Introduction to R

SCSRU Workshop

Statistical Consulting and Survey Research Unit
University of Waterloo

2024-03-01

RStudio

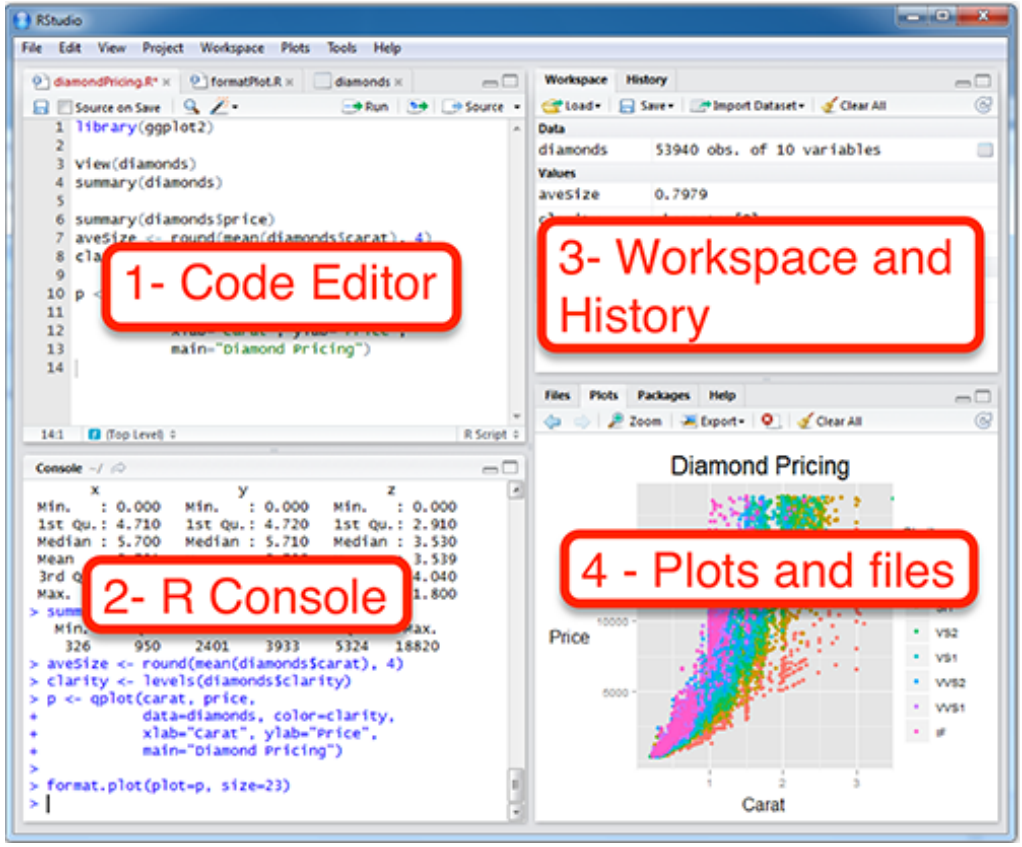


Figure 1: The RStudio Interface

- The four panels can be positioned in different locations based on your preferences, but for this session, we will use the default layout as shown above.

1. R as a calculator

- In its simplest form, R can be used as a calculator.
- In the console area, type:

```
1+2
```

you will see the following being "returned" in the console area

```
[1] 3
```

- Subtraction can be done in a similar way

```
5-10
```

```
[1] -5
```

- Other basic operations such as multiplication, division and powers include:

```
9*26
```

```
[1] 234
```

```
100/7.5
```

```
[1] 13.33333
```

```
2^3
```

```
[1] 8
```

Some basic operations involves built-in functions in R. For example,

- $\sqrt{25}$

```
sqrt(25)
```

```
[1] 5
```

- $\log_{10}(10)$

```
log(10, base=10)
```

```
[1] 1
```

- $\ln(10)$

```
log(10)
```

```
[1] 2.302585
```

Practice 1.1

Suppose we are interested to evaluate the following ratio

$$\frac{e^1}{1 - e^1}$$

Did you see?

```
[1] -1.581977
```

Practice 1.2

How about the following:

$$\frac{e^{1-\pi}}{1-e^{1-\pi}}$$

Did you see?

```
[1] 0.1331029
```

2. Variables

- The number π is recognized in R as "pi". We call "pi" a variable in R. Other default variable is the imaginary number, i.e $\sqrt{-1}$, which is recorded as 'i' ' in R.
- Variables are useful for when they need to be used repeatedly or to be recalled in the future.
- Suppose we are interested to evaluate

$$\frac{e^{1-9.2315468}}{1-e^{1-9.2315468}},$$

we can store the repeated value 9.2315468 as a variable before performing the calculation.

To store the value as the variable x,

```
x <- 9.2315468
```

What do you notice?

- In the Console panel, nothing is returned.
- In the Workspace and History panel, x appears together with the value it represents. This shows that your current workspace recognizes x as 9.2315468.
- Try typing x in the console.

