500000 | 0.01482143 | | |
| Corrected Total | 33 | 1.00582353 | | | |

At the 5% significance level, we reject the null hypothesis to conclude that the group means are significantly different from one another. This answers question (1) from Example 8.1.

To extract the estimated parameters, we can use the following code:

### R code

```
# R
summary(heifers_lm)
```

```
Call:
lm(formula = org ~ type, data = heifers)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.29000 -0.06000  0.01833  0.07250  0.18667 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.60333    0.04970  52.379  < 2e-16 ***
typeAlfacyp   0.29167    0.07029   4.150 0.000281 ***
typeEnroflox  0.10667    0.07029   1.518 0.140338    
typeFenbenda  0.23000    0.07029   3.272 0.002834 ** 
typeIvermect  0.39833    0.07029   5.667  4.5e-06 ***
typeSpiramyc  0.25167    0.07858   3.202 0.003384 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1217 on 28 degrees of freedom
Multiple R-squared:  0.5874,	Adjusted R-squared:  0.5137 
F-statistic: 7.973 on 5 and 28 DF,  p-value: 8.953e-05
```

### Python code

```
# Python
print(heifer_lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    org   R-squared:                       0.587
Model:                            OLS   Adj. R-squared:                  0.514
Method:                 Least Squares   F-statistic:                     7.973
Date:                Sun, 14 Dec 2025   Prob (F-statistic):           8.95e-05
Time:                        07:44:43   Log-Likelihood:                 26.655
No. Observations:                  34   AIC:                            -41.31
Df Residuals:                      28   BIC:                            -32.15
Df Model:                           5
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Intercept             2.8950      0.050     58.248      0.000       2.793       2.997
type[T.Control]      -0.2917      0.070     -4.150      0.000      -0.436      -0.148
type[T.Enroflox]     -0.1850      0.070     -2.632      0.014      -0.329      -0.041
type[T.Fenbenda]     -0.0617      0.070     -0.877      0.388      -0.206       0.082
type[T.Ivermect]      0.1067      0.070      1.518      0.140      -0.037       0.251
type[T.Spiramyc]     -0.0400      0.079     -0.509      0.615      -0.201       0.121
==============================================================================
Omnibus:                        2.172   Durbin-Watson:                   2.146
Prob(Omnibus):                  0.338   Jarque-Bera (JB):                1.704
Skew:                          -0.545   Prob(JB):                        0.427
Kurtosis:                       2.876   Cond. No.                         6.71
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

**SAS output**

| Level of type | N | org | |
|---|---|---|---|
| | | Mean | Std Dev |
| Alfacyp | 6 | 2.89500000 | 0.11674759 |
| Control | 6 | 2.60333333 | 0.11877149 |
| Enroflox | 6 | 2.71000000 | 0.16198765 |
| Fenbenda | 6 | 2.83333333 | 0.12355835 |
| Ivermect | 6 | 3.00166667 | 0.10943796 |
| Spiramyc | 4 | 2.85500000 | 0.05446712 |

When estimating, both R and Python set one of the $\alpha_i$ to be equal to 0. In the case of R, it is the coefficient for `Control`, since we set it as the first level in the factor. For Python, we can tell from the output that the constraint has been placed on the coefficient for `Alfacyp` (since it is missing).

However, all estimates are group means are identical. From the R output, we can compute that the estimate of the mean for the `Alfacyp` group is

$$2.603 + 0.292 = 2.895$$

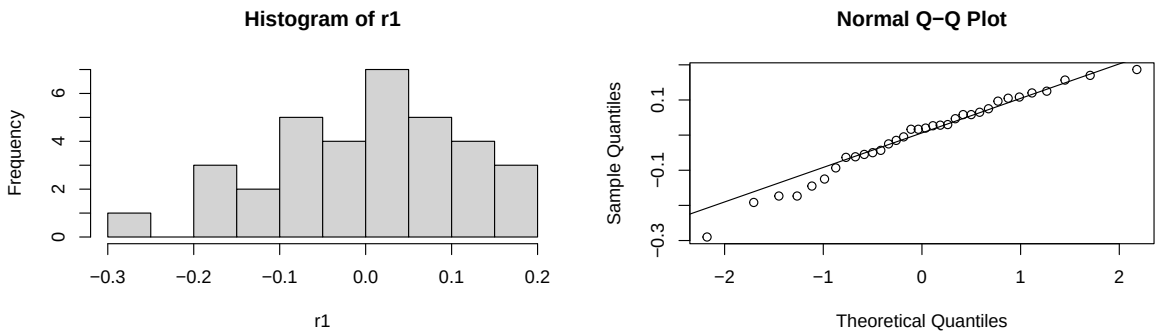
From the Python output, we can read off (the Intercept term) that the estimate for `Alfacyp` is precisely

$$2.895 + 0 = 2.895$$

To check the assumptions, we can use the following code:

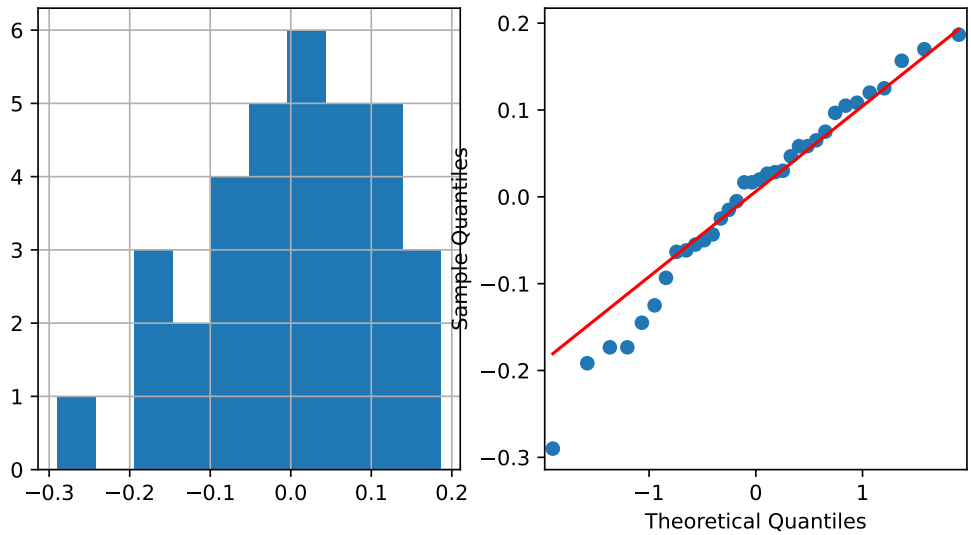
### R code

```
# R
r1 <- residuals(heifers_lm)
hist(r1)
qqnorm(r1); qqline(r1)
```




### Python code

```
# Python
import matplotlib.pyplot as plt

f, axs = plt.subplots(1, 2, figsize=(8,4))
tmp = plt.subplot(121)
heifer_lm.resid.hist();
tmp = plt.subplot(122)
sm.qqplot(heifer_lm.resid, line="q", ax=tmp);
```




For SAS, we have to create a new column containing the residuals in a temporary dataset before creating these plots.

## 8.3 Comparing specific groups

The $F$-test in a One-Way ANOVA indicates if all means are equal, but does not provide further insight into which particular groups differ. If we had specified beforehand that we wished to test if two particular groups $i_1$ and $i_2$ had different means, we could do so with a t-test. Here are the details to compute a Confidence Interval in this case:

1. Compute the estimate of the difference between the two means:
$$\overline{Y_{i_1}} - \overline{Y_{i_2}}$$
2. Compute the standard error of the above estimator:
$$\sqrt{MS_W\left(\frac{1}{n_{i_1}} + \frac{1}{n_{i_2}}\right)}$$
3. Compute the $100(1-\alpha)$ confidence interval as:
$$\overline{Y_{i_1}} - \overline{Y_{i_2}} \pm t_{n-k,\alpha/2} \times \sqrt{MS_W\left(\frac{1}{n_{i_1}} + \frac{1}{n_{i_2}}\right)}$$

**! Important**

If you notice from the output in Example 8.1, the rule-of-thumb regarding standard deviations has not been satisfied. The ratio of largest to smallest standard deviations is slightly more than 2. Hence we should in fact switch to the non-parametric version of the test; the pooled estimate of the variance may not be valid. However, we shall proceed with this dataset just to demonstrate the next few techniques, instead of introducing a new dataset.

**Example 8.3** (Enrofloxacin vs. Control)**.** Let us attempt to answer question (2), that we had set out earlier in Example 8.1.

### R code

```
# R
summary_out <- anova(heifers_lm)
est_coef <- coef(heifers_lm)
est1  <- unname(est_coef[3]) # coefficient for Enrofloxacin
MSW <- summary_out$`Mean Sq`[2]
df <- summary_out$Df[2]
q1 <- qt(0.025, df, 0, lower.tail = FALSE)

lower_ci <- est1 - q1*sqrt(MSW * (1/6 + 1/6))
upper_ci <- est1 + q1*sqrt(MSW * (1/6 + 1/6))
cat("The 95% CI for the diff. between Enrofloxacin and Control is (",
    format(lower_ci, digits = 3), ",",
    format(upper_ci, digits = 3), ").", sep="")
```

```
The 95% CI for the diff. between Enrofloxacin and Control is (-0.0373,0.251).
```

### Python code

```
# Python
est1  = heifer_lm.params.iloc[2] - heifer_lm.params.iloc[1]
MSW = heifer_lm.mse_resid
df = heifer_lm.df_resid
q1 = -stats.t.ppf(0.025, df)

lower_ci = est1 - q1*np.sqrt(MSW * (1/6 + 1/6))
upper_ci = est1 + q1*np.sqrt(MSW * (1/6 + 1/6))
print(f"""The 95% CI for the diff. between Enrofloxacin and control is
({lower_ci:.3f}, {upper_ci:.3f})."""
```

```
The 95% CI for the diff. between Enrofloxacin and control is
(-0.037, 0.251).
```

### SAS code

In order to get SAS to generate the estimate, modify the code to include `clparm` in the `model` statement, and include the `estimate` statement.

```
proc glm data=ST2137.HEIFERS;
    class type;
    model org=type / clparm;
    means type / hovtest=levene welch plots=none;
    lsmeans type / adjust=tukey pdiff alpha=.05;
    estimate 'enro_vs_control' type 0 -1 1 0 0 0;
    output out=work.Oneway_stats r=residual;
    run;
quit;
```

Dependent Variable: org

| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| | 95% Confidence Limits | |
|---|---|---|---|---|---|---|
| enro_vs_control | 0.10666667 | 0.07028852 | 1.52 | 0.1403 | -0.03731284 | 0.25064618 |

As the confidence interval contains the value 0, the binary conclusion would be to not reject the null hypothesis at the 5% level.

## 8.4 Contrast Estimation

A more general comparison, such as the comparison of a collection of $l_1$ groups with another collection of $l_2$ groups, is also possible. First, note that a linear contrast is any linear combination of the individual group means such that the linear coefficients add up to 0. In other words, consider $L$ such that

$$L = \sum_{i=1}^{k} c_i \overline{Y_i}, \text{ where } \sum_{i=1}^{k} c_i = 0$$

Note that the comparison of two groups in Section 8.3 is a special case of this linear contrast.

Here is the procedure for computing confidence intervals for a linear contrast:

1. Compute the estimate of the contrast:

$$L = \sum_{i=1}^{k} c_i \overline{Y_i}$$

2. Compute the standard error of the above estimator:

$$\sqrt{MS_W \sum_{i=1}^{k} \frac{c_i^2}{n_i}}$$

3. Compute the $100(1-\alpha)$ confidence interval as:

$$L \pm t_{n-k,\alpha/2} \times \sqrt{MS_W \sum_{i=1}^{k} \frac{c_i^2}{n_i}}$$

**Example 8.4** (Comparing collection of groups)**.** Let sub-group 1 consist of Ivermectin and Fenbendazole. Here is how we can compute a confidence interval for the difference between this sub-group, and Enrofloxacin.

### R code

```r
c1 <- c(-1, 0.5, 0.5)
n_vals <- c(6, 6, 6)
L <- sum(c1*est_coef[3:5])

#MSW <- summary_out[[1]]$`Mean Sq`[2]
#df <- summary_out[[1]]$Df[2]
se1 <- sqrt(MSW * sum( c1^2 / n_vals ) )

q1 <- qt(0.025, df, 0, lower.tail = FALSE)

lower_ci <- L - q1*se1
upper_ci <- L + q1*se1
cat("The 95% CI for the diff. between the two groups is (",
    format(lower_ci, digits = 2), ",",
    format(upper_ci, digits = 2), ").", sep="")
```

```
The 95% CI for the diff. between the two groups is (0.083,0.33).
```

### Python code

```python
c1 = np.array([-1, 0.5, 0.5])
n_vals = np.array([6, 6, 6,])
L = np.sum(c1 * heifer_lm.params.iloc[2:5])

MSW = heifer_lm.mse_resid
df = heifer_lm.df_resid
q1 = -stats.t.ppf(0.025, df)
se1 = np.sqrt(MSW*np.sum(c1**2 / n_vals))

lower_ci = L - q1*se1
upper_ci = L + q1*se1
print(f"""The 95% CI for the diff. between the two groups is
({lower_ci:.3f}, {upper_ci:.3f})."""")
```

```
The 95% CI for the diff. between the two groups is
(0.083, 0.332).
```

### SAS code

```
proc glm data=ST2137.HEIFERS;
    class type;
    model org=type / clparm;
    means type / hovtest=levene welch plots=none;
    lsmeans type / adjust=tukey pdiff alpha=.05;
    estimate 'group_A_vs_enro' type 0 0 -1 0.5 0.5 0;
    output out=work.Oneway_stats r=residual;
```

```
    run;
quit;
```

| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| | 95% Confidence Limits | |
|---|---|---|---|---|---|---|
| group_A_vs_enro | 0.20750000 | 0.06087164 | 3.41 | 0.0020 | 0.08281009 | 0.33218991 |

## 8.5 Multiple Comparisons

The procedures in the previous two subsections correspond to contrasts that we had specified before collecting or studying the data. If, instead, we wished to perform particular comparisons after studying the group means, or if we wish to compute all pairwise contrasts, then we need to adjust for the fact that we are conducting multiple tests. If we do not do so, the chance of making at least one false positive increases greatly.

### 8.5.1 Bonferroni

The simplest method for correcting for multiple comparisons is to use the Bonferroni correction. Suppose we wish to perform $m$ pairwise comparisons, either as a test or by computing confidence intervals. If we wish to maintain the significance level of each test at $\alpha$, then we should perform each of the $m$ tests/confidence intervals at $\alpha/m$.

### 8.5.2 TukeyHSD

This procedure is known as Tukey's Honestly Significant Difference. It is designed to construct confidence intervals for **all** pairwise comparisons. For the same $\alpha$-level, Tukey's HSD method provides shorter confidence intervals than a Bonferroni correction for all pairwise comparisons.

#### R code

