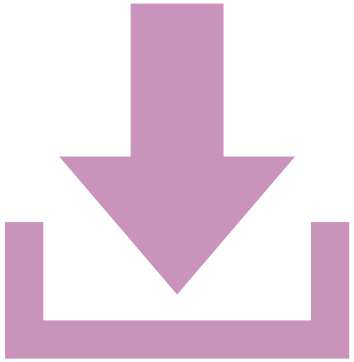
ECON6027 1b

INTRODUCTION TO R PROGRAMMING

Download R and RStudio



- Download R: https://cran.r-project.org
- Download RStudio Desktop (Integrated Development Environment): https://www.rstudio.com/products/rstudio/download/
- If you already have R and RStudio, please update.

Packages

Packages are the fundamental units of reproducible R code that include reusable functions, the documentation that describes them and sample data.

Some main packages we will use in this course are:

sf	sp	spData	tmap	gstat	spatstat	spdep	spatialreg

To download a package use command:

```
install.packages("sf", dependencies = T)
```

We will also use some other packages which can be downloaded as and when needed.

Use and update packages

- check installed packages:

  installed.packages()

- load the package

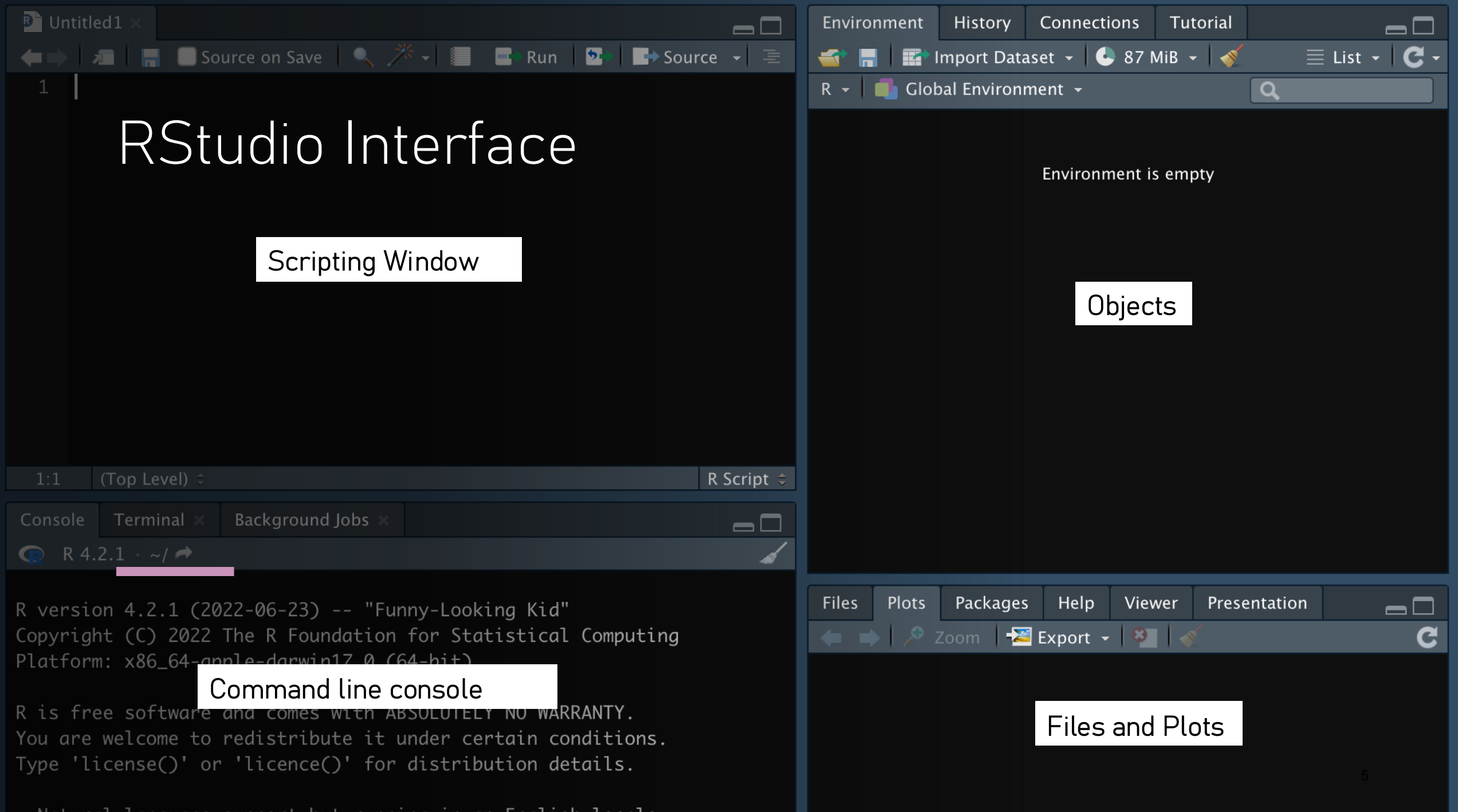
  library(sf)

