MATH9102 Fundamentals of Data Analysis

FUNDAMENTALS –GETTING STARTED WITH R

DR. DEIRDRE LAWLESS

Time to get practicing …

INSTALL R AND YOUR IDE

Choose an IDE

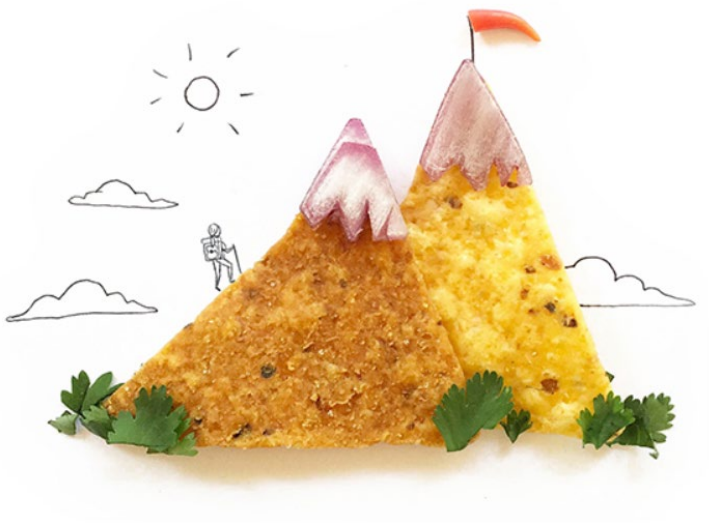
R Studio

DataSpell

VSCode

Getting Started with R

nners



rting point will serve all beginners, but here are 6 ways to begin learning R.

l **R, RStudio, and R packages like the tidyverse.** These three installation steps are often

I will be using R and Rstudio during classes.

Getting started with R and RStudio: https://education.rstudio.com/learn/beginner/

R

R is a programming language

- "base" R
  - Comes with R out of the box, no extra packages.
  - Philosophy
    - More "procedural" and function-based.
    - Less opinionated than other languages: you can do the same task in many different ways.
  - Functions like mean(), sum(), lm(), plot().
  - You manipulate data with vectors, matrices, lists, and data.frames.
  - You need to fully read and understand all the parameters and their values to understand what the code is doing

R

There are lots of different aspects of behaviour you will need to use.

These are available as *packages* you can install as needed.

Packages are collections of functions and datasets that add new features to R.

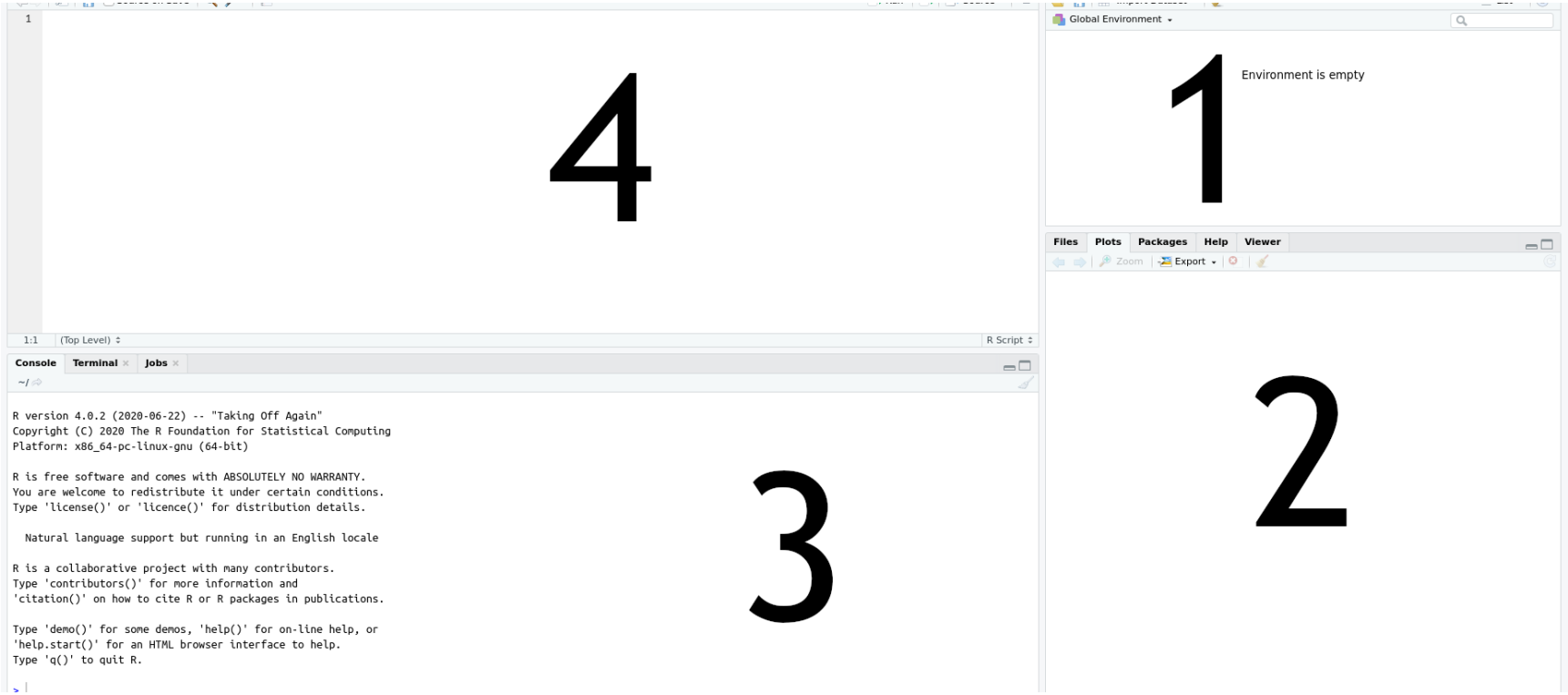
Most R packages are available from CRAN, the official R repository – a network of servers (so-called mirrors) around the world.