```r
TukeyHSD(aov(heifers_lm), ordered = TRUE)
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level
    factor levels have been ordered

Fit: aov(formula = heifers_lm)

$type
                        diff         lwr       upr     p adj
Enroflox-Control  0.10666667 -0.10812638 0.3214597 0.6563131
Fenbenda-Control  0.23000000  0.01520695 0.4447930 0.0304908
Spiramyc-Control  0.25166667  0.01152074 0.4918126 0.0358454
Alfacyp-Control   0.29166667  0.07687362 0.5064597 0.0034604
```

```
Ivermect-Control  0.39833333  0.18354028 0.6131264 0.0000612
Fenbenda-Enroflox 0.12333333 -0.09145972 0.3381264 0.5093714
Spiramyc-Enroflox 0.14500000 -0.09514593 0.3851459 0.4549043
Alfacyp-Enroflox  0.18500000 -0.02979305 0.3997930 0.1225956
Ivermect-Enroflox 0.29166667  0.07687362 0.5064597 0.0034604
Spiramyc-Fenbenda 0.02166667 -0.21847926 0.2618126 0.9997587
Alfacyp-Fenbenda  0.06166667 -0.15312638 0.2764597 0.9488454
Ivermect-Fenbenda 0.16833333 -0.04645972 0.3831264 0.1923280
Alfacyp-Spiramyc  0.04000000 -0.20014593 0.2801459 0.9953987
Ivermect-Spiramyc 0.14666667 -0.09347926 0.3868126 0.4424433
Ivermect-Alfacyp  0.10666667 -0.10812638 0.3214597 0.6563131
```

**Python code**

```python
import statsmodels.stats.multicomp as mc

cp = mc.MultiComparison(heifers.org, heifers.type)
tk = cp.tukeyhsd()
print(tk)
```

```
  Multiple Comparison of Means - Tukey HSD, FWER=0.05
=========================================================
 group1   group2  meandiff p-adj   lower   upper  reject
---------------------------------------------------------
 Alfacyp  Control  -0.2917 0.0035 -0.5065 -0.0769   True
 Alfacyp Enroflox   -0.185 0.1226 -0.3998  0.0298  False
 Alfacyp Fenbenda  -0.0617 0.9488 -0.2765  0.1531  False
 Alfacyp Ivermect   0.1067 0.6563 -0.1081  0.3215  False
 Alfacyp Spiramyc    -0.04 0.9954 -0.2801  0.2001  False
 Control Enroflox   0.1067 0.6563 -0.1081  0.3215  False
 Control Fenbenda     0.23 0.0305  0.0152  0.4448   True
 Control Ivermect   0.3983 0.0001  0.1835  0.6131   True
 Control Spiramyc   0.2517 0.0358  0.0115  0.4918   True
Enroflox Fenbenda   0.1233 0.5094 -0.0915  0.3381  False
Enroflox Ivermect   0.2917 0.0035  0.0769  0.5065   True
Enroflox Spiramyc    0.145 0.4549 -0.0951  0.3851  False
Fenbenda Ivermect   0.1683 0.1923 -0.0465  0.3831  False
Fenbenda Spiramyc   0.0217 0.9998 -0.2185  0.2618  False
Ivermect Spiramyc  -0.1467 0.4424 -0.3868  0.0935  False
---------------------------------------------------------
```

**SAS output**

| Least Squares Means for effect type<br>Pr > \|t\| for H0: LSMean(i)=LSMean(j)<br><br>Dependent Variable: org | | | | | | |
|---|---|---|---|---|---|---|
| i/j | 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | | 0.0035 | 0.1226 | 0.9488 | 0.6563 | 0.9954 |
| 2 | 0.0035 | | 0.6563 | 0.0305 | <.0001 | 0.0358 |
| 3 | 0.1226 | 0.6563 | | 0.5094 | 0.0035 | 0.4549 |
| 4 | 0.9488 | 0.0305 | 0.5094 | | 0.1923 | 0.9998 |
| 5 | 0.6563 | <.0001 | 0.0035 | 0.1923 | | 0.4424 |
| 6 | 0.9954 | 0.0358 | 0.4549 | 0.9998 | 0.4424 | |

## 8.6 Kruskal-Wallis Procedure

If the assumptions of the ANOVA procedure are not met, we can turn to a non-parametric version - the Kruskal Wallis test. This latter procedure is a generalisation of the Wilcoxon Rank-Sum test for 2 independent samples.

### 8.6.1 Formal Set-up

The test statistic compares the average ranks in the individual groups. If these are close together, we would be inclined to conclude the treatments are equally effective.

The null hypothesis is that all groups follow the same distribution. The alternative hypothesis is that at least one of the groups' distribution differs from another by a location shift. We then proceed with:

1. Pool the observations over all samples, thus constructing a combined sample of size $N = \sum n_i$. Assign ranks to individual observations, using average rank in the case of tied observations. Compute the rank sum $R_i$ for each of the $k$ samples.
2. If there are no ties, compute the test statistic as

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(N+1)$$

3. If there *are* ties, compute the test statistic as

$$H^* = \frac{H}{1 - \frac{\sum_{j=1}^{g}(t_j^3 - t_j)}{N^3 - N}}$$

where $t_j$ refers to the number of observations with the same value in the $j$-th cluster of tied observations and $g$ is the number of tied groups.

Under $H_0$, the test statistic follows a $\chi^2$ distribution with $k-1$ degrees of freedom.

> **❗ Important**
>
> This test should only be used if $n_i \geq 5$ for all groups.

**Example 8.5** (Kruskal-Wallis Test)**.** Here is the code and output from running the Kruskal-Wallis test in the three software.

#### R code

```r
kruskal.test(heifers$org, heifers$type)
```

```
    Kruskal-Wallis rank sum test

data:  heifers$org and heifers$type
Kruskal-Wallis chi-squared = 19.645, df = 5, p-value = 0.001457
```

#### Python code

```python
out = [x[1] for x in heifers.org.groupby(heifers.type)]
kw_out = stats.kruskal(*out)
print(f"""The test statistic is {kw_out.statistic:.3f},
the p-value is {kw_out.pvalue:.3f}.""")
```

```
The test statistic is 19.645,
the p-value is 0.001.
```

**SAS output**

| Wilcoxon Scores (Rank Sums) for Variable org Classified by Variable type | | | | | |
|---|---|---|---|---|---|
| type | N | Sum of Scores | Expected Under H0 | Std Dev Under H0 | Mean Score |
| Ivermect | 6 | 172.00 | 105.0 | 22.120719 | 28.666667 |
| Alfacyp | 6 | 132.50 | 105.0 | 22.120719 | 22.083333 |
| Enroflox | 6 | 67.50 | 105.0 | 22.120719 | 11.250000 |
| Spiramyc | 4 | 78.00 | 70.0 | 18.695420 | 19.500000 |
| Fenbenda | 6 | 110.00 | 105.0 | 22.120719 | 18.333333 |
| Control | 6 | 35.00 | 105.0 | 22.120719 | 5.833333 |
| Average scores were used for ties. | | | | | |

| Kruskal-Wallis Test | | |
|---|---|---|
| Chi-Square | DF | Pr > ChiSq |
| 19.6447 | 5 | 0.0015 |

## 8.7 Summary

The purpose of this topic is to introduce you to the one-way ANOVA model. While there are restrictive distributional assumptions that it entails, I once again urge you to look past, at the information the method conveys. It attempts to compare the within-group variance to the between-group variance. Try to avoid viewing statistical procedures as flowcharts. If an assumption does not hold, or a p-value is borderline significant, try to investigate further on how sensitive the result is to those assumptions.

Our job as analysts does not end after reporting the p-value from the $F$-test. We should try to dig deeper to uncover which groups are the ones that are different from the rest.

Finally, take note that we should specify the contrasts we wish to test/estimate upfront, even before collecting the data. Only the Tukey comparison method (HSD) is valid if we perform multiple comparisons after inspecting the data.

Most of the theoretical portions in this topic were taken from the textbook Rosner (2015).

## 8.8 References

### 8.8.1 Website References

1. Welch's ANOVA This website discusses an alternative test when the equal variance assumption has not been satisfied. It is for information only; it will not be tested.
2. scipy stats This website contains documentation on the distribution-related functions that we might need from scipy stats, e.g. retrieving quantiles.

3. Contrast coding
4. Type I,II,III SS

# 9 Linear Regression

## 9.1 Introduction

Regression analysis is a technique for investigating and modeling the relationship between variables like X and Y. Here are some examples:

1. Within a country, we may wish to use per capita income (X) to estimate the life expectancy (Y) of residents.
2. We may wish to use the size of a crab claw (X) to estimate the closing force that it can exert (Y).
3. We may wish to use the height of a person (X) to estimate their weight (Y).

In all the above cases, we refer to $X$ as the explanatory or independent variable. It is also sometimes referred to as a predictor. $Y$ is referred to as the response or dependent variable. In this topic, we shall first introduce the case of simple linear regression, where we model the $Y$ on a single $X$. In later sections, we shall model the $Y$ on multiple $X$'s. This latter technique is referred to as multiple linear regression.

Regression models are used for two primary purposes:

1. To understand how certain explanatory variables affect the response variable. This aim is typically known as *estimation*, since the primary focus is on estimating the unknown parameters of the model.
2. To predict the response variable for new values of the explanatory variables. This is referred to as *prediction*.

In our course, we shall focus on the estimation aim, since prediction models require a paradigm of their own, and are best learnt alongside a larger suite of models e.g. decision trees, support vector machines, etc.

In subsequent sections, we shall revisit a couple of datasets from earlier topics to run linear regression models on them.

**Example 9.1** (Concrete Data: Flow on Water)**.** Recall the concrete dataset that we first encountered in the topic on summarising data. We shall go on to fit a linear regression to understand the relationship between the output of the flow test, and the amount of water used to create the concrete.

Note that trend in the scatterplot in Figure 9.1. In this topic, we shall figure out how to estimate this line in this topic.

**Example 9.2** (Bike Rental Data)**.** In the introduction to SAS, we encountered data on bike rentals in the USA over a period of 2 years. Here, we shall attempt to model the number of registered users on the number of casual users.

Contingent on whether the day is a working one or not, it does appear that the trendline is different (see Figure 9.2).

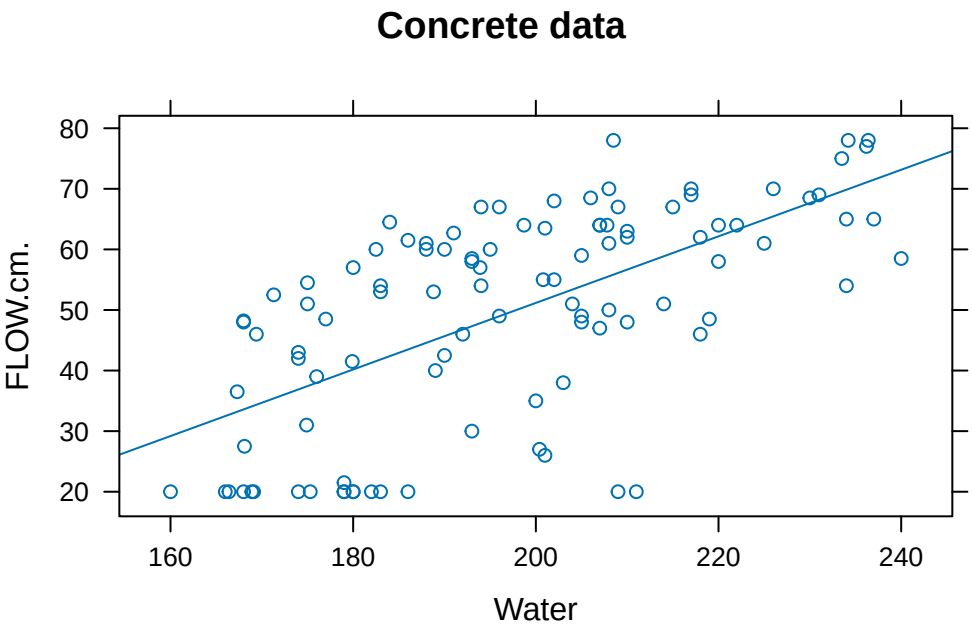



Figure 9.1: Scatterplot with simple linear regression model

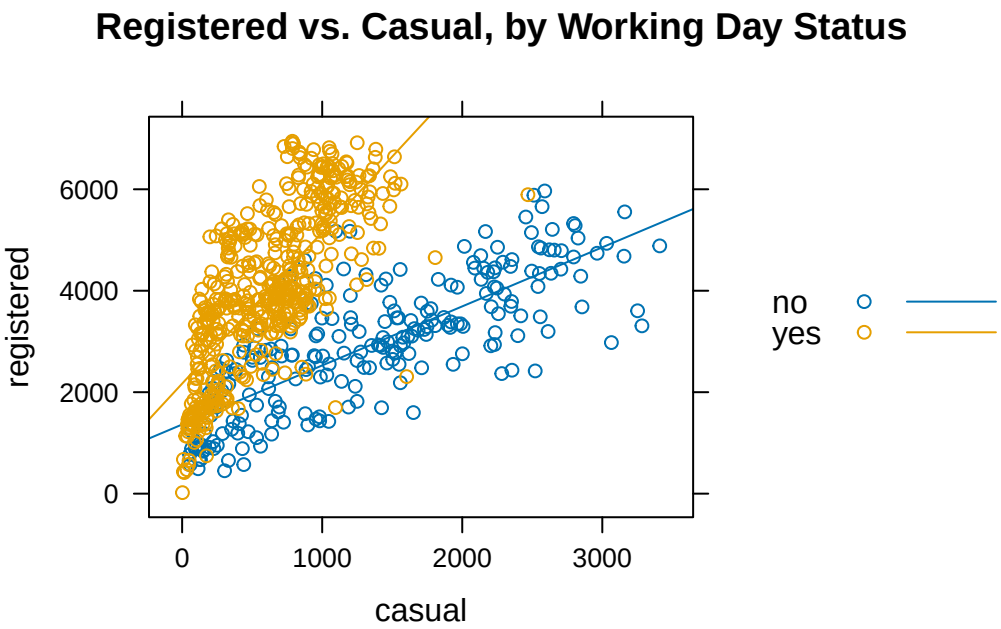



Figure 9.2: Scatterplot of registered vs. casual bike renters

## 9.2 Simple Linear Regression

### 9.2.1 Formal Set-up

The simple linear regression model is applicable when we have observations $(X_i, Y_i)$ for $n$ individuals. For now, let's assume both the $X$ and $Y$ variables are quantitative.

The simple linear regression model is given by

$$Y_i = \beta_0 + \beta_1 X_i + e_i \tag{9.1}$$

where

- $\beta_0$ is intercept term,
- $\beta_1$ is the slope, and
- $e_i$ is an error term, specific to each individual in the dataset.

$\beta_0$ and $\beta_1$ are unknown constants that need to be estimated from the data. There is an implicit assumption in the formulation of the model that there is a linear relationship between $Y_i$ and $X_i$. In terms of distributions, we assume that the $e_i$ are i.i.d Normal.

$$e_i \sim N(0, \sigma^2),\ i = 1 \dots, n \tag{9.2}$$

The constant variance assumption is also referred to as homoscedascity (homo-skee-das-city). The validity of the above assumptions will have to be checked after the model is fitted. All in all, the assumptions imply that:

1. $E(Y_i|X_i) = \beta_0 + \beta_1 X_i$, for $i = 1, \dots, n$.
2. $Var(Y_i|X_i) = Var(e_i) = \sigma^2$, for $i = 1, \dots, n$.
3. The $Y_i$ are independent.
4. The $Y_i$'s are Normally distributed.

### 9.2.2 Estimation

Before deploying or using the model, we need to estimate optimal values to use for the unknown $\beta_0$ and $\beta_1$. We shall introduce the method of Ordinary Least Squares (OLS) for the estimation. Let us define the *error Sum of Squares* to be

$$SS_E = S(\beta_0, \beta_1) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2 \tag{9.3}$$

Then the OLS estimates of $\beta_0$ and $\beta_1$ are given by

$$\underset{\beta_0, \beta_1}{\arg\min} \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2$$

The minimisation above can be carried out analytically, by taking partial derivative with respect to the two parameters and setting them to 0.

$$
\begin{aligned}
\frac{\partial S}{\partial \beta_0} &= -2\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i) = 0 \\
\frac{\partial S}{\partial \beta_1} &= -2\sum_{i=1}^{n} X_i(Y_i - \beta_0 - \beta_1 X_i) = 0
\end{aligned}
$$

Solving and simplifying, we arrive at the following:

$$
\begin{aligned}
\hat{\beta}_1 &= \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n}(X_i - \bar{X})^2} \\
\hat{\beta}_0 &= \bar{Y} - \hat{\beta}_0 \bar{X}
\end{aligned}
$$

where $\bar{Y} = (1/n)\sum Y_i$ and $\bar{X} = (1/n)\sum X_i$.

If we define the following sums:

$$
\begin{aligned}
S_{XY} &= \sum_{i=1}^{n} X_i Y_i - \frac{(\sum_{i=1}^{n} X_i)(\sum_{i=1}^{n} Y_i)}{n} \\
S_{XX} &= \sum_{i=1}^{n} X_i^2 - \frac{(\sum_{i=1}^{n} X_i)^2}{n}
\end{aligned}
$$

then a form convenient for computation of $\hat{\beta}_1$ is

$$
\hat{\beta}_1 = \frac{S_{XY}}{S_{XX}}
$$

Once we have the estimates, we can use Equation 9.1 to compute *fitted* values for each observation, corresponding to our best guess of the mean of the distributions from which the observations arose:

$$
\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i, \quad i = 1, \dots, n
$$

As always, we can form residuals as the deviations from fitted values.

$$
r_i = Y_i - \hat{Y}_i \tag{9.4}
$$

Residuals are our best guess at the unobserved error terms $e_i$. Squaring the residuals and summing over all observations, we can arrive at the following decomposition, which is very similar to the one in the ANOVA model:

$$
\underbrace{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}_{SS_T} = \underbrace{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}_{SS_{Res}} + \underbrace{\sum_{i=1}^{n}(\hat{Y}_i - \bar{Y})^2}_{SS_{Reg}}
$$

where

- $SS_T$ is known as the total sum of squares.
- $SS_{Res}$ is known as the residual sum of squares.
- $SS_{Reg}$ is known as the regression sum of squares.

In our model, recall from Equation 9.2 that we had assumed equal variance for all our observations. We can estimate $\sigma^2$ with

$$\hat{\sigma^2} = \frac{SS_{Res}}{n-2}$$

Our distributional assumptions lead to the following for our estimates $\hat{\beta}_0$ and $\hat{\beta}_1$:

$$\hat{\beta}_0 \sim N(\beta_0,\ \sigma^2(1/n + \bar{X}^2/S_{XX})) \tag{9.5}$$
$$\hat{\beta}_1 \sim N(\beta_1,\ \sigma^2/S_{XX}) \tag{9.6}$$

The above are used to construct confidence intervals for $\beta_0$ and $\beta_1$, based on $t$-distributions.

## 9.3 Hypothesis Test for Model Significance

The first test that we introduce here is to test if the coefficient $\beta_1$ is significantly different from 0. It is essentially a test of whether it was worthwhile to use a regression model of the form in Equation 9.1, instead of a simple mean to represent the data.

The null and alternative hypotheses are:

$$\begin{aligned} H_0 &: \beta_1 = 0 \\ H_1 &: \beta_1 \neq 0 \end{aligned}$$

The test statistic is

$$F_0 = \frac{SS_{Reg}/1}{SS_{Res}/(n-2)} \tag{9.7}$$

Under the null hypothesis, $F_0 \sim F_{1,n-2}$.

It is also possible to perform this same test as a $t$-test, using the result earlier. The statement of the hypotheses is equivalent to the $F$-test. The test statistic

$$T_0 = \frac{\hat{\beta}_1}{\sqrt{\hat{\sigma^2}/S_{XX}}} \tag{9.8}$$

Under $H_0$, the distribution of $T_0$ is $t_{n-2}$. This $t$-test and the earlier $F$-test in this section are *identical*. It can be proved that $F_0 = T_0^2$; the obtained $p$-values will be identical.

### 9.3.1 Coefficient of Determination, $R^2$

The coefficient of determination $R^2$ is defined as

$$R^2 = 1 - \frac{SS_{Res}}{SS_T} = \frac{SS_{Reg}}{SS_T}$$

It can be interpreted as the proportion of variation in $Yi$, explained by the inclusion of $X_i$. Since $0 \leq SS_{Res} \leq SS_T$, we can easily prove that $0 \leq R^2 \leq 1$. The larger the value of $R^2$ is, the better the model is.

When we get to the case of multiple linear regression, take note that simply including more variables in the model will increase $R^2$. This is undesirable; it is preferable to have a parsimonious model that explains the response variable well.

**Example 9.3** (Concrete Data Model)**.** In this example, we focus on the estimation of the model parameters for the two variables we introduced in Example 9.1

#### R code