- list all packages where an update is available

  old.packages()

- update all available packages without prompts for permission/clarification

  update.packages(ask = FALSE)


Untitled1
Source on Save
Run
Source
1
RStudio Interface
Scripting Window
1:1
(Top Level)
R Script
Environment
History
Connections
Tutorial
Import Dataset
87 MiB
List
R
Global Environment
Environment is empty
Objects
Console
Terminal
Background Jobs
R 4.2.1 · ~/
R version 4.2.1 (2022-06-23) -- "Funny-Looking Kid"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)
Command line console
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Files and Plots

Quick start

"**>**" in the command line console is called the "prompt"

To execute a command, type the syntax in the console and hit enter.

It is recommended that you prepare your code in a "script file" first and then run the commands.

- To run a command, move the cursor to the line you want to run and press "run" (keyboard shortcuts are available.)
- The script file associated with this deck is **"R intro.R"**

If you want to add explanatory notes to your script that you want R to ignore, use the comment key "#".

Set working directory

To let R know where to open and save files to.

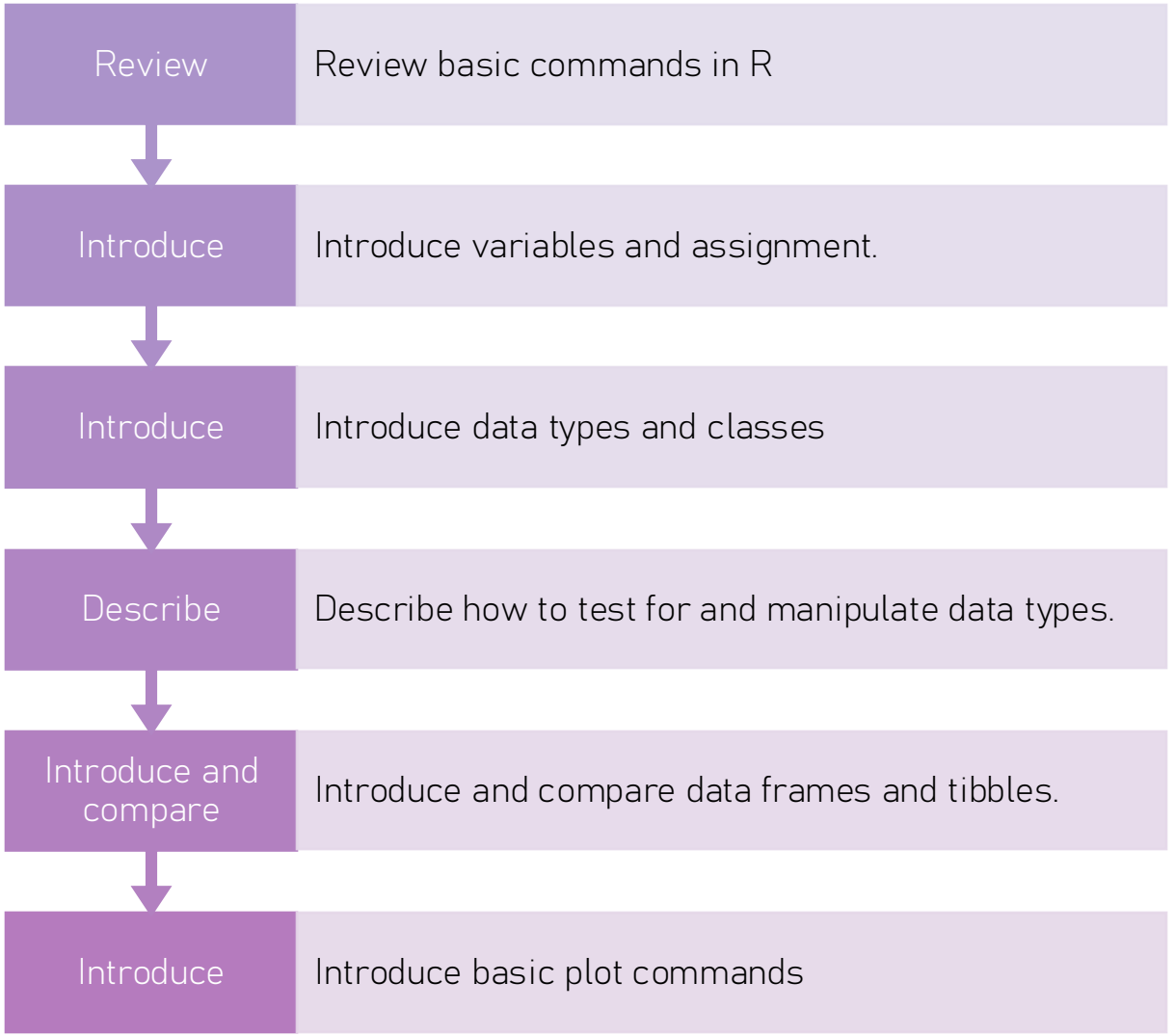
To set working directory:

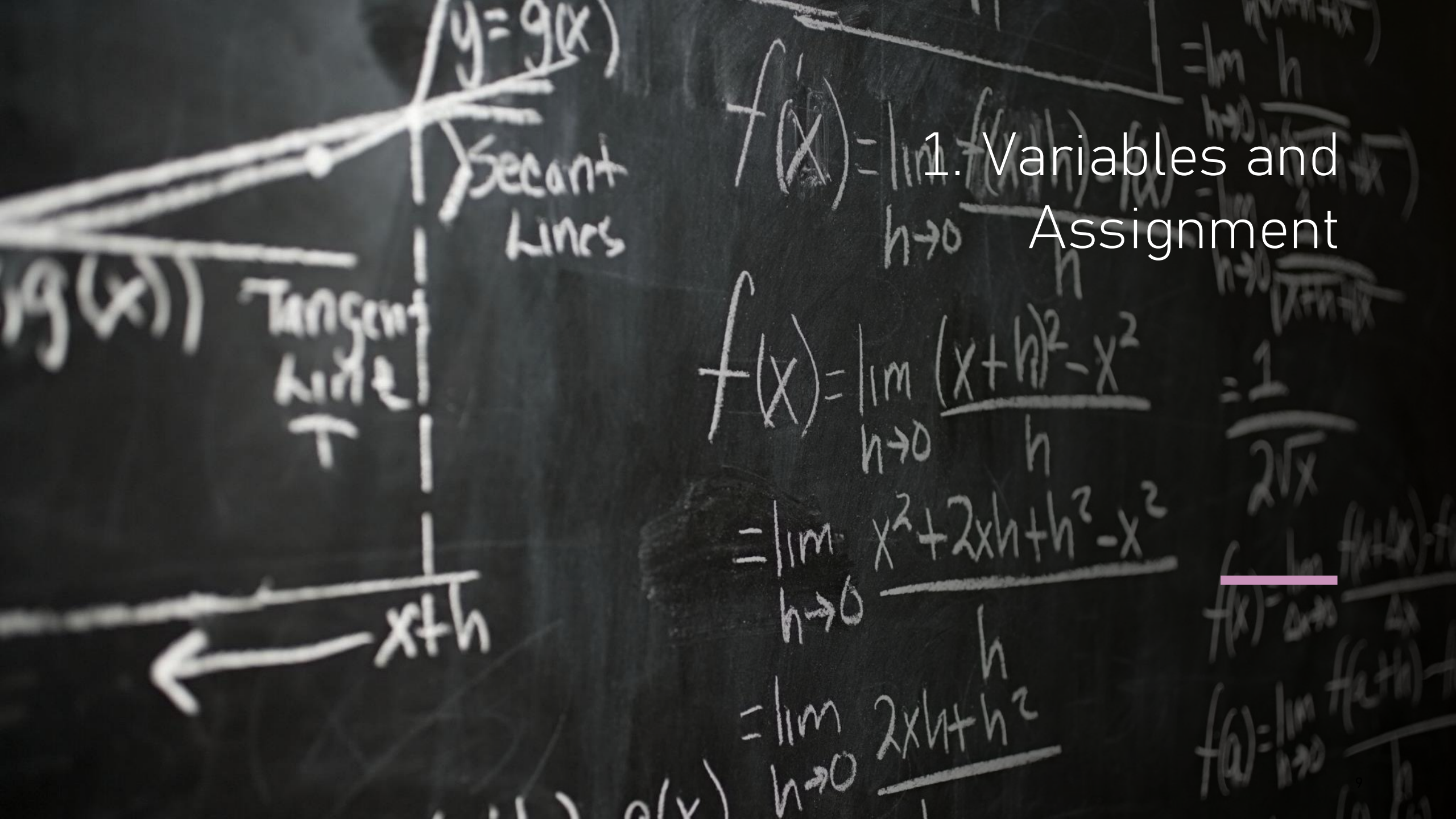
- Go to “Files” tab in Files and Plots window in RStudio.
- Navigate to the folder you wish to choose.
- Click on the more button.
- **“Set as Working Directory”**