We will talk about how to install these packages in a later slide.

R

R is a programming language

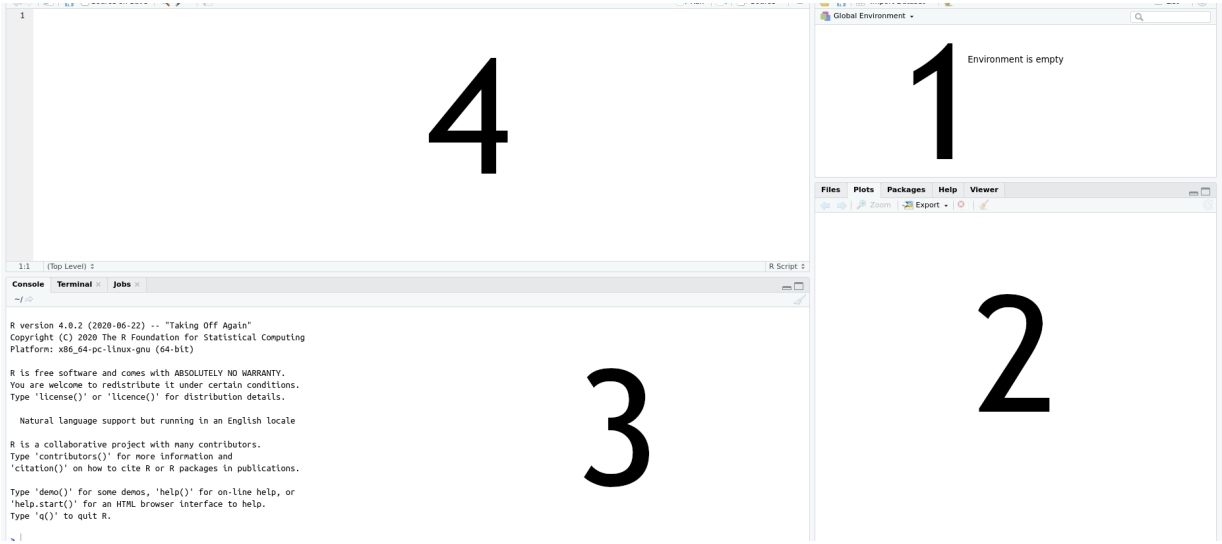
- “tidyverse”
  - A collection of add-on packages for working with different types of data.
  - There are separate functions for everything, which is perfect for code that relies on pipes
    - **Pipes (|)** are operators that let you improve your code’s readability and restructure your code so that it is read from left to right instead of from the inside out
  - Both **ggplot2** and **dplyr** are part of the tidyverse, which are key data visualization and manipulation packages.