```
#R
concrete <- read.csv("data/concrete+slump+test/slump_test.data")
names(concrete)[c(1,11)] <- c("id", "Comp.Strength")
lm_flow_water <- lm(FLOW.cm. ~ Water, data=concrete)
summary(lm_flow_water)
```

```
Call:
lm(formula = FLOW.cm. ~ Water, data = concrete)

Residuals:
    Min      1Q  Median      3Q     Max 
-37.211 -10.836   2.734  11.031  22.163 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -58.72755   13.28635  -4.420 2.49e-05 ***
Water         0.54947    0.06704   8.196 8.10e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 13.68 on 101 degrees of freedom
Multiple R-squared:  0.3995,	Adjusted R-squared:  0.3935 
F-statistic: 67.18 on 1 and 101 DF,  p-value: 8.097e-13
```

#### Python code

```python
#Python
import pandas as pd
import numpy as np
import statsmodels.api as sm
from statsmodels.formula.api import ols

concrete = pd.read_csv("data/concrete+slump+test/slump_test.data")
concrete.rename(columns={'No':'id',
                         'Compressive Strength (28-day)(Mpa)':'Comp_Strength',
                         'FLOW(cm)': 'Flow'},
                inplace=True)
lm_flow_water = ols('Flow ~ Water', data=concrete).fit()
print(lm_flow_water.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Flow   R-squared:                       0.399
Model:                            OLS   Adj. R-squared:                  0.394
Method:                 Least Squares   F-statistic:                     67.18
Date:                Sun, 14 Dec 2025   Prob (F-statistic):           8.10e-13
Time:                        07:44:51   Log-Likelihood:                -414.60
No. Observations:                 103   AIC:                             833.2
Df Residuals:                     101   BIC:                             838.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -58.7276     13.286     -4.420      0.000     -85.084     -32.371
Water          0.5495      0.067      8.196      0.000       0.416       0.682
==============================================================================
Omnibus:                        6.229   Durbin-Watson:                   1.843
Prob(Omnibus):                  0.044   Jarque-Bera (JB):                5.873
Skew:                          -0.523   Prob(JB):                       0.0530
Kurtosis:                       2.477   Cond. No.                     1.95e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.95e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

**SAS output**

Model: MODEL1
Dependent Variable: FLOW(cm)

| | |
|---|---|
| Number of Observations Read | 103 |
| Number of Observations Used | 103 |

| Analysis of Variance | | | | | |
|---|---|---|---|---|---|
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Model | 1 | 12576 | 12576 | 67.18 | <.0001 |
| Error | 101 | 18907 | 187.19667 | | |
| Corrected Total | 102 | 31483 | | | |

| | | | |
|---|---|---|---|
| Root MSE | 13.68198 | R-Square | 0.3995 |
| Dependent Mean | 49.61068 | Adj R-Sq | 0.3935 |
| Coeff Var | 27.57871 | | |

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| Variable | DF | Parameter Estimate | Standard Error | t Value | Pr > \|t\| |
| Intercept | 1 | -58.72755 | 13.28635 | -4.42 | <.0001 |
| Water | 1 | 0.54947 | 0.06704 | 8.20 | <.0001 |

From the output, we can note that the *estimated* model for Flow ($Y$) against Water ($X$) is:

$$Y = -58.73 + 0.55X$$

The estimates are $\hat{\beta}_0 = -58.73$ and $\hat{\beta}_1 = 0.55$. This is the precise equation that was plotted in Figure 9.1. The $R^2$ is labelled as "Multiple R-squared" in the R output. The value is 0.3995, which means that about 40% of the variation in $Y$ is explained by $X$.

A simple interpretation[1] of the model is as follows:

> For every 1 unit increase in Water, there is an average associated increase in Flow rate of 0.55 units.

To obtain confidence intervals for the parameters, we can use the following code in R. The Python summary already contains the confidence intervals.

[1]This interpretation has to be taken very cautiously, especially when there are other explanatory variables in the model.

### R code

```
#R
confint(lm_flow_water)
```

```
                  2.5 %      97.5 %
(Intercept) -85.0841046 -32.3709993
Water         0.4164861   0.6824575
```

We can read off that the 95% Confidence intervals are:

- For $\beta_0$: (-85.08, -32.37)
- For $\beta_1$: (0.42, 0.68)

**Example 9.4** (Bike Data F-test)**.** In this example, we shall fit a simple linear regression model to the bike data, *constrained to the non-working days*. In other words, we shall focus on fitting just the blue line, from the blue points, in Figure 9.2.

### R code

```
#R
bike2 <- read.csv("data/bike2.csv")
bike2_sub <- bike2[bike2$workingday == "no", ]
lm_reg_casual <- lm(registered ~ casual, data=bike2_sub)
anova(lm_reg_casual)
```

```
Analysis of Variance Table

Response: registered
           Df    Sum Sq   Mean Sq F value    Pr(>F)
casual      1 237654556 237654556  369.25 < 2.2e-16 ***
Residuals 229 147386970    643611
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

### Python code

```
#Python
bike2 = pd.read_csv("data/bike2.csv")
bike2_sub = bike2[bike2.workingday == "no"]

lm_reg_casual = ols('registered ~ casual', bike2_sub).fit()
anova_tab = sm.stats.anova_lm(lm_reg_casual,)
anova_tab
```