Back to our problem

We wanted to evaluate

$$\frac{e^{1-9.2315468}}{1-e^{1-9.2315468}},$$

Since $x = 9.2315468$ is stored in our work environment, we can now type

```
exp(1-x)/(1-exp(1-x))
```

```
[1] 0.0002661952
```

Practice 2.1

Can you simplify the code further?

```
exp(1-x)/(1-exp(1-x))
```

3. Getting help

- All users will need help to a certain degree at different stages of their work.
- Before getting help from other users, it is generally a good idea to try to solve the problem by yourself.

3.1 R documentation

- R has an extensive documentation and resources for help.
- For example, to find out how to use the function `round()`, type

```
?round
```

- The bottom right panel will show a description of the function and examples of how to use it.
- This help feature is particularly useful for when you vaguely remember how to use the function.

Practice 3.1

Evaluate the following formula

$$99 + 1.5 \times (\frac{10}{99} - 99 \times tryMe),$$

where $tryMe = 0.456789$. Use the function `round()` to round your output to 2 decimal places.

Note that R is case sensitive.

3.2 Online resources

- There are a lot of basic functions or default variables that have not been mentioned so far.
- When working with R, we often encounter situations in which we need to use an unknown or unfamiliar functions. We often rely on online search engines to find those functions. Some reliable sources are Stack Overflow and R-bloggers.
- Throughout this workshop, we encourage you to explore the online resources. Doing so will make you more confident to work independently in the future.
- We will encounter situations where different set of codes can produce the exact same output. There are many criteria that can determine the superiority of a set of code. However, we recommend selecting a set of codes that you can understand best.

4. Vectors

In the real world, we often encounter a sequence of numbers. For example,

- the height of 10 students,
- the grades of the ECON 101 students in the Fall term,
- the age of the attendees,
- and many more.

In R, the sequence of numbers can be recorded as vectors.

4.1 Creating vectors

Suppose the age of the attendees in this workshop are:

$$18, 21, 19, 20, 21$$

We can create a vector for our record

```
age <- c(18, 21, 19, 20, 21)
```

What do you see?

- In the environment panel?
- After typing age in the console panel?

4.2 The use of vectors

- Vectors may not appear to be be useful for many since most of the required functions are ready for use.
- For those intending to create your own R functions, it is important to understand how to create and manipulate vectors.

5. Code editor

- Do you still remember all the codes created? Do you want to save the codes for future references?
- The code editor allows us to write and save all the written codes.
- The lines of codes in the code editor are not processed by R, until they are inputted into the console.

5.1 Running codes in the code editor

There are many ways to do so:

1. Highlight all the codes you want R to process, click `Run` on the top right corner of the Console Editor.
2. For Windows users, you can run the highlighted codes by pressing `Ctrl` + `Enter`. For Mac users, you can do the same with `Command` + `Enter`.
3. If you only want to run one line of code, you can perform (2) without highlighting the entire line of codes. Placing the cursor at the line of interest is sufficient.

Practice 5.1

Consider inputting some of the codes we have used so far in the code editor and practice running the lines of codes using the short cuts.

From hereon, we recommend to type the codes in the code editor and then run the codes. This way, you have a copy of what you did in the workshop today for your future reference.

5.2 Saving the Environment

When quitting R or RStudio, we can choose to save either the Environment or the History that we were working with in the files called `.RData` and `.RHistory` respectively. When we open the R code file next time, the two files will be automatically loaded.

- We do not recommend saving the Environment. Start in a clean environment.
- If you want to save the Environment, we recommend to do so using the function `save.image()` rather than using the default files `.RData`.
- If we only want to save certain values, we can use the function `save()` and then load the saved Environment later using the `load()` function.

6. Good coding practices

- Comment your code. R does not process anything behind #.

```
# I am trying to like R!!!!
```

- Use comments to mark off sections of code. Type the following in the code editor:

```
# ----------------
# Try Me!
# ----------------
```

What do you notice next to the line number?

6.1 Our recommendations

- Start each program with a description of what it does.
- Load all required packages at the beginning.
- Consider your choice of working directory.
- Use comments to mark off sections of code.
- Put function definitions at the top of your file, or in a separate file if there are many.
- Name and style code consistently.
- Break code into small, discrete pieces.
- Factor out common operations rather than repeating them.
- Keep all of the source files for a project in one directory and use relative paths to access them.
- Have someone else review your code.
- Use version control.

7. Importing and exporting data

- In the real world, data are recorded in different ways. The common formats are CSV, TXT and XLS.
- R can import the data into its environment with functions such as: `read.csv()` and `read.table()`.

7.1 Setting working directory

To start, it is important to inform R the directory that the data file is stored. For Mac/Windows users of RStudio, choose Session > Set Working Directory > Choose Directory.

An alternative is to set the working directory from the Plots and Files panel (bottom right panel).