R Studio Interface

R Studio Interface

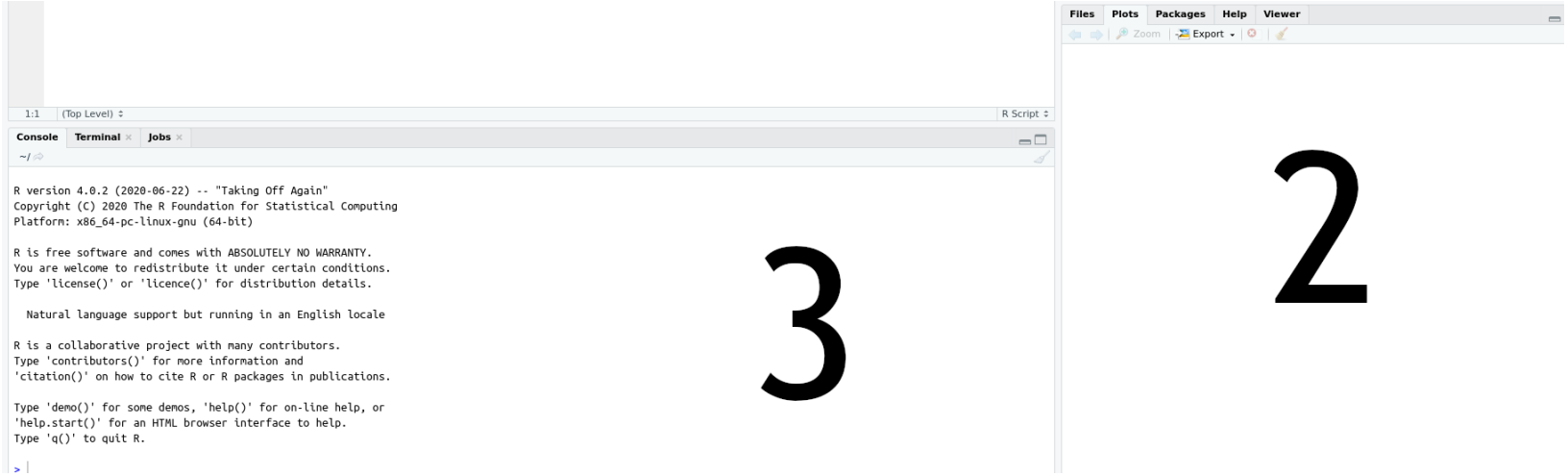
1. The Environment pane
   - Where a list of the data you have imported and created can be found.
2. The Files, Plots and Help pane
   - Where you can see a list of available files, will be able to view graphs that you produce, and can find help documents for different parts of R.
3. The Console pane
   - Used for running code. This is where we'll start with the first few examples.
4. The Script pane
   - Used for writing code. This is where you'll spend most of your time working



R Studio Interface

The Console pane (3)

- Used for running code. This is where we'll start with the first few examples
- On startup this will show the R startup message and give you a prompt >
- You can type commands (code) directly into the Console and the results will be displayed below it or in the Viewer in the Files, Plots and Help pane (2)