```
             df        sum_sq       mean_sq           F        PR(>F)
casual      1.0  2.376546e+08  2.376546e+08  369.251728  1.183368e-49
Residual  229.0  1.473870e+08  6.436112e+05         NaN           NaN
```

**SAS output**

Model: MODEL1
Dependent Variable: registered

| | |
|---|---|
| Number of Observations Read | 231 |
| Number of Observations Used | 231 |

| Analysis of Variance | | | | | |
|---|---|---|---|---|---|
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Model | 1 | 237654556 | 237654556 | 369.25 | <.0001 |
| Error | 229 | 147386970 | 643611 | | |
| Corrected Total | 230 | 385041526 | | | |

| | | | |
|---|---|---|---|
| Root MSE | 802.25384 | R-Square | 0.6172 |
| Dependent Mean | 2959.03463 | Adj R-Sq | 0.6155 |
| Coeff Var | 27.11201 | | |

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| Variable | DF | Parameter Estimate | Standard Error | t Value | Pr > \|t\| |
| Intercept | 1 | 1362.63124 | 98.42765 | 13.84 | <.0001 |
| casual | 1 | 1.16429 | 0.06059 | 19.22 | <.0001 |

The output above includes the sum-of-squares that we need to perform the $F$-test outlined in Section 9.3. From the output table, we can see that $SS_{Reg} = 237654556$ and $SS_{Res} = 147386970$. The value of $F_0$ for this dataset is 369.25. The $p$-value is extremely small ($2 \times 10^{-16}$), indicating strong evidence against $H_0$, i.e. that $\beta_1 = 0$.

Actually, if you observe carefully in Example 9.3, the output from R contains both the $t$-test for significance of $\beta_1$, and the $F$-test statistic based on sum-of-squares. The $p$-value in both cases is $8.10 \times 10^{1-3}$.

In linear regression, we almost always wish to use the model to understand what the mean of future observations would be. In the concrete case, we may wish to use the model to understand how the Flow test output values change as the amount of Water in the mixture changes. This is because, based on our formulation,

$$E(Y|X) = \beta_0 + \beta_1 X$$

After estimating the parameters, we would have:

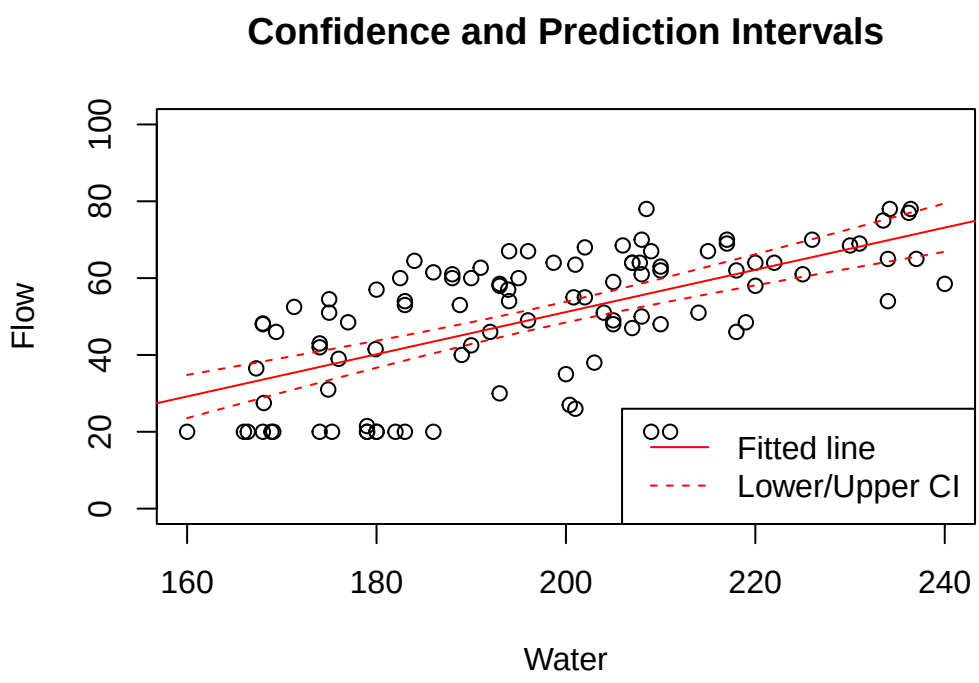
$$\widehat{E(Y|X)} = \hat{\beta}_0 + \hat{\beta}_1 X$$

Thus we can vary the values of $X$ to study how the mean of $Y$ changes. Here is how we can do so in the concrete model for data.

**Example 9.5** (Concrete Data Predicted Means)**.** In order to create the predicted means, we shall have to create a dataframe with the new values for which we require the predictions. We are first going to set up a new matrix of $X$-values corresponding to the desired range.

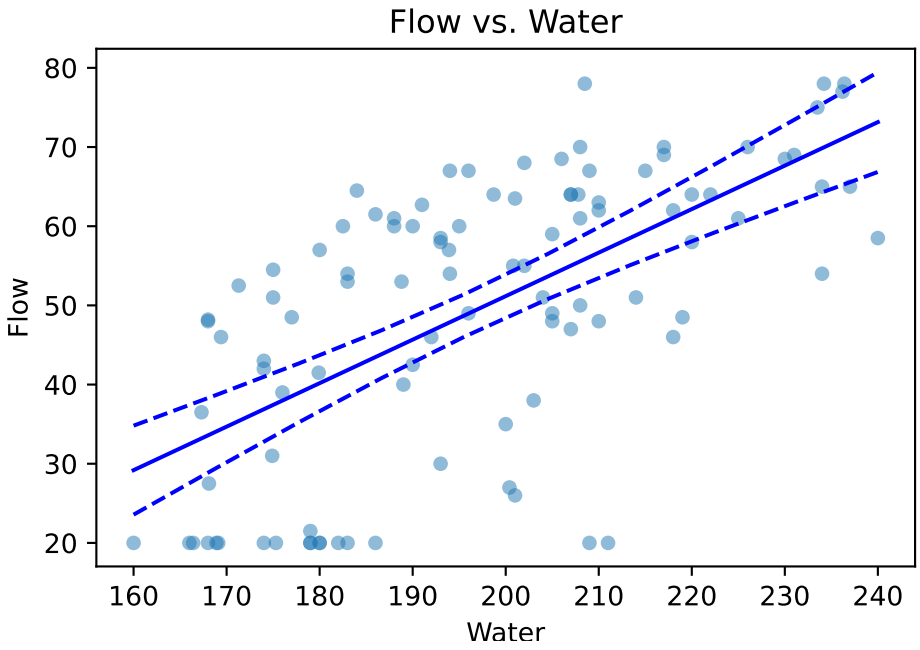
**R code**

```
#R
new_df <- data.frame(Water = seq(160, 240, by = 5))
conf_intervals <- predict(lm_flow_water, new_df, interval="conf")

plot(concrete$Water, concrete$FLOW.cm., ylim=c(0, 100),
     xlab="Water", ylab="Flow", main="Confidence and Prediction Intervals")
abline(lm_flow_water, col="red")
lines(new_df$Water, conf_intervals[,"lwr"], col="red", lty=2)
lines(new_df$Water, conf_intervals[,"upr"], col="red", lty=2)
legend("bottomright", legend=c("Fitted line", "Lower/Upper CI"),
       lty=c(1,2), col="red")
```

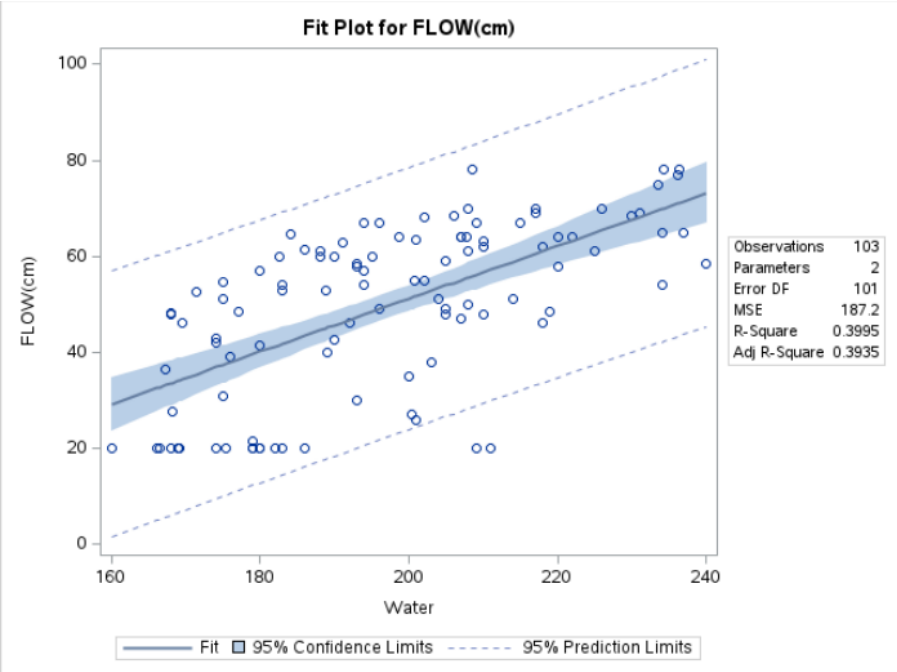


**Python code**

```
# Python
new_df = sm.add_constant(pd.DataFrame({'Water' : np.linspace(160,240, 10)}))

predictions_out = lm_flow_water.get_prediction(new_df)

ax = concrete.plot(x='Water', y='Flow', kind='scatter', alpha=0.5 )
ax.set_title('Flow vs. Water');
ax.plot(new_df.Water, predictions_out.conf_int()[:, 0].reshape(-1),
        color='blue', linestyle='dashed');
ax.plot(new_df.Water, predictions_out.conf_int()[:, 1].reshape(-1),
        color='blue', linestyle='dashed');
ax.plot(new_df.Water, predictions_out.predicted, color='blue');
```



**SAS Output**




The fitted line is the straight line formed using $\hat{\beta}_0$ and $\hat{\beta}_1$. The dashed lines are 95% Confidence Intervals for $E(Y|X)$, for varying values of $X$. They are formed by joining up the lower bounds and the upper bounds separately. Notice how the limits get wider the further away we are from $\bar{X} \approx 200$.

## 9.4 Multiple Linear Regression

### 9.4.1 Formal Setup

When we have more than 1 explanatory variable, we turn to multiple linear regression - generalised version of what we have been dealing with so far. We would still have observed information from $n$ individuals, but for each one, we now observe a vector of values:

$$Y_i,\ X_{1,i},\ X_{2,i}, \dots,\ X_{p-1,i}, X_{p,i}$$

In other words, we observe $p$ independent variables and 1 response variable for each individual in our dataset. The analogous equation to Equation 9.1 is

$$Y_i = \beta_0 + \beta_1 X_{1,i} + \cdots + \beta_p X_{p,i} + e \tag{9.9}$$

It is easier to write things with matrices for multiple linear regression:

$$\mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix},\ \mathbf{X} = \begin{bmatrix} 1 & X_{1,1} & X_{2,1} & \cdots & X_{p,1} \\ 1 & X_{1,2} & X_{2,2} & \cdots & X_{p,2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & X_{1,n} & X_{2,n} & \cdots & X_{p,n} \end{bmatrix},\ \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix},\ e = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

With the above matrices, we can re-write Equation 9.9 as

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{e}$$

We retain the same distributional assumptions as in Section 9.2.1.

### 9.4.2 Estimation

Similar to Section 9.2.2, we can define $SS_E$ to be

$$SS_E = S(\beta_0, \beta_1, \dots, \beta_p) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_{1,i} - \dots - \beta_p X_{p,i})^2 \tag{9.10}$$

Minimising the above cost function leads to the OLS estimates:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

The fitted values can be computed with

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

Residuals are obtained as

$$\mathbf{r} = \mathbf{Y} - \hat{\mathbf{Y}}$$

Finally, we estimate $\sigma^2$ using

$$\hat{\sigma^2} = \frac{SS_{Res}}{n-p} = \frac{\mathbf{r}'\mathbf{r}}{n-p}$$

### 9.4.3 Coefficient of Determination, $R^2$

In the case of multiple linear regression, $R^2$ is calculated exactly as in simple linear regression, and its interpretation remains the same:

$$R^2 = 1 - \frac{SS_{Res}}{SS_T}$$

However, note that $R^2$ can be inflated simply by adding more terms to the model (even insignificant terms). Thus, we use the adjusted $R^2$, which penalizes us for adding more and more terms to the model:

$$R^2_{adj} = 1 - \frac{SS_{Res}/(n-p)}{SS_T/(n-1)}$$

### 9.4.4 Hypothesis Tests

The $F$-test in the multiple linear regression helps determine if our regression model provides any advantage over the simple mean model. The null and alternative hypotheses are:

$$\begin{aligned} H_0 &: \beta_1 = \beta_2 = \dots = \beta_p = 0 \\ H_1 &: \beta_j \neq 0 \text{ for at least one } j \in \{1, 2, \dots, p\} \end{aligned}$$

The test statistic is

$$F_1 = \frac{SS_{Reg}/p}{SS_{Res}/(n-p-1)} \tag{9.11}$$

Under the null hypothesis, $F_0 \sim F_{p,n-p-1}$.

It is also possible to test for the significance of individual $\beta$ terms, using a $t$-test. The output is typically given for all the coefficients in a table. The statement of the hypotheses pertaining to these tests is:

$$\begin{aligned} H_0 &: \beta_j = 0 \\ H_1 &: \beta_j \neq 0 \end{aligned}$$

However, note that these $t$-tests are partial because it should be interpreted as a test of the contribution of $\beta_j$, *given that all other terms are already in the model.*

**Example 9.6** (Concrete Data Multiple Linear Regression)**.** In this second model for concrete, we add a second predictor variable, Slag. The updated model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + e$$

where $X_1$ corresponds to Water, and $X_2$ corresponds to Slag.

### R code

```
# R
lm_flow_water_slag <- lm(FLOW.cm. ~ Water + Slag, data=concrete)
summary(lm_flow_water_slag)
```

```
Call:
lm(formula = FLOW.cm. ~ Water + Slag, data = concrete)

Residuals:
    Min      1Q  Median      3Q     Max 
-32.687 -10.746   2.010   9.224  23.927 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -50.26656   12.38669  -4.058 9.83e-05 ***
Water         0.54224    0.06175   8.781 4.62e-14 ***
Slag         -0.09023    0.02064  -4.372 3.02e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.6 on 100 degrees of freedom
Multiple R-squared:  0.4958,	Adjusted R-squared:  0.4857 
F-statistic: 49.17 on 2 and 100 DF,  p-value: 1.347e-15
```

**Python code**

```python
# Python
lm_flow_water_slag = ols('Flow ~ Water + Slag', data=concrete).fit()
print(lm_flow_water_slag.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Flow   R-squared:                       0.496
Model:                            OLS   Adj. R-squared:                  0.486
Method:                 Least Squares   F-statistic:                     49.17
Date:                Sun, 14 Dec 2025   Prob (F-statistic):           1.35e-15
Time:                        07:44:52   Log-Likelihood:                -405.59
No. Observations:                 103   AIC:                             817.2
Df Residuals:                     100   BIC:                             825.1
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -50.2666     12.387     -4.058      0.000     -74.841     -25.692
Water          0.5422      0.062      8.781      0.000       0.420       0.665
Slag          -0.0902      0.021     -4.372      0.000      -0.131      -0.049
==============================================================================
Omnibus:                        5.426   Durbin-Watson:                   2.029
Prob(Omnibus):                  0.066   Jarque-Bera (JB):                4.164
Skew:                          -0.371   Prob(JB):                        0.125
Kurtosis:                       2.353   Cond. No.                     2.14e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.14e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

**SAS output**

Model: MODEL1
Dependent Variable: FLOW(cm)

| | |
|---|---|
| Number of Observations Read | 103 |
| Number of Observations Used | 103 |

| Analysis of Variance | | | | | |
|---|---|---|---|---|---|
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Model | 2 | 15610 | 7804.88076 | 49.17 | <.0001 |
| Error | 100 | 15873 | 158.73157 | | |
| Corrected Total | 102 | 31483 | | | |

| | | | |
|---|---|---|---|
| Root MSE | 12.59887 | R-Square | 0.4958 |
| Dependent Mean | 49.61068 | Adj R-Sq | 0.4857 |
| Coeff Var | 25.39548 | | |

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| Variable | DF | Parameter Estimate | Standard Error | t Value | Pr > \|t\| |
| Intercept | 1 | -50.26656 | 12.38669 | -4.06 | <.0001 |
| Water | 1 | 0.54224 | 0.06175 | 8.78 | <.0001 |
| Slag | 1 | -0.09023 | 0.02064 | -4.37 | <.0001 |

The $F$-test is now concerned with the hypotheses:

$$\begin{aligned} H_0 &: \beta_1 = \beta_2 = 0 \\ H_1 &: \beta_1 \neq 0 \text{ or } \beta_2 \neq 0 \end{aligned}$$

From the output above, we can see that $F_1 = 49.17$, with a corresponding $p$-value of $1.3 \times 10^{-15}$. The individual $t$-tests for the coefficients all indicate significant differences from 0. The final estimated model can be written as

$$Y = -50.27 + 0.54X_1 - 0.09X_2$$

Notice that the coefficients have changed slightly from the model in Example 9.3. Notice also that we have an improved $R^2$ of 0.50. However, as we pointed out earlier, we should be using the adjusted $R^2$, which adjusts for the additional variable included. This value is 0.49.