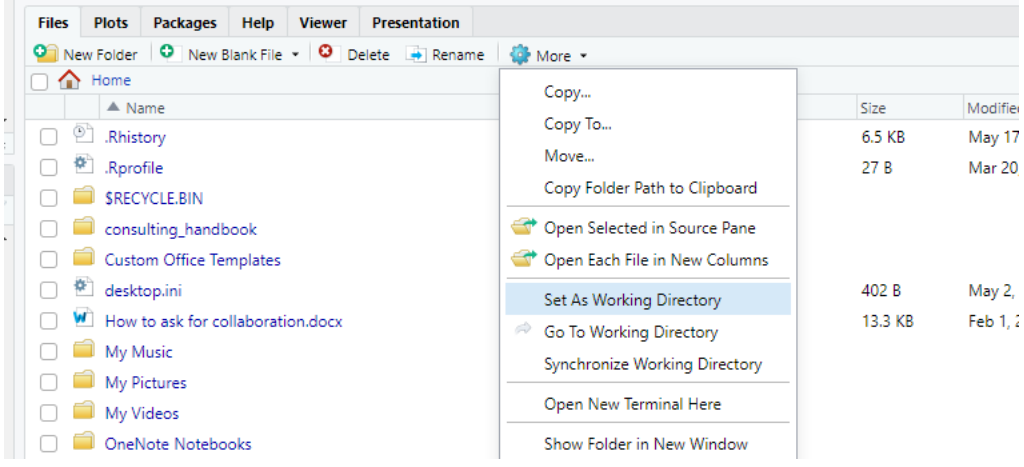


Figure 2: Setting working directory

Using command lines to set working directory

The function `setwd()` can also be used to set the working directory if the directory string is available. For example,

```
setwd("D:/")
```

will set the working directory to "D:/".

A common question that you may have about working directory is, "Where am I?"

The address is shown at the top of Console.

Practice 7.1

- Did you download the data file?
- Do you remember where you saved the file?

If you have not downloaded nor saved them, please do so now.

When you are done,

- set the working directory to the folder where the file was saved.

7.2 Importing data set

- In reality, data are recorded in different formats such as Excel spreadsheet and Comma Separated Values (CSV).
- Each row of a data files is an observation while each column is a variable.
- Data are imported into the R environment and stored as a data frame object.
- In this workshop, we will focus on the data set `caliRain.csv`.

Practice 7.2

Import `caliRain.csv` into the R environment and save it as data frame called`rain_df`.

```
rain_df <- read.csv("caliRain.csv")
```

7.3 A first look at the data set

- It is important to take a look at the data set imported into the environment before performing the analysis.

```
head(rain_df)
```

```
                 STATION PRECIP ALTITUDE LATITUDE DISTANCE SHADOW
1 Eureka                   39.57       43     40.8        1      1
2 RedBluff                 23.27      341     40.2       97      2
3 Thermal                  18.20     4152     33.8       70      2
4 FortBragg                37.48       74     39.4        1      1
5 SodaSprings              49.26     6752     39.3      150      1
6 SanFrancisco             21.82       52     37.8        5      1
```

About the `caliRain.csv` data

The file contains daily rainfall recorded at numerous meteorological stations monitored by the state of California. The variables recorded are:

- STATION: Name of the station
- PRECIP: precipitation (inches)
- ALTITUDE: altitude (feet)
- LATITUDE: latitude (degree)
- DISTANCE: distance to the Pacific Ocean (miles)
- SHADOW: slope face (1: Westward, 2: Leeward)

The variables `STATION` and `SHADOW` are categorical variables, whereas the remaining are continuous variables.

7.4 Accessing data frame

To get an overview of the data, we usually take a look at the dimension data frame to get an overview.

- dim(rain_df): shows the number of rows and columns
- nrow(rain_df): shows the number of rows
- ncol(rain_df): shows the number of columns

To view the spreadsheet in R, try

```
View(rain_df)
```

Individual column/row

Oftentimes, we are interested to access an individual column (or variable) within the data frame. For example, the precipitation variable. There are two ways to do so:

```
rain_df$PRECIP
```

```
 [1] 39.57 23.27 18.20 37.48 49.26 21.82 18.07 14.17 42.63 13.85  9.44 19.33
[13] 15.67  6.00  5.73 47.82 17.95 18.20 10.03  4.63 14.74 15.02 12.36  8.26
[25]  4.05  9.94  4.25  1.66 74.87 15.95
```

```
rain_df[,2]
```

```
 [1] 39.57 23.27 18.20 37.48 49.26 21.82 18.07 14.17 42.63 13.85  9.44 19.33
[13] 15.67  6.00  5.73 47.82 17.95 18.20 10.03  4.63 14.74 15.02 12.36  8.26
[25]  4.05  9.94  4.25  1.66 74.87 15.95
```

Similarly, there are times we want to investigate a particular row (or observation). Suppose we are interested in the 10th observation,

```
rain_df[10,]
```

```
                  STATION PRECIP ALTITUDE LATITUDE DISTANCE SHADOW
10 Salinas                  13.85       74     36.7       12      2
```

Suppose we are interested in the precipitation of the 5th observation,

```
rain_df$PRECIP[5]
```

```
[1] 49.26
```

```
rain_df[5,2]
```

```
[1] 49.26
```

Accessing a random variable, an observation or a specific value coming from an observation are all useful for data manipulation and transformation purpose.