You should see something like below in the command line.

- > setwd("~/Dropbox/MSc/Lessons/1 Introduction")

Objectives of this lesson





1. Variables and Assignment

Simple Numerical Variable Assignment

- R can be used as a calculator.
- It is convenient to "assign" values to an "R object"
- The objects created can be manipulated or subject to further operations.
- *Note: I have provided the script for the code chunks in this deck. However, I highly recommend you create your own.*

```
# examples of simple assignment
x <- 5
y <- 4
# the variables can be used in other operations
x+y
```

```
[1] 9
```

```
# including defining new variables
z <- x + y
z
```

```
[1] 9
```

```
# which can then be passed to other functions
sqrt(z)
```

```
[1] 3
```

```r
# example of Vector assignment
tree.heights <- c(4.3,7.1,6.3,5.2,3.2,2.1)
tree.heights
```

```
[1] 4.3 7.1 6.3 5.2 3.2 2.1
```

```r
tree.heights**2
```

```
[1] 18.49 50.41 39.69 27.04 10.24  4.41
```

```r
sum(tree.heights)
```

```
[1] 28.2
```

```r
mean(tree.heights)
```

```
[1] 4.7
```

```r
max.height <- max(tree.heights)
max.height
```

```
[1] 7.1
```

```r
tree.heights [1]    # first element
```

```
[1] 4.3
```

```r
tree.heights[1:3]   # a subset of elements 1 to 3
```

```
[1] 4.3 7.1 6.3
```

```r
sqrt(tree.heights[1:3]) #square roots of the subset
```

```
[1] 2.073644 2.664583 2.509980
```

```r
tree.heights[c(5,3,2)]  # a subset of elements 5,3,2: note the ordering
```

```
[1] 3.2 6.3 7.1
```

Vector Assignment

- “c” instructs R to combine or concatenate multiple values to form vectors: eg, tree.heights
- You can perform operations on all the values in tree.heights such as illustrated here.

Character or Logical Variable Assignment

- Example of character variable assignment.

```
# Character Variable assignment
name <- "Your Name"
name
```

```
## [1] "Your Name"
```

```
cities <- c("Singapore","Tokyo","Beijing","Mumbai","Colombo")
cities
```

```
## [1] "Singapore" "Tokyo"     "Beijing"   "Mumbai"    "Colombo"
```

```
length(cities)
```

```
## [1] 5
```

- Example of logical variable assignment.

```
country <- c(TRUE, FALSE, FALSE, FALSE, FALSE)
country
```

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

```
cities[country]
```

```
## [1] "Singapore"
```

2. Data Types and Data Classes

Data Types

- Data in R can be considered as being organised into a hierarchy of data types which can then be used to hold data values in different structures.
- The basic data types and associated tests and conversions are shown in the table.