While we seem to have found a better model than before, we still have to assess if all the assumptions listed in Section 9.2.1 have been met. We shall do so in subsequent sections.

## 9.5 Indicator Variables

### 9.5.1 Including a Categorical Variable

The explanatory variables in a linear regression model do not need to be continuous. Categorical variables can also be included in the model. In order to include them, they have to be coded using dummy variables.

For instance, suppose that we wish to include gender in a model as $X_3$. There are only two possible genders in our dataset: Female and Male. We can represent $X_3$ as an indicator variable, with

$$X_{3,i} = \begin{cases} 1 & \text{individual } i \text{ is male} \\ 0 & \text{individual } i \text{ is female} \end{cases}$$

The model (without subscripts for the $n$ individuals) is then:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + e$$

For females, the value of $X_3$ is 0. Hence the model reduces to

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + e$$

On the other hand, for males, the model reduces to

$$Y = (\beta_0 + \beta_3) + \beta_1 X_1 + \beta_2 X_2 + e$$

The difference between the two models is in the intercept. The other coefficients remain the same.

In general, if the categorical variable has $a$ levels, we will need $a - 1$ columns of indicator variables to represent it. This is in contrast to machine learning models which use one-hot encoding. The latter encoding results in columns that are linearly dependent if we include an intercept term in the model.

**Example 9.7** (Bike Data Working Day)**.** In this example, we shall improve on the simple linear regression model from Example 9.4. Instead of a single model for just non-working days, we shall fit separate models for working and non-working days by including that variable as a categorical one.

#### R code

```
# R
lm_reg_casual2 <- lm(registered ~ casual + workingday, data=bike2)
summary(lm_reg_casual2)
```

```
Call:
lm(formula = registered ~ casual + workingday, data = bike2)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-3381.8  -674.8   -22.5   792.4  2683.6 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   6.052e+02  1.188e+02   5.095 4.45e-07 ***
casual        1.717e+00  6.893e-02  24.905  < 2e-16 ***
workingdayyes 2.332e+03  1.017e+02  22.921  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1094 on 728 degrees of freedom
Multiple R-squared:  0.5099,	Adjusted R-squared:  0.5086 
F-statistic: 378.7 on 2 and 728 DF,  p-value: < 2.2e-16
```

### Python code

```python
# Python
lm_reg_casual2 = ols('registered ~ casual + workingday', bike2).fit()
print(lm_reg_casual2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             registered   R-squared:                       0.510
Model:                            OLS   Adj. R-squared:                  0.509
Method:                 Least Squares   F-statistic:                     378.7
Date:                Sun, 14 Dec 2025   Prob (F-statistic):          1.81e-113
Time:                        07:44:52   Log-Likelihood:                -6150.8
No. Observations:                 731   AIC:                         1.231e+04
Df Residuals:                     728   BIC:                         1.232e+04
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================================
                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
Intercept            605.2254    118.790      5.095      0.000     372.013     838.438
workingday[T.yes]   2331.7334    101.730     22.921      0.000    2132.015    2531.452
casual                 1.7167      0.069     24.905      0.000       1.581       1.852
==============================================================================
Omnibus:                        2.787   Durbin-Watson:                   0.595
Prob(Omnibus):                  0.248   Jarque-Bera (JB):                2.479
Skew:                          -0.059   Prob(JB):                        0.290
Kurtosis:                       2.740   Cond. No.                     4.05e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.05e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

**SAS output**

**Least Squares Model (No Selection)**

| Analysis of Variance | | | | | |
|---|---|---|---|---|---|
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Model | 2 | 906183210 | 453091605 | 378.73 | <.0001 |
| Error | 728 | 870928763 | 1196331 | | |
| Corrected Total | 730 | 1777111972 | | | |

| | |
|---|---|
| Root MSE | 1093.76904 |
| Dependent Mean | 3656.17237 |
| R-Square | 0.5099 |
| Adj R-Sq | 0.5086 |
| AIC | 10966 |
| AICC | 10966 |
| SBC | 10247 |

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| Parameter | DF | Estimate | Standard Error | t Value | Pr > \|t\| |
| Intercept | 1 | 2936.958715 | 64.348633 | 45.64 | <.0001 |
| casual | 1 | 1.716688 | 0.068929 | 24.91 | <.0001 |
| workingday no | 1 | -2331.733359 | 101.729640 | -22.92 | <.0001 |
| workingday yes | 0 | 0 | . | . | . |

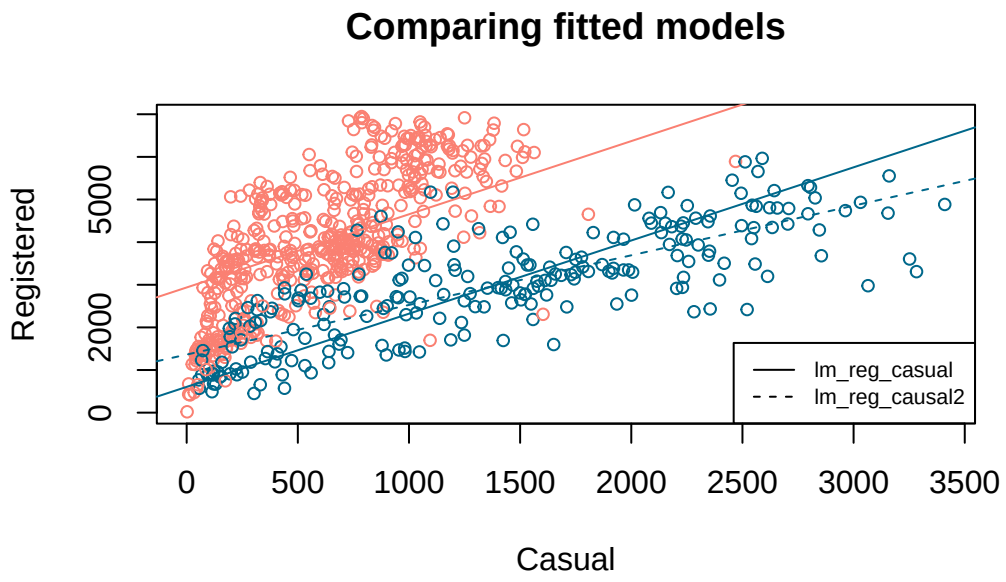
The estimated model is now

$$Y = 605 + 1.72X_1 + 2330X_2$$

But $X_2 = 1$ for working days and $X_2 = 0$ for non-working days. This results in two *separate* models for the two types of days:

$$Y = \begin{cases} 605 + 1.72X_1, & \text{for non-working days} \\ 2935 + 1.72X_1, & \text{for working days} \end{cases}$$

We can plot the two models on the scatterplot to see how they work better than the original model.




The dashed line corresponds to the earlier model, from Example 9.7. With the new model, we have fitted separate intercepts to the two days, but the same slope. The benefit of fitting the model in this way, instead of breaking up the data into two portions and a different model on each one is that we use the entire dataset to estimate the variability. If we wish to fit separate intercepts *and* slopes, we need to include an *interaction term*, which is what the next subsection is about.

## 9.6 Interaction term

A more complex model arises from an interaction between two terms. Here, we shall consider an interaction between a continuous variable and a categorical explanatory variable. Suppose that we have three predictors: height ($X_1$), weight ($X_2$) and gender ($X_3$). As spelt out in Section 9.5.1, we should use indicator variables to represent $X_3$ in the model.

If we were to include an *interaction* between gender and weight, we would be allowing for a males and females to have separate coefficients for $X_2$. Here is what the model would appear as:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_2 X_3 + e$$

Remember that $X_3$ will be 1 for males and 0 for females. The simplified equation for males would be:

$$Y = (\beta_0 + \beta_3) + \beta_1 X_1 + (\beta_2 + \beta_4) X_2 + e$$

For females, it would be:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + e$$

Both the intercept and coefficient of $X_2$ are different now. Recall that in Section 9.5.1, only the intercept term was different.

**Example 9.8** (Bike Data Working Day)**.** Finally, we shall include an interaction in the model for bike rentals, resulting in separate intercepts and separate slopes.

### R code

```r
# R
lm_reg_casual3 <- lm(registered ~ casual * workingday, data=bike2)
summary(lm_reg_casual3)
```

```
Call:
lm(formula = registered ~ casual * workingday, data = bike2)

Residuals:
    Min      1Q  Median      3Q     Max 
-4643.5  -733.0   -57.3   675.9  2532.1 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)          1.363e+03  1.199e+02  11.361  < 2e-16 ***
casual               1.164e+00  7.383e-02  15.769  < 2e-16 ***
workingdayyes        8.063e+02  1.446e+02   5.578 3.44e-08 ***
casual:workingdayyes 1.819e+00  1.340e-01  13.575  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 977.6 on 727 degrees of freedom
Multiple R-squared:  0.609, Adjusted R-squared:  0.6074 
F-statistic: 377.5 on 3 and 727 DF,  p-value: < 2.2e-16
```

### Python code

```python
# Python
lm_reg_casual3 = ols('registered ~ casual * workingday', bike2).fit()
print(lm_reg_casual3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             registered   R-squared:                       0.609
Model:                            OLS   Adj. R-squared:                  0.607
Method:                 Least Squares   F-statistic:                     377.5
Date:                Sun, 14 Dec 2025   Prob (F-statistic):          9.38e-148
Time:                        07:44:52   Log-Likelihood:                -6068.3
No. Observations:                 731   AIC:                         1.214e+04
Df Residuals:                     727   BIC:                         1.216e+04
Df Model:                           3
Covariance Type:            nonrobust
==================================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
Intercept                  1362.6312    119.942     11.361      0.000    1127.158    1598.105
```

```
workingday[T.yes]          806.2675    144.551      5.578      0.000     522.480    1090.055
casual                       1.1643      0.074     15.769      0.000       1.019       1.309
casual:workingday[T.yes]     1.8186      0.134     13.575      0.000       1.556       2.082
==============================================================================
Omnibus:                        7.936   Durbin-Watson:                   0.528
Prob(Omnibus):                  0.019   Jarque-Bera (JB):               11.531
Skew:                          -0.025   Prob(JB):                      0.00313
Kurtosis:                       3.613   Cond. No.                     5.72e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 5.72e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

### SAS output

| Analysis of Variance | | | | | |
|---|---|---|---|---|---|
| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
| Model | 3 | 1082305155 | 360768385 | 377.48 | <.0001 |
| Error | 727 | 694806817 | 955718 | | |
| Corrected Total | 730 | 1777111972 | | | |

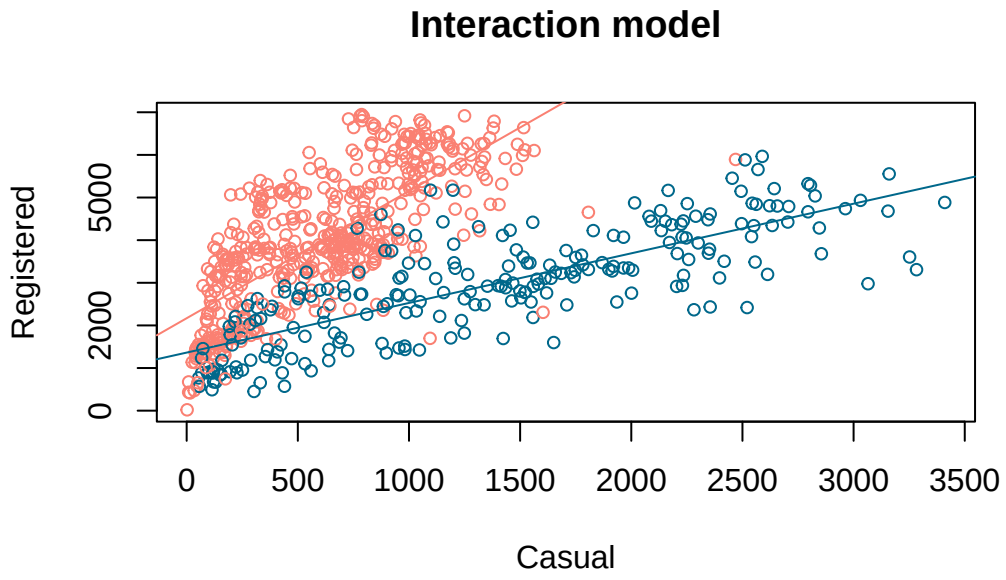
| | |
|---|---|
| Root MSE | 977.60819 |
| Dependent Mean | 3656.17237 |
| R-Square | 0.6090 |
| Adj R-Sq | 0.6074 |
| AIC | 10803 |
| AICC | 10803 |
| SBC | 10088 |

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| Parameter | DF | Estimate | Standard Error | t Value | Pr > \|t\| |
| Intercept | 1 | 2168.898711 | 80.678926 | 26.88 | <.0001 |
| casual | 1 | 2.982922 | 0.111786 | 26.68 | <.0001 |
| workingday no | 1 | -806.267470 | 144.551360 | -5.58 | <.0001 |
| workingday yes | 0 | 0 | . | . | . |
| casual*workingday no | 1 | -1.818628 | 0.133968 | -13.58 | <.0001 |
| casual*workingday yes | 0 | 0 | . | . | . |

Notice that $R^2_{adj}$ has increased from 50.8% to 60.7%. The estimated models for each type of day are:

$$Y = \begin{cases} 1362 + 1.16X_1, & \text{for non-working days} \\ 2168 + 2.97X_1, & \text{for working days} \end{cases}$$

Here is visualisation of the lines that have been estimated for each sub-group of day. This is the image that we had earlier on Figure 9.2.