7.5 Data manipulation

Sometimes, we want to make changes to the data frame such as

- Make changes to existing records
- Add new observations or variables
- Remove outliers from the data set

This process is sometimes known as data manipulation. Data manipulation is essential to ensure that the data is cleaned and ready for further processing and analysis. This process may be tedious and boring, but reliable results depend on the data.

How does R understand the data?

```
str(rain_df)
```

```
'data.frame':   30 obs. of  6 variables:
 $ STATION : chr  "Eureka                  " "RedBluff                 " "Thermal                  " "Fo
 $ PRECIP  : num  39.6 23.3 18.2 37.5 49.3 ...
 $ ALTITUDE: int  43 341 4152 74 6752 52 25 95 6360 74 ...
 $ LATITUDE: num  40.8 40.2 33.8 39.4 39.3 37.8 38.5 37.4 36.6 36.7 ...
 $ DISTANCE: int  1 97 70 1 150 5 80 28 145 12 ...
 $ SHADOW  : int  1 2 2 1 1 1 2 2 1 2 ...
```

Notice that the variable `SHADOW` is recorded as a numeric value. This is not an accurate depiction of the data set.

Data transformation: `factor`

To ensure analysis can be done properly, we need to convert the values in `SHADOW` into categorical values in the data set.

```
rain_df$SHADOW_f <- factor(rain_df$SHADOW,
                           levels=c("1", "2"),
                           labels=c("Westward", "Leeward"))
```

Here we are making references to the numerical values 1 and 2, by setting them as “Westward” and “Leeward”, respectively.

```r
str(rain_df)
```

```
'data.frame':   30 obs. of  7 variables:
 $ STATION : chr  "Eureka                   " "RedBluff                  " "Thermal                   " "For
 $ PRECIP  : num  39.6 23.3 18.2 37.5 49.3 ...
 $ ALTITUDE: int  43 341 4152 74 6752 52 25 95 6360 74 ...
 $ LATITUDE: num  40.8 40.2 33.8 39.4 39.3 37.8 38.5 37.4 36.6 36.7 ...
 $ DISTANCE: int  1 97 70 1 150 5 80 28 145 12 ...
 $ SHADOW  : int  1 2 2 1 1 1 2 2 1 2 ...
 $ SHADOW_f: Factor w/ 2 levels "Westward","Leeward": 1 2 2 1 1 1 2 2 1 2 ...
```

8. Descriptive statistics

Descriptive statistics are numerical and graphical summaries used to illustrate and describe a data set. We will go over some common ones:

- Mean
- Median
- Variance and standard deviation
- Minimum and maximum
- Quantile

- Mean or average of a sequence of numbers can be obtained using the function `mean()`.

```
mean(rain_df$PRECIP)
```

```
[1] 19.80733
```

- Median of a sequence of numbers can be obtained using the function `median()`.

```
median(rain_df$PRECIP)
```

```
[1] 15.345
```

- The variance of a sequence of numbers can be obtained using the function `var()`.

```
var(rain_df$PRECIP)
```

```
[1] 276.2639
```

- The standard deviation can be obtained using the function `sd()` or by taking the positive square root of the variance.

```
sd(rain_df$PRECIP)
```

```
[1] 16.62119
```

```
sqrt(var(rain_df$PRECIP))
```

```
[1] 16.62119
```

- The minimum and maximum of a set of numbers can be obtained through `min()` and `max()`.

```
min(rain_df$PRECIP)
```

```
[1] 1.66
```

```
max(rain_df$PRECIP)
```

```
[1] 74.87
```

9. Data visualization using the base package

The numerical summaries discussed so far can be visualized with some simple graphs:

- Histograms
- Boxplots
- Barplots
- Scatterplots

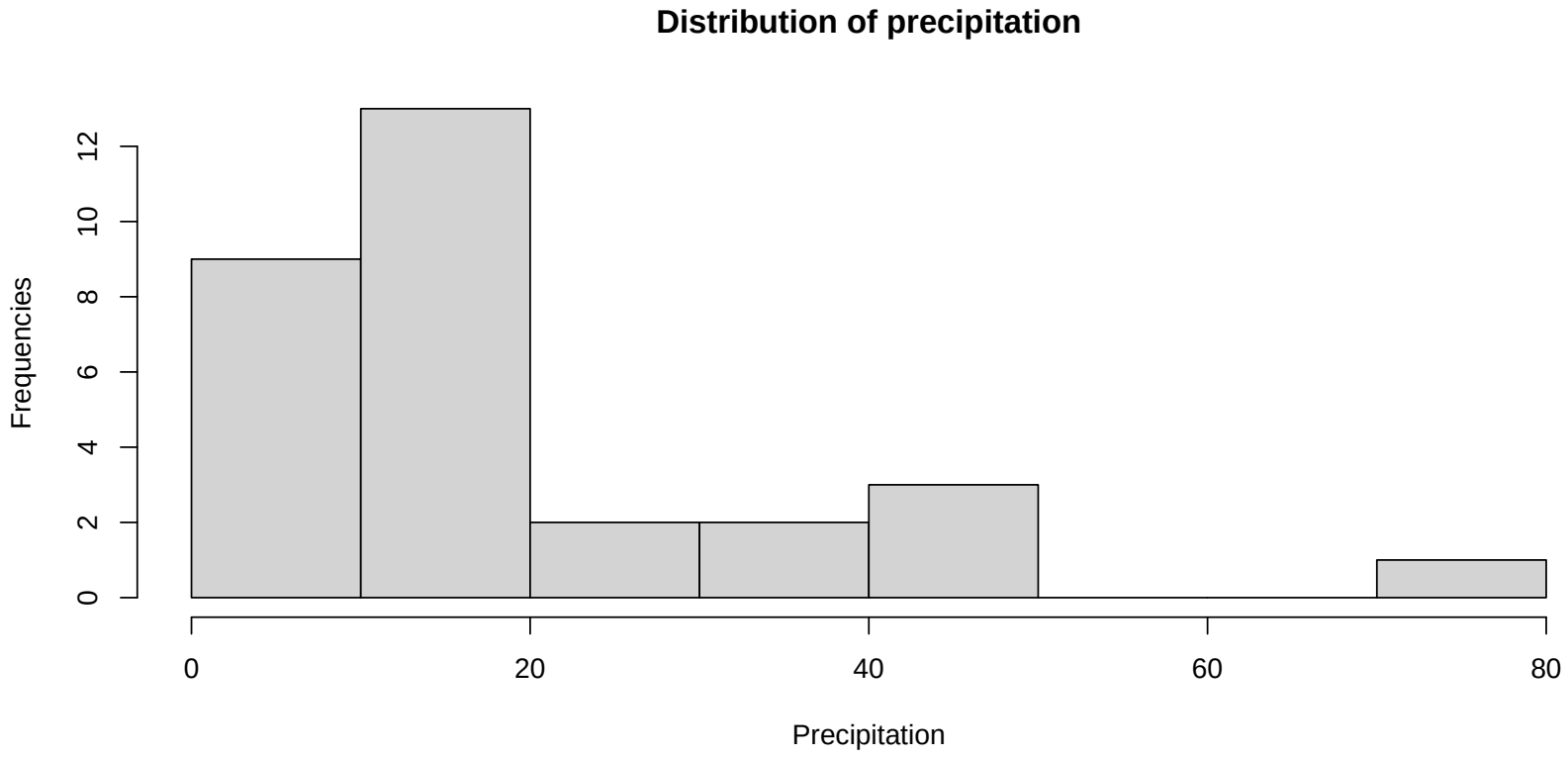
9.1 Histogram

Histograms are commonly used to visualize the distribution of continuous variables. When looking at a histogram, pay attention to its:

- Shape: symmetric vs asymmetric
- Center
- Spread

Histograms

```
hist(rain_df$PRECIP,main="Distribution of precipitation",
     xlab="Precipitation", ylab= "Frequencies")
```



Notice that there is no space in between the bars/bins.

Practice 9.1

When presented with a histogram, the shape of distribution is dependent on the width of the bins or the number of the bins. Wider bins will result in lesser number of bins, whereas narrower bins will have more bins.

We can change the number of bins in the histogram by specifying the number through `breaks`.

```
hist(rain_df$PRECIP,main="Distribution of precipitation",
     breaks = 5, # Specify the number of bins
     xlab="Precipitation", ylab= "Inches")
```

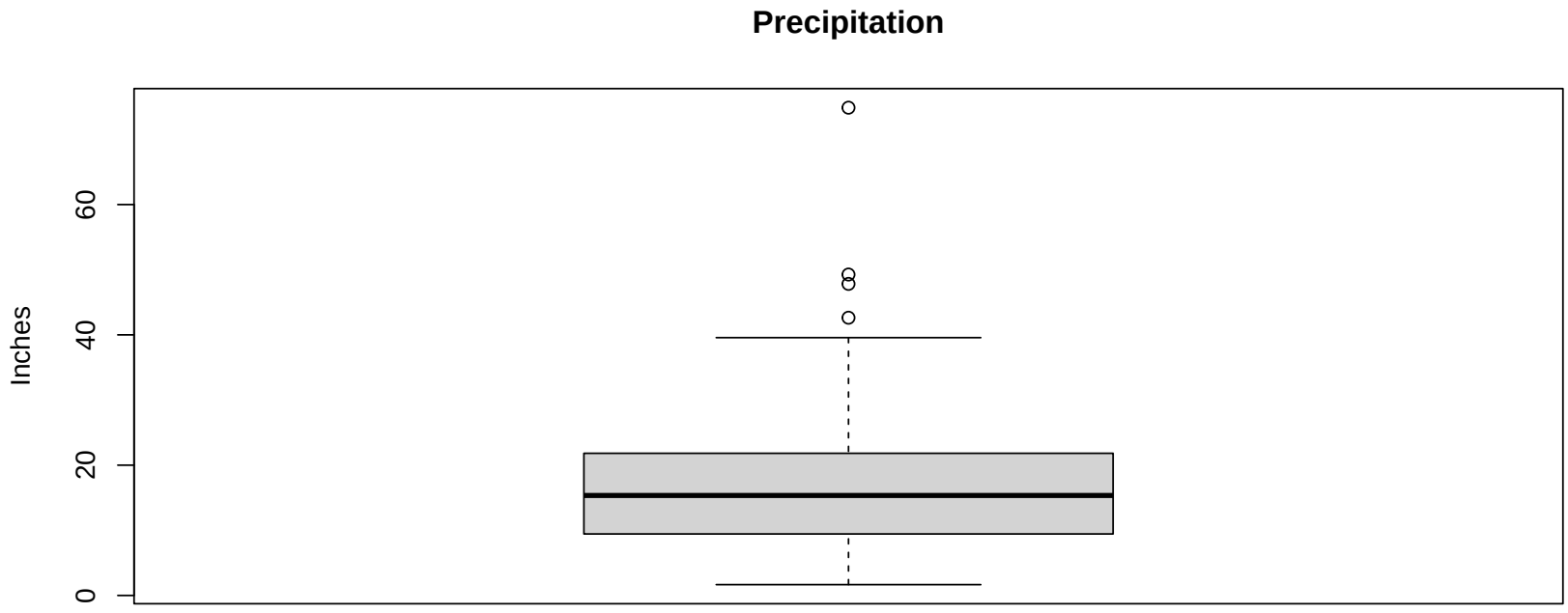
Try a few values (e.g. 5, 10, 20, 50) to see how the effect of the width of the bins and the shape of the distribution.