Type	Test	Conversion
character	is.character	as.character
complex	is.complex	as.complex
double	is.double	as.double
expression	is.expression	as.expression
integer	is.integer	as.integer
list	is.list	as.list
logical	is.logical	as.logical
numeric	is.numeric	as.numeric
single	is.single	as.single
raw	is.raw	as.raw

Example: numeric

Create a numeric string

```
> numeric(5)
[1] 0 0 0 0 0
```

Check whether an object is numeric

```
> is.numeric(tree.heights)
[1] TRUE
```

Convert a variable to numeric (notice the error message since August is not numeric!)

```
> as.numeric(c("9", "August", "2022"))
[1]    9   NA 2022
Warning message:
NAs introduced by coercion
```

Example: Logical

Different ways to create a logical string:

```
> logical(5)
[1] FALSE FALSE FALSE FALSE FALSE
> as.logical(c("TRUE", "T", "True", "true"))
[1] TRUE TRUE TRUE TRUE
> as.logical(c("FALSE", "F", "False", "false"))
[1] FALSE FALSE FALSE FALSE
> as.logical(c(7, 5, 0, -4,5)) # 0 is false, any other
  value is true
[1]  TRUE  TRUE FALSE  TRUE  TRUE
```

Check whether a vector is logical

```
> is.numeric(country)
[1] FALSE
> is.logical(country)
[1] TRUE
```

Logical test

```
> (large <- (tree.heights>5))
  [1] FALSE  TRUE  TRUE  TRUE FALSE FALSE
> tree.heights[large]
  [1] 7.1 6.3 5.2
```

Example: Characters

Create a character string

```
> character(5)
[1] "" "" "" "" ""
```

Tests

```
> is.character(cities)
[1] TRUE
> is.character(country)
[1] FALSE
```

Data Classes

- The different data types can be used to populate different data structures known as “classes”.

```
class(tree.heights)
```

```
## [1] "numeric"
```

```
class(cities)
```

```
## [1] "character"
```

```
class(large)
```

```
## [1] "logical"
```

Popular Classes (Try these!)

Class: data.frame

```
> my.data =
  data.frame(a=10:15,
  b=15:20)
> my.data
> class(my.data)
```

Class: vector

```
> is.vector(my.data)
> as.vector(my.data)
```

Class: matrix

```
my.matrix <-
  matrix(1:6, ncol =
  2)
class(my.matrix)
colnames(my.matrix)
  <- c("c1", "c2")
rownames(my.matrix)
  <- c("r1",
  "r2","r3")
my.matrix
> rowSums(my.matrix)
```

Class: factor (creates specific categories)

See the script file for factor and ordered factor

For more matrix algebra functions: https://www.statmethods.net/advstats/matrix.html

List (combine many different classes togehter)

```
## $country
## [1] "Singapore"
##
## $year
## [1] 2019 2020 2021
##
## $indicator
## [1] "inflation.pct"
##
## $value
##      [,1]  [,2] [,3]
## [1,] 0.57 -0.18  2.3
```

Create

Create list

```
> my.list <- list("Singapore", c(2019,2020,2021),
 "inflation.pct", matrix(c(0.57,-0.18, 2.3),c(1,3)))
> my.list
```

Select

Select list item

```
> my.list[[3]]
```

Assign

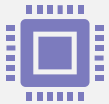
Assign names to the list items

```
> my.list <- list(country="Singapore",
 year=c(2019,2020,2021), indicator="inflation.pct",
 value=matrix(c(0.57,-0.18, 2.3),c(1,3)))
> my.list
```

Things to note

In R you can create your own classes and associate it with specific behaviours such as its own way of printing, drawing, etc.

This is how contributors keep adding new and fun ways of analysing and visualising data in R.



Spatial data for example has its own class(es) in order to manipulate the locational awareness (coordinates).

We will encounter many more interesting classes as we proceed.

“data.frames” and “tibbles”

Datasets in R

- In a general dataset

  Columns: attribute

  Rows: feature (observation)

- In R, data can be in a matrix, but a matrix can only hold one "type" of data. However, datasets can contain different types of data (numeric, character, etc)!

  solution: data.frames and tibbles

data.frame

- The most used method of storing data in R.
- A data frame can be created using the data.frame() function.
- Type ?data.frame() to learn more about the function.
- Type data() to get a list of all the datasets loaded to the current session.

```
# create data.frame
L3 <- LETTERS[1:3] # create an object consis
ting of A, B & C
fac <- sample(L3, 10, replace = TRUE) # gene
rate a sample of size 10 from L3 with replac
ement
df <- data.frame(x = 1, y = 1:10, fac = fac)
 # create a dataframe with three attributes:
 x, y, & fac
df
```

```
##    x  y fac
## 1  1  1   A
## 2  1  2   B
## 3  1  3   C
## 4  1  4   B
## 5  1  5   A
## 6  1  6   A
## 7  1  7   C
## 8  1  8   C
## 9  1  9   B
## 10 1 10   B
```

```
df$y # extract column y
```