## 9.7 Residual Diagnostics

Recall from Equation 9.4 that residuals are computed as

$$r_i = Y_i - \hat{Y}_i$$

Residual analysis is a standard approach for identifying how we can improve a model. In the case of linear regression, we can use the residuals to assess if the distributional assumptions hold. We can also use residuals to identify influential points that are masking the general trend of other points. Finally, residuals can provided some direction on how to improve the model.

### 9.7.1 Standardised Residuals

It can be shown that the variance of the residuals is in fact not constant! Let us define the hat-matrix as

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$$

The diagonal values of $\mathbf{H}$ will be denoted $h_{ii}$, for $i = 1, \dots, n$. It can then be shown that

$$Var(r_i) = \sigma^2(1 - h_{ii}), \quad Cov(r_i, r_j) = -\sigma^2 h_{ij}$$

As such, we use the standardised residuals when checking if the assumption of Normality has been met.

$$r_{i,std} = \frac{r_i}{\hat{\sigma}\sqrt{1-h_{ii}}}$$

If the model fits well, standardised residuals should look similar to a $N(0,1)$ distribution. In addition, large values of the standardised residual indicate potential outlier points.

By the way, $h_{ii}$ is also referred to as the *leverage* of a point. It is a measure of the potential influence of a point (on the parameters, and future predictions). $h_{ii}$ is a value between 0 and 1. For a model with $p$ parameters, the average $h_{ii}$ should be should be $p/n$. We consider points for whom $h_{ii} > 2 \times p/n$ to be high leverage points.

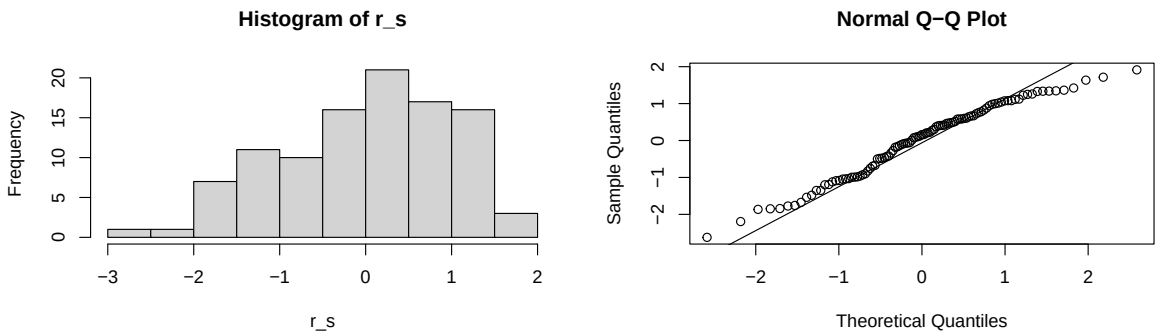
In the literature and in textbooks, you will see mention of residuals, standardised residuals and studentised residuals. While they differ in definitions slightly, they typically yield the same information. Hence we shall stick to standardised residuals for our course.

### 9.7.2 Normality

**Example 9.9** (Concrete Data Normality Check)**.** Let us inspect the residuals from the concrete model for Normality.

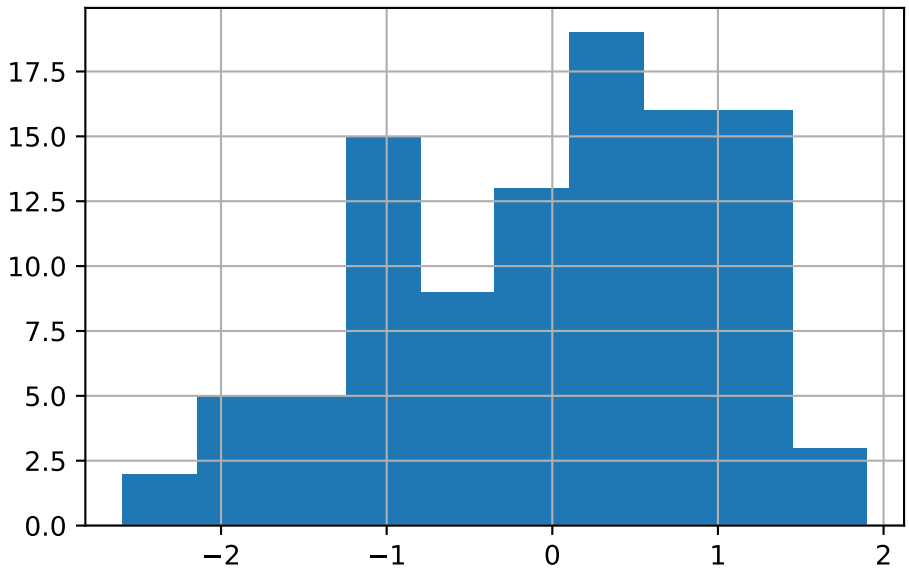
#### R code

```
r_s <- rstandard(lm_flow_water_slag)
hist(r_s)
qqnorm(r_s)
qqline(r_s)
```




#### Python code

```
# Python
r_s = pd.Series(lm_flow_water_slag.resid_pearson)
r_s.hist()
```




While it does appear that we have slightly left-skewed data, the departure from Normality seems to arise mostly from a thinner tail on the right.

```
shapiro.test(r_s)
##
##  Shapiro-Wilk normality test
##
## data:  r_s
## W = 0.97223, p-value = 0.02882
ks.test(r_s, "pnorm")
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  r_s
## D = 0.08211, p-value = 0.491
## alternative hypothesis: two-sided
```

At 5% level, the two Normality tests do not agree on the result either. In any case, please do keep in mind where Normality is needed most: in the hypothesis tests. The estimated model is still valid - it is the best fitting line according to the least-squares criteria.

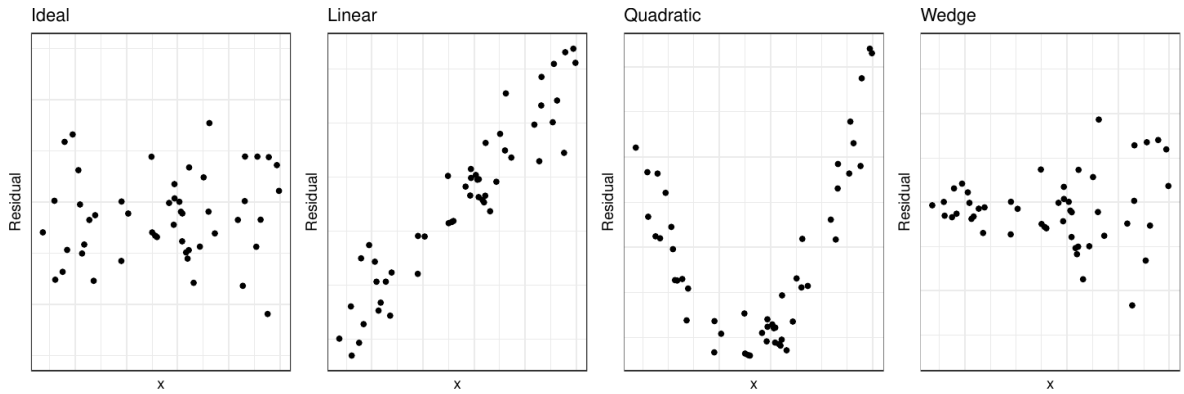
### 9.7.3 Scatterplots

To understand the model fit better, a set of scatterplots are typically made. These are plots of standardised residuals (on the $y$-axis) against

- fitted values
- explanatory variables, one at a time.
- potential variables.

Residuals are meant to contain only the information that our model cannot explain. Hence, if the model is good, the residuals should only contain random noise. There should be no apparent pattern to them. If we find such a pattern in one of the above plots, we would have some clue as to how we could improve the model.

We typically inspect the plots for the following patterns:

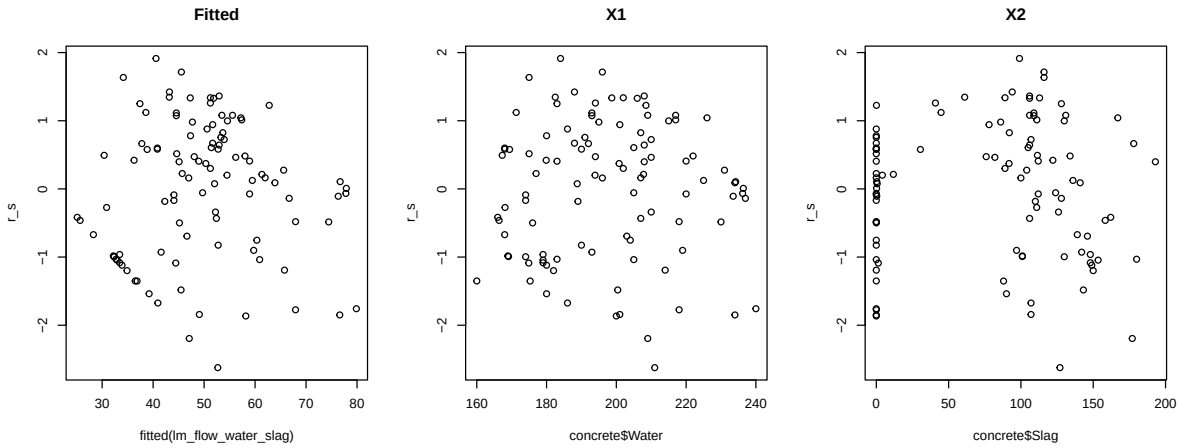



1. A pattern like the one on the extreme left is ideal. Residuals are randomly distributed around zero; there is no pattern or trend in the plot.
2. The second plot is something rarely seen. It would probably appear if we were to plot residuals against a *new* variable that is not currently in the model. If we observe this plot, we should then include this variable in the model.
3. This plot indicates we should include a quadratic term in the model.
4. The wedge shape (or funnel shape) indicates that we do not have homoscedascity. The solution to this is either a transformation of the response, or weighted least squares. You will cover these in your linear models class.

**Example 9.10** (Concrete Data Residual Plots)**.** These are residual plots for the concrete data example.

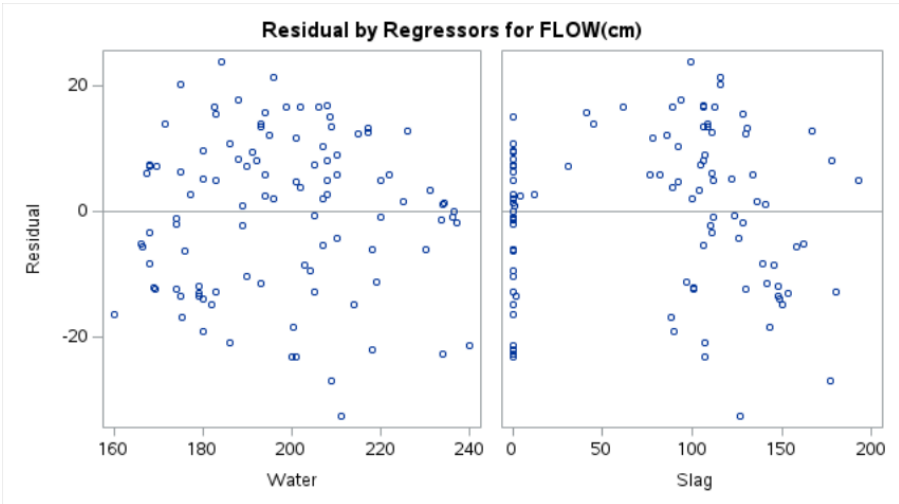
### R code

```
opar <- par(mfrow=c(1,3))
plot(x=fitted(lm_flow_water_slag), r_s, main="Fitted")
plot(x=concrete$Water, r_s, main="X1")
plot(x=concrete$Slag, r_s, main="X2")
```




```
par(opar)
```

### SAS Plots




While the plots of residuals versus explanatory variables look satisfactory, the plot of the residual versus fitted values appears to have funnel shape. Coupled with the observations about the deviations from Normality of the residuals in Example 9.6 (thin right-tail), we might want to try a square transform of the response.

## 9.7.4 Influential Points

The influence of a point on the inference can be judged by how much the inference changes with and without the point. For instance to assess if point $i$ is influential on coefficient $j$:

1. Estimate the model coefficients with all the data points.
2. Leave out the observations $(Y_i, X_i)$ one at a time and re-estimate the model coefficients.
3. Compare the $\beta$'s from step 2 with the original estimate from step 1.

While the above method assesses influence on parameter estimates, Cook's distance performs a similar iteration to assess the influence on the fitted values. Cook's distance values greater than 1 indicate possibly influential points.

**Example 9.11** (Concrete Data Influential Points)**.** To inspect influential points for the concrete data, we can use this code.