9.2 Boxplots

- Minimum
- First quartile, Q_1
- Second quartile, Median
- Third quartile, Q_3
- Maximum

Potential outliers are shown as dots outside the boxplots.

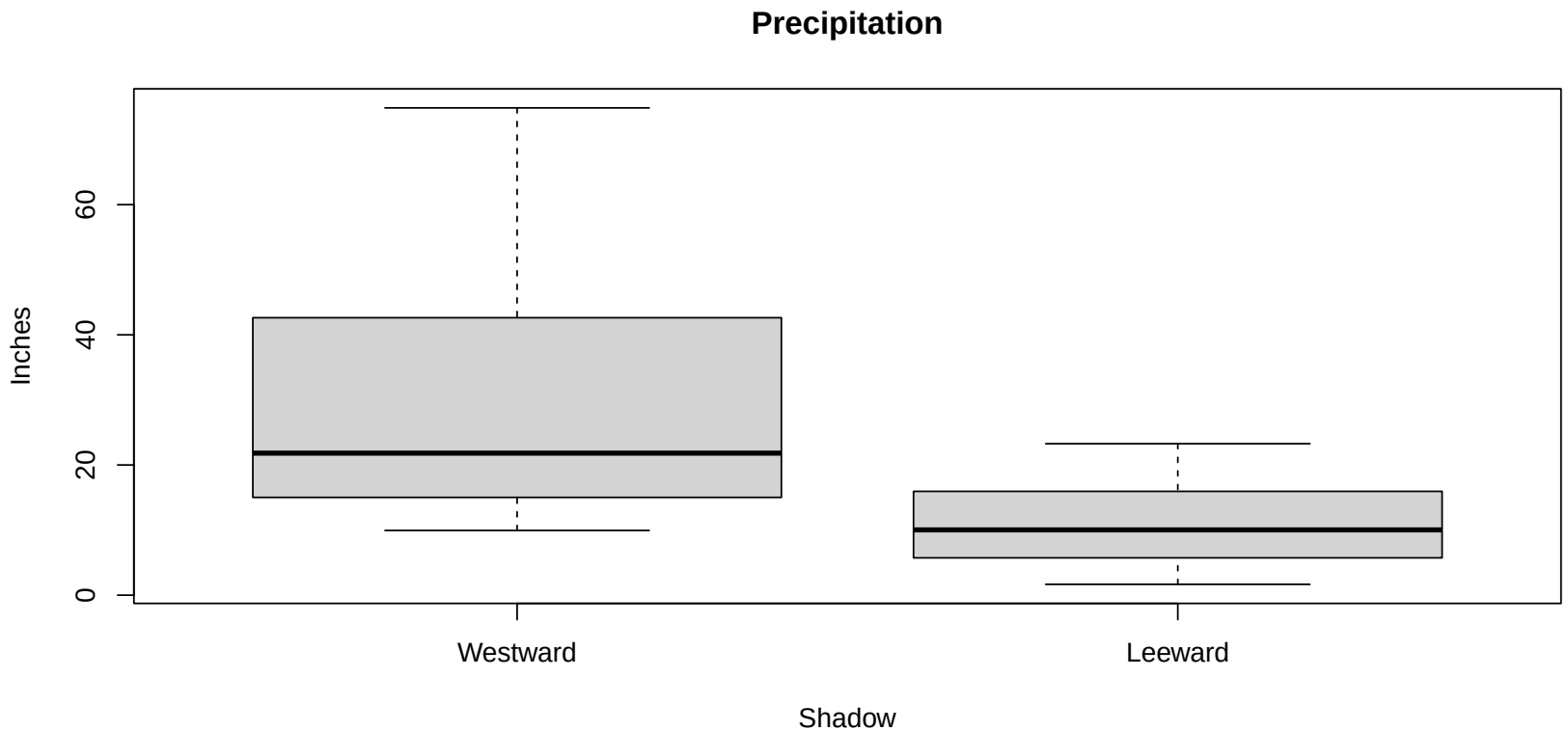
```r
boxplot(rain_df$PRECIP, main="Precipitation",
        ylab= "Inches")
```



Do you see potential outlier?