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

“trees” data frame

```
# load "trees" dataset
data("trees")
head(trees) # view the top 6 entries
```

```
##   Girth Height Volume
## 1   8.3     70   10.3
## 2   8.6     65   10.3
## 3   8.8     63   10.2
## 4  10.5     72   16.4
## 5  10.7     81   18.8
## 6  10.8     83   19.7
```

- Load the “trees” dataset from the library of inbuilt datasets in R (the full list and explanatory notes can be found here).

  > data(trees)

tibble (from the package "tibble")

```
# create tibble
library(tibble)
tb = tibble(x = 1, y = 1:10, fac = fac)

# 1. versatile
class(tb)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

- "tibble" class is a reworking of of the data.frame class.
- tibbles retain the operational advantages of data frames but eliminate aspects that are less effective.
- tibbles have many advantages over data frames.
  a. Versatile: tibbles are also data frames
  b. No partial matching
  c. A subset of a tibble is also a tibble (whereas a subset of a data frame will lose its original properties)

```
# 2. partial matching
df$f
```

```
##  [1] "A" "C" "A" "B" "A" "B" "A" "B" "B" "C"
```

```
tb$f
```

```
## Warning: Unknown or uninitialised column: `f`
```

```
## NULL
```

b. Partial matching

- $ operator can be used to extract a variable.
- Notice df$f extracts the column “fac” by partial matching although there is no variable called “f”. This is not the case when you run tb$f

c. Sub-setting

- df[,2] extracts the second column of df.
- Notice the class of df[,2] is integer whereas the class of tb[,2] is a tibble (has not lost its original properties!)

```
# subsetting
class(df[,2]) # class of column 2 in df
```

```
## [1] "integer"
```

```
class(tb[,2])
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Points to note