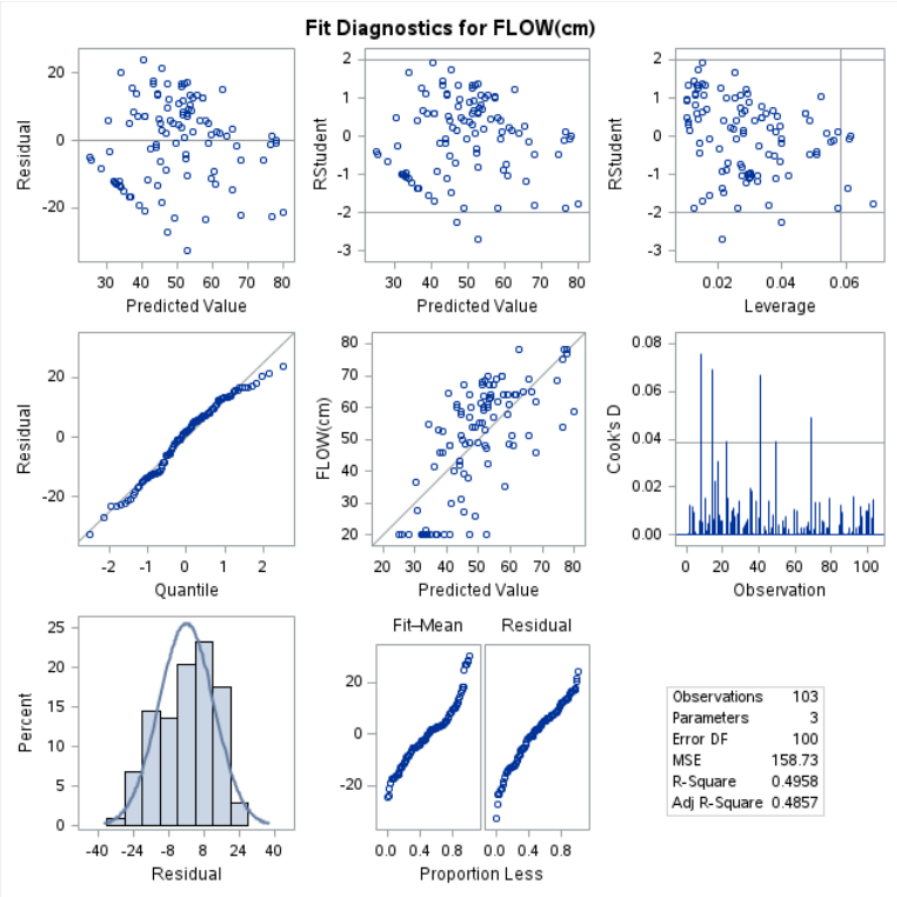
### R code

```
infl <- influence.measures(lm_flow_water_slag)
summary(infl)
```

```
Potentially influential observations of
     lm(formula = FLOW.cm. ~ Water + Slag, data = concrete) :

   dfb.1_ dfb.Watr dfb.Slag dffit cov.r   cook.d hat
60  0.03  -0.03    -0.01    -0.03  1.09_*  0.00   0.06
69  0.19  -0.19    -0.22    -0.40  0.85_*  0.05   0.02
83  0.02  -0.02     0.01    -0.03  1.09_*  0.00   0.06
88 -0.02   0.02    -0.01     0.03  1.09_*  0.00   0.06
93  0.00   0.00     0.00     0.00  1.10_*  0.00   0.06
98  0.01  -0.01     0.01    -0.02  1.10_*  0.00   0.06
```

The set of 6 points above appear to be influencing the covariance matrix of the parameter estimates greatly. To proceed, we would typically leave these observations out one-at-a-time to study the impact on our eventual decision.

**SAS Output**




## 9.8 Further Reading

The topic of linear regression is vast. It is an extremely well-established technique with numerous variations for a multitude of scenarios. Even a single course on it (ST3131) will not have sufficient time to cover all of it's capabilities. Among topics that will be useful in your career are :

- Generalised additive models, which allow the use of piecewise polynomials for flexible modeling of non-linear functions.
- Generalised linear models, for modeling non-numeric response.
- Generalised least squares, to handle correlated observations, and so on.
- Principal component regression, to handle the issue of multicollinearity (correlated predictors).

The textbooks Draper and Smith (1998) and James et al. (2013) are excellent reference books for this topic.

## 9.9 References

### 9.9.1 Website References


3. Stats models documentation
4. Diagnostics
5. On residual plots

# 10 Simulation

## 10.1 Introduction

The objective of any simulation study is to estimate an expectation $E(X)$. Simulation studies involve the use of a computer to generate independent copies of the random variable of interest $X$. Here are a couple of examples where simulation studies would be applicable.

**Example 10.1** (Insurance Claims)**.** Before the financial year begins, an insurance company has to decide how much cash to keep, in order to pay out the claims for that year. Suppose that claims are independent of each other and are distributed as $Exp(1/200)$[1] dollars. Also suppose that the number of claims in a year is a Poisson random variable with mean 8.2.

An actuary has been asked to determine the size of the reserve fund that should be set up, and he recommends \$12,000. We might consider answering the following question using simulation:

- What is the probability that the total claims will exceed the reserve fund?

If we let $Y$ be the random variable representing the total sum of claims, we are interested in estimating $P(Y > 12000)$. Since probabilities are expectations, we can use simulation to estimate this value.

Here is a slightly more sophisticated example.

**Example 10.2** (Sandwich Shop Closing Time)**.** Suppose that you run a sandwich shop, which is open from 9am till 5pm. Your philosophy has always been to serve every customer who has entered before 5pm, even if that requires you to stay back until they have been served. You would like to estimate the mean amount of overtime you have to work.

If you are willing to assume that the inter-arrival times of customers is $Exp(3)$ hours, then it is possible to simulate this process to estimate the mean time that you would have to remain open, beyond 5pm.

The two examples above are known as Discrete Event Simulations. In our course, we will not get to working with such scenarios. However, we will try to understand and familiarise ourselves with the basic building blocks of simulation studies. For more knowledge, do enrol yourself in ST3247!

The basic steps in a simulation study are:

1. Identify the random variable of interest and write a program to simulate it.
2. Generate an iid sample $X_1, X_2, \dots, X_n$ using this program.
3. Estimate $E(X)$ using $\bar{X}$.

This is sometimes referred to as Monte Carlo Simulation. Before proceeding, let us refresh our knowledge of the properties of the sample mean.

[1] $f_X(x) = \frac{1}{200}\exp(-x/200),\ x > 0$

## 10.2 Theory

There are two important theorems that simulation studies rely on. The first is the Strong Law of Large Numbers (SLLN).

**Theorem 10.1** (Strong Law of Large Numbers)**.** *If $X_1, X_2, \dots, X_n$ are independent and identically distributed with $E(X) < \infty$, then*

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \to E(X) \quad \textit{with probability 1.}$$

In the simulation context, it means that as we generate more and more samples (i.e. increase $n$), our sample mean $\bar{X}$ converges to the desired value $E(X)$, *no matter what the distribution of $X$ is.*

The second theorem that aids us is the Central Limit Theorem (CLT).

**Theorem 10.2** (Central Limit Theorem)**.** *Let $X_1, X_2, \dots, X_n$ be i.i.d., and suppose that*

- $-\infty < E(X_1) = \mu < \infty$.
- $Var(X_1) = \sigma^2 < \infty$.

*Then*

$$\frac{\sqrt{n}(\bar{X} - \mu)}{\sigma} \Rightarrow N(0, 1)$$

*where $\Rightarrow$ denotes convergence in distribution.*

This is sometimes informally interpreted to mean that when $n$ is large, $\bar{X}$ is approximately Normal with mean $\mu$ and variance $\sigma^2/n$. In the simulation context, we can use this theorem to obtain a confidence interval for the expectation that we are estimating.

Also take note of the following properties of the sample mean and variance:

**Theorem 10.3** (Sample Estimates)**.** *It can be shown that both the sample mean and sample standard deviation are unbiased estimators.*

$$E(\bar{X}) = E(X), \quad E(s^2) = \sigma^2$$

*where $s^2 = \frac{\sum(X_i - \bar{X})^2}{n-1}$.*

To obtain a $(1-\alpha)100$ confidence interval for $\mu$, we use the following formula, from the CLT:

$$\bar{X} \pm z_{1-\alpha/2}\frac{s}{\sqrt{n}}$$

When our goal is to estimate a probability $p$, we have to introduce a corresponding indicator variable $X$ such that

$$X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}$$

In this case, the formula for the CI becomes

$$\bar{X} \pm z_{1-\alpha/2}\sqrt{\frac{\bar{X}(1-\bar{X})}{n}}$$

## 10.3 Generating Random Variables in R and Python

Both R and Python contain built-in routines for generating random variables from common "named" distributions, e.g. Normal, Uniform, Gamma, etc. All software that can generate random variables utilise Pseudo-Random Number Generators (PRNG). These are routines that generate sequences of deterministic numbers with very very long cycles. However, since they pass several tests of randomness, they can be treated as truly random variables for all intents and purposes.

In both software, we can set the "seed". This initialises the random number generator. When we reset the seed, we can reproduce the stream of random variables exactly. This feature exists:

1. For debugging purposes,
2. To allow us to study the sensitivity of our results to the seed.

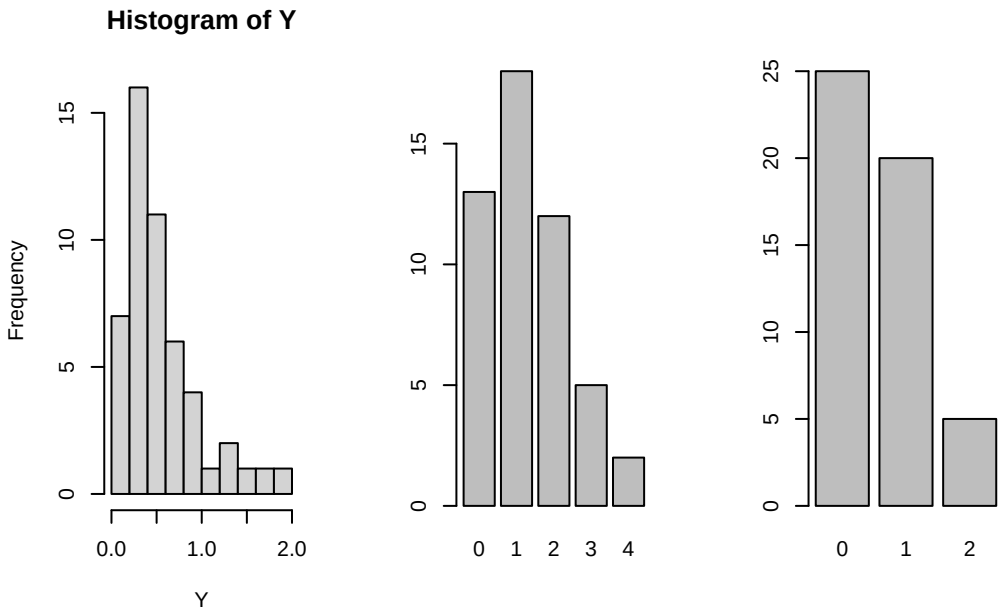
**Example 10.3** (Random Variable Generation)**.**

### R code

In R, all the functions for generating random variables begin with `r` (for random). Here are a few such functions:

| Function name | Random Variable |
|---|---|
| `rnorm` | Normal |
| `runif` | Uniform |
| `rgamma` | Gamma |
| `rpois` | Poisson |
| `rbinom` | Binomial |

```
#R
set.seed(2137)

opar <- par(mfrow=c(1,3))
Y <- rgamma(50, 2, 3)
hist(Y)

W <- rpois(50, 1.3) # 50 obs from Pois(1.3)
barplot(table(W))

Z <- rbinom(50, size=2, 0.3) # 50 obs from Binom(2, 0.3)
barplot(table(Z))
```




```
par(opar)
```

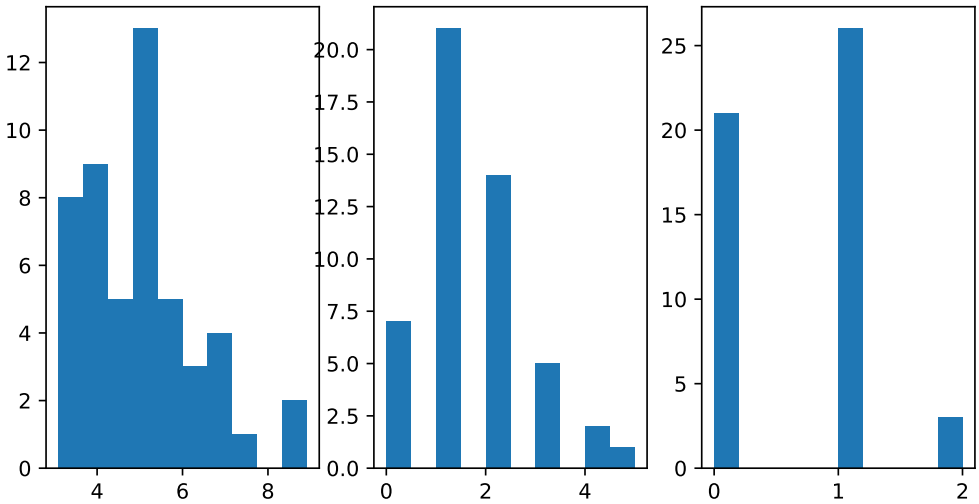
### Python code

In Python, we can generate random variables using `numpy` and/or `scipy`. In our course, we shall use the `scipy` routines because its implementation is closer in spirit to R.

| Function name | Random Variable |
|---|---|
| `norm` | Normal |
| `uniform` | Uniform |
| `gamma` | Gamma |
| `poisson` | Poisson |
| `binom` | Binomial |

```
#Python
import numpy as np
import pandas as pd
from scipy.stats import binom, gamma, norm, poisson
from scipy import stats
import matplotlib.pyplot as plt

rng = np.random.default_rng(2137)
fig, ax = plt.subplots(1, 3, figsize=(8,4))

ax1 = plt.subplot(131)
r = gamma.rvs(2, 3, size=50, random_state=rng)
ax1.hist(r);
```

```
ax1 = plt.subplot(132)
r = poisson.rvs(1.3, size=50, random_state=rng)
ax1.hist(r);

ax1 = plt.subplot(133)
r = binom.rvs(2, 0.3, size=50, random_state=rng)
ax1.hist(r);
```




## 10.4 Monte-Carlo Integration

Suppose we wish to evaluate

$$\int_{-\infty}^{\infty} h(x)f(x)\mathrm{d}x$$

where $f(x)$ is a pdf. The integral above is in fact equal to $E(h(X))$, where $X \sim f$. Hence we can use everything we have discussed so far, to evaluate the expression using simulation! Everything depends on:

1. Being able to introduce a pdf to the integral
2. Being to able to simulate from that pdf.

> **!** Important
>
> It is critical that the support of the pdf is the same as the range of integration.

**Example 10.4** (Monte Carlo Integration over (0,1))**.** Suppose Suppose we wish to evaluate

$$\theta = \int_0^1 e^{2x} dx = \int_0^1 e^{2x} \cdot 1 \; dx$$

We can identify that this is equal to $E(h(X))$ where

- $X \sim Unif(0,1)$.
- $h(X) = e^{2x}$

Thus we can follow this pseudo-code:

1. Generate $X_1, X_2, \dots, X_n \sim Unif(0,1)$.
2. Estimate the integral using

$$\frac{1}{n}\sum_{i=1}^{n} e^{2X_i}$$

In this simple case, we can in fact work out analytically that the integral is equal to

$$\frac{1}{2}(e^2 - 1) = 3.195$$

#### R code

```
set.seed(2138)
X <- runif(50000, 0, 1)
hX <- exp(2*X)
(mc_est <- mean(hX))
```

```
[1] 3.185626
```

#### Python code

```
from scipy.stats import uniform

X = uniform.rvs(0,1, size=50000, random_state=rng)
hX = np.exp(2*X)
hX.mean()
```

```
np.float64(3.2090955591090906)
```

## 10.5 Simulation Studies

In this section, we shall touch on how we can use simulation in some scenarios that are closer to the real world.