Side-by-side boxplots

```
boxplot(rain_df$PRECIP~rain_df$SHADOW_f,
        main="Precipitation",
        xlab = "Shadow", ylab= "Inches")
```



Side-by-side boxplots are commonly used to visualize relationship between a continuous variable and a categorical variable.

Practice 9.2

The variable `ALTITUDE` is a continuous variable.

1. Visualize this variable using the histogram.
2. Visualize this variable using the boxplot that lies horizontally.
3. Comment on the similarities between the two graphs.

9.3 Bar graphs

- Bar graphs are commonly used to visualize categorical variables.
- In order to make a bar in R, we begin by preparing a table.

Suppose we are interested to know the proportion of station in `rain_df‘` that are located in different locations.

```
shadow_table<- table(rain_df$SHADOW_f)
shadow_table
```

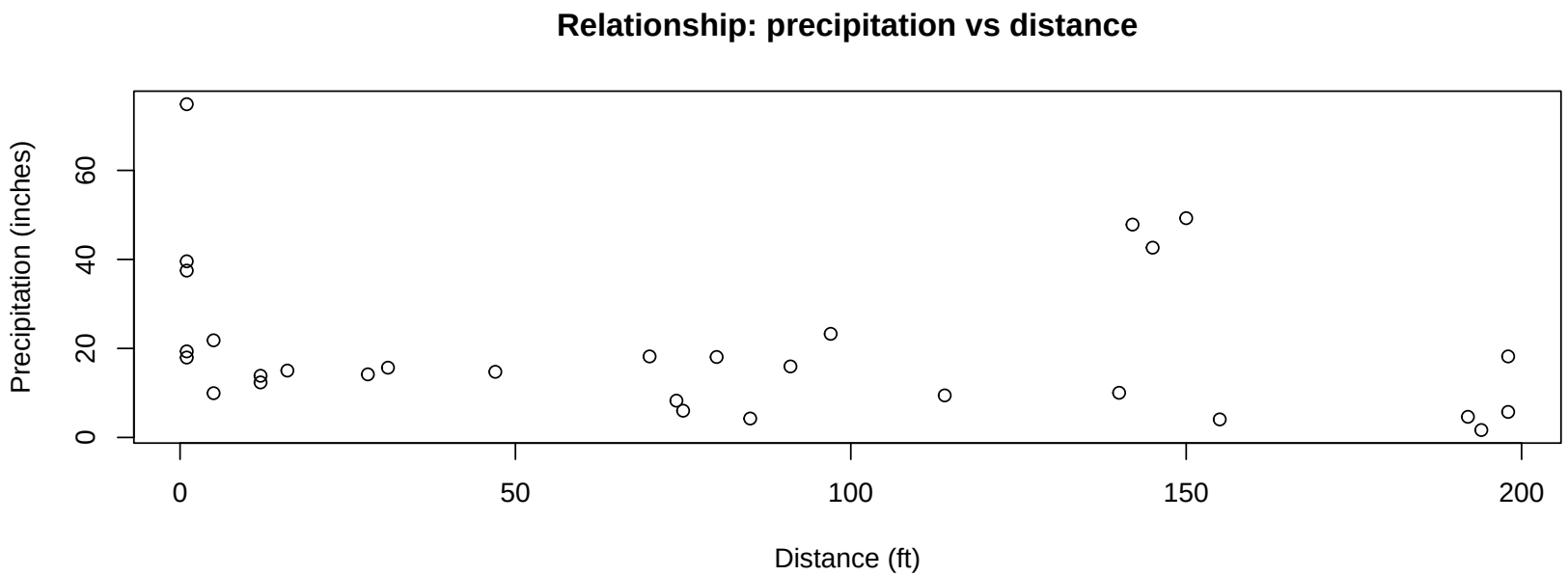
```
Westward  Leeward
      13        17
```

```
barplot(shadow_table, main="Distribution of Drinks",
        ylab="Frequency")
```

9.4 Scatterplots

Scatterplots are used to visualize relationship between two continuous variables.

```
plot(rain_df$DISTANCE, rain_df$PRECIP,
     main="Relationship: precipitation vs distance",
     xlab="Distance (ft)", ylab="Precipitation (inches)")
```



10. Library installation

- The R user community creates functions and data sets to share. We call them packages or libraries.
- All packages are free and can be installed as long as there is access to the internet.
- Some commonly used packages are: `ggplot2` for data visualization, `dplyr` for data management, `MASS`, `car`, etc.
- To install a library, say `ggplot2`, you can either use the RStudio interface, or you can do it from the command line as follows:

```
install.packages("ggplot2", dependencies=TRUE)
```