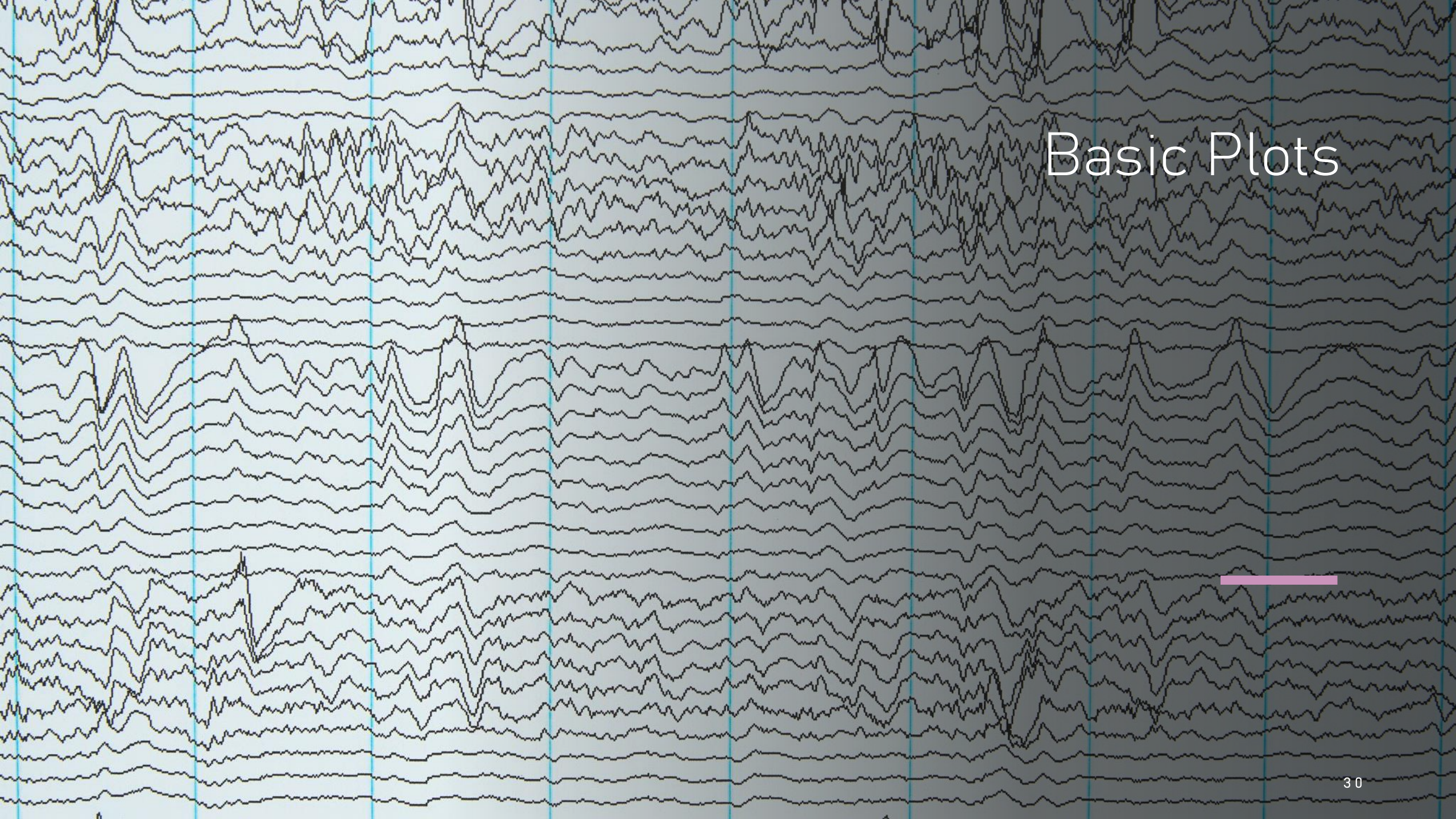
- It is possible to convert a data.frame to a tibble using as_tibble() function.
- The print method for tibble is to report the top 10 entries by default, whereas the data fame will print all the entries. The head() function is used to print the top 6 entries of a data frame.

```
# convert data.frame to tibble
trees.tb <- as_tibble(trees)
class(trees.tb)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

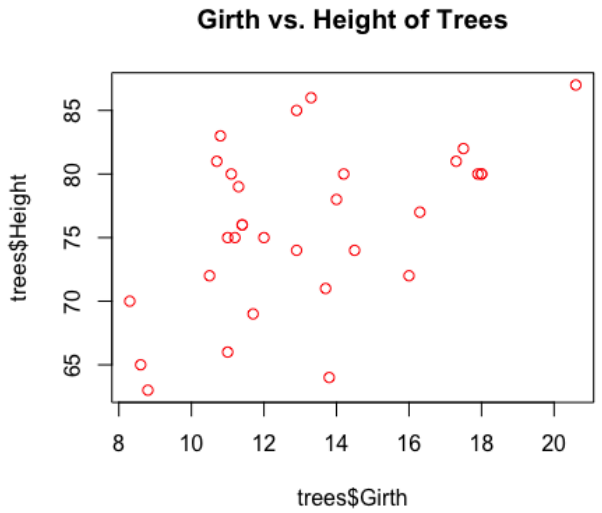
```
trees.tb # returns only the top 10 entries
```

```
## # A tibble: 31 × 3
##    Girth Height Volume
##    <dbl>  <dbl>  <dbl>
##  1   8.3     70   10.3
##  2   8.6     65   10.3
##  3   8.8     63   10.2
##  4  10.5     72   16.4
##  5  10.7     81   18.8
##  6  10.8     83   19.7
##  7  11       66   15.6
##  8  11       75   18.2
##  9  11.1     80   22.6
## 10  11.2     75   19.9
## # … with 21 more rows
## # ℹ Use `print(n = ...)` to see more rows
```



Basic Plots

Scatterplot: girth and height of trees



- Use plot() function to create a scatterplot.
- By default, the first argument is along the x-axis and the second argument is along the y-axis.
- Add colour to the plot using the "col" argument.
- Add a title to the plot using the "main" argument.
- Type ?plot to see other arguments in the function. Make changes to your plot.

  Change the plot character: use the "pch" argument. See here for various characters.

  Change X and Y axis label: "xlab", "ylab"

  Change background colour: "bg"

Line chart & others

- Check here for a good tutorial on plotting line charts in R.
- Check here for a tutorial on preparing some other types of charts.

References

- The R guide: https://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf
- R for Beginners: https://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- An introduction to R: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
- Brunsdon and Comber, Ch 2: https://bookdown.org/lexcomber/brunsdoncomber2e/Ch2.html