### 10.5.1 Confidence Intervals

The usual 95% confidence interval for a mean is given by

$$\bar{X} \pm t_{0.025} s/\sqrt{n}$$

where $t_{0.025}$ is the 0.025 quantile of the t-distribution with $n-1$ degrees. The resulting interval should contain the true mean in 95% of the experiments. However, it is derived under the assumption that the data is Normally distributed. Let us see if it still works if the data is from an asymmetric distribution: $Pois(0.5)$.

**Example 10.5** (Coverage of Confidence Interval)**.**

#### R code

```
# R
set.seed(2139)
output_vec <- rep(0, length=100)
n <- 20
lambda <- 0.5
for(i in 1:length(output_vec)) {
  X <- rpois(n, .5)
  Xbar <- mean(X)
  s <- sd(X)
  t <- qt(0.975, n-1)
  CI <- c(Xbar - t*s/sqrt(n), Xbar + t*s/sqrt(n))
  if(CI[1] < lambda & CI[2] > lambda) {
    output_vec[i] <- 1
  }
}
mean(output_vec)
```

```
[1] 0.92
```

#### Python code

```
rng = np.random.default_rng(2137)
output_vec = np.zeros(100)
n = 20
lambda_ = 0.5
for i in range(100):
    X = poisson.rvs(0.5, size=n, random_state=rng)
    Xbar = X.mean()
    s = X.std()
    t = norm.ppf(0.975)
    CI = [Xbar - t*s/np.sqrt(n), Xbar + t*s/np.sqrt(n)]
    if CI[0] < lambda_ and CI[1] > lambda_:
        output_vec[i] = 1
output_vec.mean()
```

```
np.float64(0.86)
```

### 10.5.2 Type I Error

Consider the independent two-sample $t$-test that we introduced in topic 7. The formal set-up also includes the assumption that our data arises from a Normal distribution. According to the theory of the $t$-test, if both groups have the same mean, we should *falsely* reject the null hypothesis 10% of the time if we perform it at 10% significance level. Let us assess if this is what actually happens.

**Example 10.6** (T-test Type I Error)**.**

#### R code

```r
generate_one_test <- function(n=100) {
  X <- rnorm(n)
  Y <- rnorm(n)
  t_test <- t.test(X, Y,var.equal = TRUE)
  # extract the p-value from the t_test
  if(t_test$p.value < 0.10)
    return(1L)
  else
    return(0L)
}

set.seed(11)
output_vec <- vapply(1:2000,
                     function(x) generate_one_test(),
                     1L)
mean(output_vec)
```

```
[1] 0.108
```

#### Python code

```python
def generate_one_test(n=100):
    X = norm.rvs(0, 1, size=n, random_state=rng)
    Y = norm.rvs(0, 1, size=n, random_state=rng)
    t_test = stats.ttest_ind(X, Y, equal_var=True)
    if t_test.pvalue < 0.10:
        return 1
    else:
        return 0
output_vec = np.array([generate_one_test() for j in range(2000)])
output_vec.mean()
```

```
np.float64(0.0955)
```

### 10.5.3 Newspaper Inventory

**Example 10.7.** Suppose that daily demand for newspaper is approximately gamma distributed, with mean 10,000 and variance 1,000,000. The newspaper prints and distributes 11,000 copies each day. The profit on each newspaper sold is \$1, and the loss on each unsold newspaper is 0.25. Formally, the daily profit function h is

$$h(X) = \begin{cases} 11000 & \text{if } X \geq 11000 \\ \lfloor X \rfloor + (11000 - \lfloor X \rfloor)(-0.25) & \text{if } X < 11000 \end{cases}$$

where $X$ represents the daily demand. Let us estimate the expected daily profit using simulation.

#### R code

```
# R code to estimate the expected daily profit
set.seed(2141)
n <- 10000
X <- rgamma(n, 100, rate=1/100)
hX <- ifelse(X >= 11000, 11000, floor(X) + (11000 - floor(X)) * (-0.25))
#mean(hX)

# 90% CI for the mean
s <- sd(hX)
q1 <- qnorm(0.95)
CI <- c(mean(hX) - q1*s/sqrt(n), mean(hX) + q1*s/sqrt(n))
cat("The 90% CI for the mean is (", format(CI[1], digits=2, nsmall=2), ", ",
    format(CI[2], digits=2, nsmall=2), ").\n", sep="")
```

```
The 90% CI for the mean is (9639.10, 9673.69).
```

#### Python code

```
n = 10000
X = gamma.rvs(100, scale=100, size=n, random_state=rng)
hX = np.where(X >= 11000, 11000, np.floor(X) + (11000 - np.floor(X)) * (-0.25))

# 90% CI for the mean
Xbar = hX.mean()
s = hX.std()
t = norm.ppf(0.95)
CI = [Xbar - t*s/np.sqrt(n), Xbar + t*s/np.sqrt(n)]
print(f"The 90% CI for the mean is ({CI[0]: .3f}, {CI[1]: .3f}).")
```

```
The 90% CI for the mean is ( 9623.651,  9658.579).
```

## 10.6 Resampling Methods

The next section introduces two techniques that are based on resampling the data.

### 10.6.1 Permutation Test

Consider the two-sample t-test that we introduced in topic 06. This parametric test requires us to check if the data from the two groups came from Normal distributions. The non-parametric version requires each group to have at least 10 observations. What if our data satisfies neither criteria?

The permutation test is applicable in such a situation. It makes no distributional assumptions whatsoever on the data. Here is pseudo-code for how it works:

1. Compute the difference in group means. This observed difference is the test statistic.
2. Treating the observed values as fixed, combine the two vectors of observations.
3. Permute the observations, and then re-assign them to the two groups.
4. Compute the difference between the group means.
5. Repeat steps 2 - 4 multiple times (order of 1000).

The $p$-value for the null hypothesis can be computed by computing the proportion of simulated statistics that were larger in absolute value than the observed one.

**Example 10.8** (Abalone Data)**.** In the abalone dataset, this was the output from the t-test:

```
abl <- read.csv("data/abalone_sub.csv")
x <- abl$viscera[abl$gender == "M"]
y <- abl$viscera[abl$gender == "F"]

t.test(x, y, var.equal=TRUE)
```

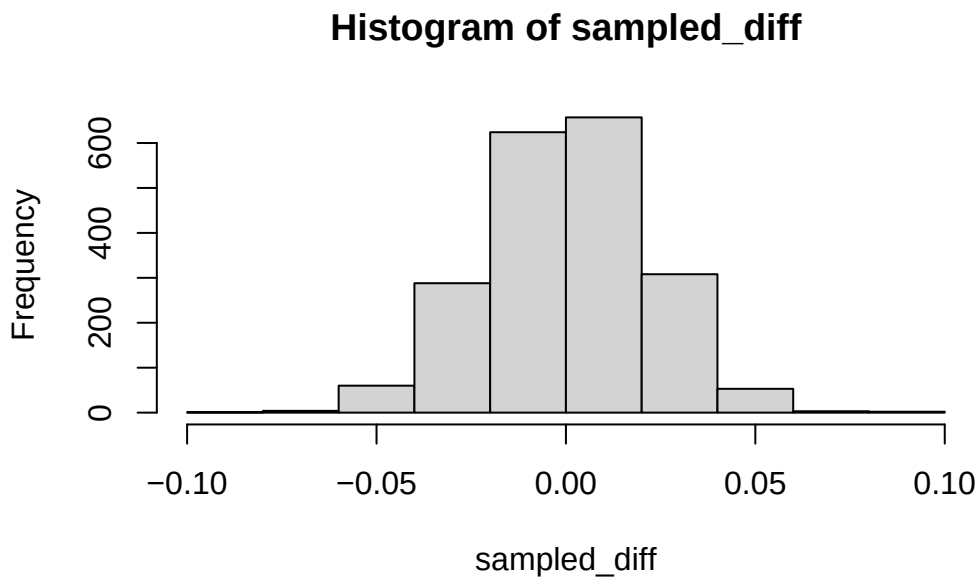
```
    Two Sample t-test

data:  x and y
t = 0.91008, df = 98, p-value = 0.365
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.02336287  0.06294287
sample estimates:
mean of x mean of y
  0.30220   0.28241
```

This would be the procedure for a permutation test:

```
d1 <- mean(x)  - mean(y)
print(d1)
```

```
[1] 0.01979
```

```r
generate_one_perm <- function(x, y) {
  n1 <- length(x)
  n2 <- length(y)
  xy <- c(x,y)
  xy_sample <- sample(xy)
  d1 <- mean(xy_sample[1:n1]) - mean(xy_sample[-(1:n1)])
  d1
}
sampled_diff <- replicate(2000, generate_one_perm(x,y))
hist(sampled_diff)
```




```r
(p_val <- 2*mean(sampled_diff > d1))
```

```
[1] 0.369
```

### 10.6.2 Bootstrapping

The video on Canvas provides a very brief introduction to bootstrapping. One of the greatest benefits of the bootstrap is the ability to provide confidence intervals for estimators for which we may not have the know-how to derive analytic or asymptotic results.

Consider obtaining a confidence interval for the trimmed mean, that we encountered in Section 5.1.

**Example 10.9.** This is how we can use bootstrapping to obtain a confidence interval for a trimmed mean.

```r
library(MASS)

mean(chem)
```

```
[1] 4.280417
```

```r
t.test(chem)
```

```
    One Sample t-test

data:  chem
t = 3.9585, df = 23, p-value = 0.0006236
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.043523 6.517311
sample estimates:
mean of x
 4.280417
```

```r
## [1] 4.280417
```

Notice how the CI from the non-robust technique is very wide.

```r
library(boot)

stat_fn <- function(d, i) {
  b <- mean(d[i], trim=0.1)
  b
}
boot_out <- boot(chem, stat_fn, R = 1999, stype="i")
# Returns two types of bootstrap intervals:
boot.ci(boot.out = boot_out, type=c("perc", "bca"))
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1999 bootstrap replicates

CALL :
boot.ci(boot.out = boot_out, type = c("perc", "bca"))

Intervals :
Level     Percentile            BCa
95%   ( 2.955,  4.700 )   ( 2.970,  4.773 )
Calculations and Intervals on Original Scale
```

The boot function requires a function (`stat_fn`) that computes the statistic from the bootstrapped sample. Note that the intervals returned from the trimmed mean are much narrower.

## 10.7 Summary

In this chapter, we have seen how simulation studies can be used to estimate expectations. Although we have restricted ourselves to very straightforward examples, the same principles can be applied to more complex scenarios.

In particular, there are three types of simulation models widely used to model complex systems:

1. Agent-based models. These are regularly used to model the interactions of agents in a system, e.g. humans.
2. Discrete Event Simulations. These are used to model systems where events occur at discrete points in time. Typically, these are systems where there is an arrival of entities, a service, and a departure.
3. Process modeling. This approach is used to model the flow of entities through a system. They are sometimes also referred to as compartment models.

Please refer to the sections below for more information. For our course, please be familiar with the basic building blocks of simulation studies:

1. Generate iid samples, and then
2. Compute the sample mean, and the CI for the true mean.

## 10.8 References

### 10.8.1 Website References

1. Some GUI-based software for simulation:
   - Anylogic A very very powerful software for agent-based modeling.
   - Arena A discrete event simulator, with a free academic license.
   - Netlogo An open source software for agent-based modeling.
2. Python software:
   - simpy for discrete event simulations.
   - mesa for agent-based modeling.
3. R software:
   - simmer for discrete event simulation
4. scipy documentation
5. Introduction to Bootstrapping

# Academic References


Agresti, Alan. 2012. *Categorical Data Analysis*. Vol. 792. John Wiley & Sons.

Cortez, Paulo, and Alice Maria Gonçalves Silva. 2008. "Using Data Mining to Predict Secondary School Student Performance."

Draper, Norman R, and Harry Smith. 1998. *Applied Regression Analysis*. Vol. 326. John Wiley & Sons.

Ekstrøm, Claus Thorn, and Helle Sørensen. 2015. "Statistical Data Analysis for the Life Sciences." CRC Press, London.

Fanaee-T, Hadi, and Joao Gama. 2013. "Event Labeling Combining Ensemble Detectors and Background Knowledge." *Progress in Artificial Intelligence*, 1–15. https://doi.org/10.1007/s13748-013-0040-3.

Huber, Peter J. 1992. "Robust Estimation of a Location Parameter." In *Breakthroughs in Statistics: Methodology and Distribution*, 492–518. Springer.

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

R Core Team. 2025. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rosner, Bernard. 2015. *Fundamentals of Biostatistics*. Cengage learning.

Venables, William N, and Brian D Ripley. 2013. *Modern Applied Statistics with s-PLUS*. Springer Science & Business Media.

Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science*. O'Reilly Media, Inc.

Wilcox, R, and R. 2011. *Introduction to Robust Estimation and Hypothesis Testing*. Academic press.

Yeh, I-Cheng. 2007. "Modeling Slump Flow of Concrete Using Second-Order Regressions and Artificial Neural Networks." *Cement and Concrete Composites* 29 (6): 474–80.

# Index


Abalone
- 2-sample t-test, 112
- non-parametric two-sample test, 122
- Normality checks, 116
- permutation test, 185

Bike rentals
- boxplots, 103
- chi-squared test, 105
- description, 99
- histograms, 102
- interaction term, 165
- numerical summaries, 101
- QQ plots, 105
- registered vs. casual, work day, 162
- regression F-test, 153
- scatterplot, 101

Chest pain
- chi-squared test, 73
- description, 65
- expected counts, 72
- odds ratio, 79

Claritin
- bar chart, 67
- Fisher's exact test, 75

Concrete slump
- description, 57
- flow vs. water, 150
- flow vs. water, slag, 159
- influential points, 173
- predicted means, 155
- QQ plots, 57
- residual Normality, 169
- residual plots, 171
- scatterplots and correlation, 61

Copper
- bootstrap CI, 186
- histogram, outlier, 84
- robust location, 92
- robust scale, 93

Heart failure
- conditional density plot, 70

Heifers
- comparing two groups, 135
- contrasts, 137
- description, 127
- F-test, 131
- Kruskal-Wallis test, 142

Job satisfaction
- ordinal association, 80

Political association
- bar chart, 66
- chi-squared test, 77
- mosaic plot, 68

Self-awareness
- histogram, outlier, 85

Student performance
- boxplots, 54
- density plots, 52
- description, 44
- histograms, 50
- Kendall's tau, 105
- numerical summaries, 45

Treadmill
- non-parametric paired test, 124
- paired t-test, 119