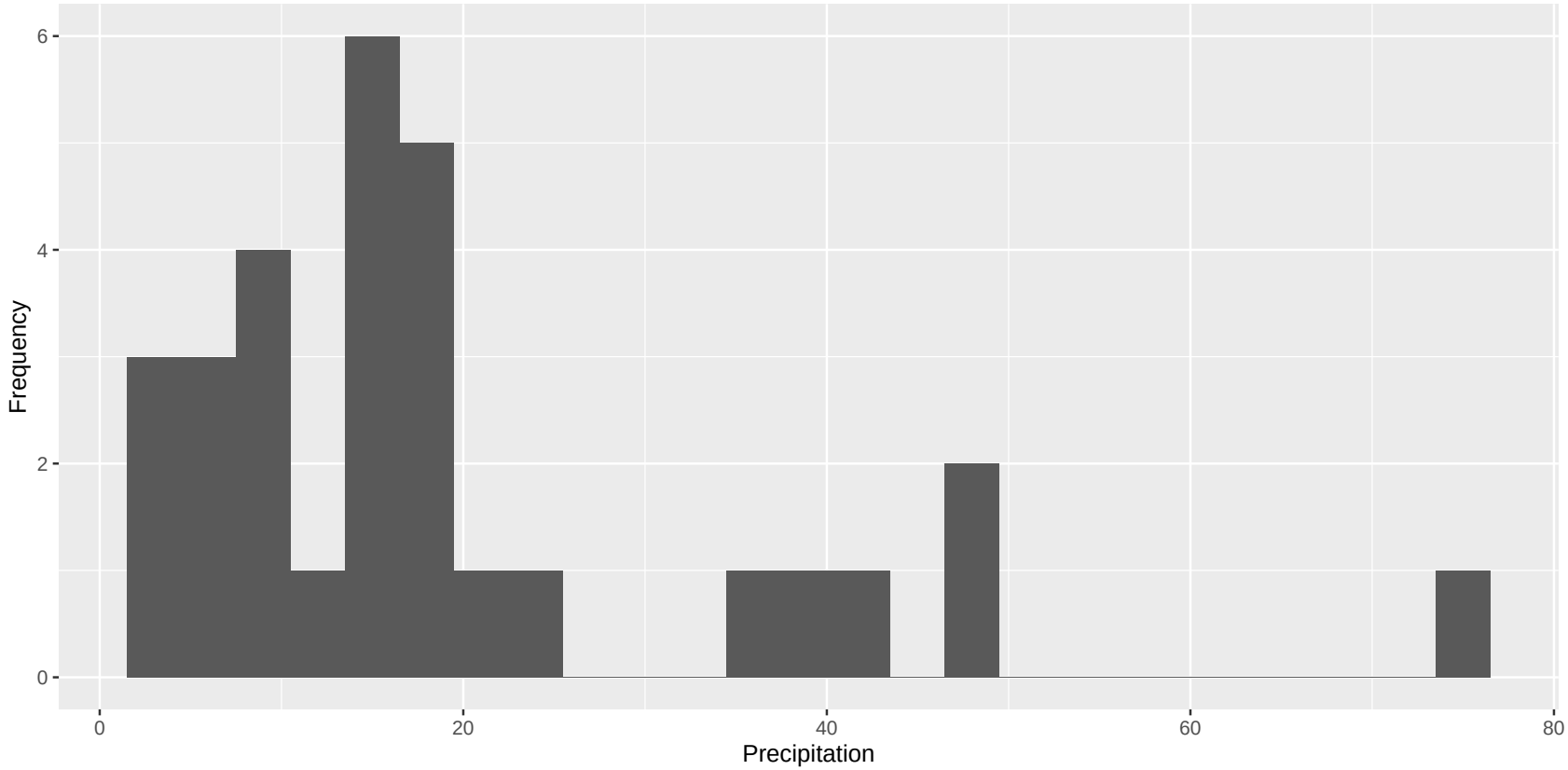
- Every time you want to use the package in a new R session, the following code needs to be executed:

```
library(ggplot2)
```

- We also recommend updating the installed libraries regularly. To do so, go to `Tools` > `Check for Package Updates` and follow the instructions to complete the updates.

Demonstration of ggplot2

```
library(ggplot2)
ggplot(rain_df, aes(x=PRECIP)) +
  geom_histogram(binwidth=3) +
  xlab("Precipitation") + ylab("Frequency")
```



Demonstration of dplyr

```
library(dplyr)
rain_df %>%
  group_by(SHADOW) %>%
  summarize(avg_precip = mean(PRECIP),
            sd_PRECIP=sd(PRECIP))
```

```
## # A tibble: 2 x 3
##   SHADOW avg_precip sd_PRECIP
##    <int>      <dbl>     <dbl>
## 1      1       31.0     19.4
## 2      2       11.3      6.39
```

11. Next steps

Now that you have learnt the fundamentals of R, you can explore more about R such as

- data visualization with `ggplot2`,
- data management and manipulation with `dplyr`,
- exploratory data analysis,
- linear regression with R, etc.

The SCSRU organizes similar workshops to this on a regular basis to improve quality of research and data literacy among the UWaterloo community. We also provide 1-1 free consultation to all researchers on campus. More information are available on our website.

Thank you!