R Studio Interface

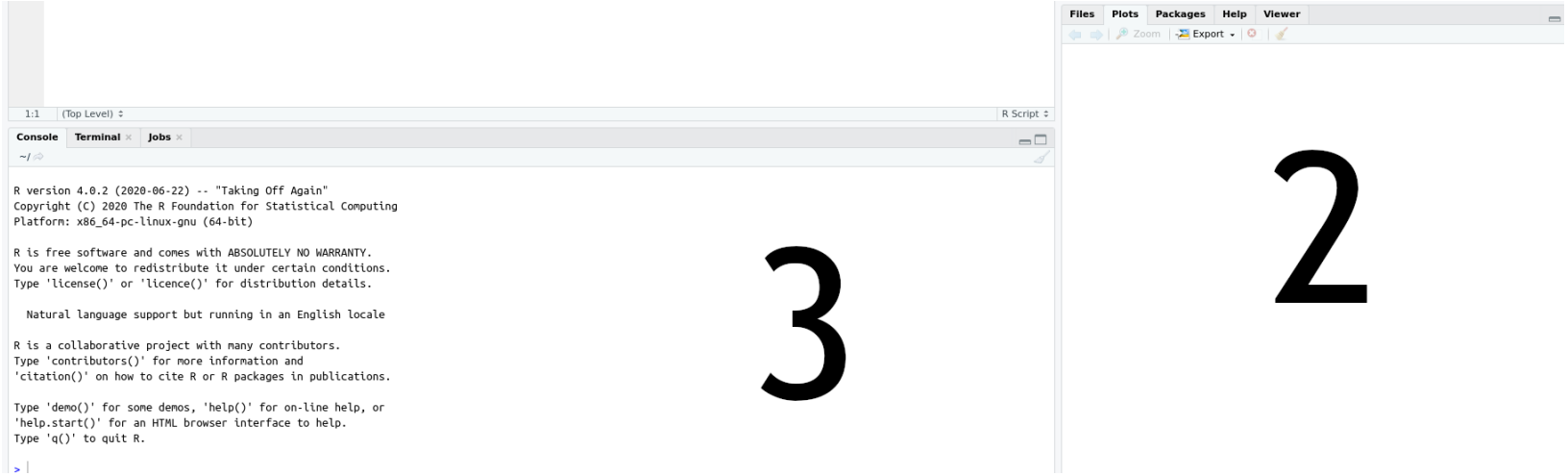
Or you can create scripts and execute them

You will see the output in the Console and viewer

- Or you can redirect your output to a file

You can run a script line by line (hit CTRL + Enter or position the cursor on the line and hit run or) OR

Run the entire script (hit CTRL + Shift + Enter or select all the code and hit Run)



R

You can **install** packages from the Console or within a script:

- install.packages("tidyverse")
- OR in your IDE
  - R Studio Tools -> Install Packages

In order to use the functions contained in the packages you need to **load** it

- library("tidyverse")

R variables

You can store data into variables using the <- operator

E.g.

```
x <- 4
```

You can see what is stored in the variable by inputting the variable name at the console

Example:

```
income <- 100
taxes <- 20
net_income <- income - taxes


net_income
```

R variables

You can use the variable/variables in functions e.g.

```
abs(income + taxes )
```

Task 1

Install the **tidyverse** package into your IDE (from the Console)

Load the package (this will verify that it has successfully installed)

Task 2

Try entering some commands at the console e.g.

```
1+1
2*7
sqrt(4)
```

Working with data

We will usually want to work with sets of data

We can create a single variable to handle a list of values of set of variables with values

Vector

Each item is an element. E.g.

```
age <- c(28, 48, 47, 71, 22, 80, 48, 30, 31)
```

We can apply calculations to it

```
age * 12
```

```
You can use it in functions e.g.:
```

```
min(age)
```

```
max(age)
```

```
sum(age)
```

Working with data

Usually we want to work with more than one variable

In this case we use a **dataframe**

We can use a second vector:

```
purchase <- c(20, 59, 2, 12, 22, 160, 34, 34, 29)
```

And combine it with age into a dataframe:

```
techsales<- data.frame(age, purchase)
```

If you enter techsales into the Console you should see each element in age displayed beside the equivalent element in purchase

Task 3

Try out some of the tidyverse functions

```
# Example: Filtering rows in a data fame
filtered_data <- filter(techsales, age > 40)
filtered_data


#View all the data
view(techsales)
```

Task 3

```
# Create a categorical description using mutate

#%>% operator, known as the pipe operator, is used in the tidyverse


df_categorized <- techsales %>%
  mutate(
    purchase_category = case_when(
      purchase > 100 ~ "Expensive",
      purchase <= 100 & purchase >= 20 ~ "Reasonable",
      TRUE ~ "Cheap"
    )
  )
df_categorized
```

Getting Started

Download the script `MATH9102W1.R`

Put this somewhere you can find it and open it in your chosen editor

We will review it section by section

Getting Started

SECTION ONE: INSTALL AND LOAD THE PACKAGES

Getting Started Interrogating the Dataset

SECTION TWO : SETUP THE DATAFRAME

Getting Started Interrogating the Dataset

SECTION THREE : LOOK AT THE DATA