Heave away:
Draw or name something you would find hard to live without while far from home

We will start at 9:00 P / 10:00 M / 11:00 C / 12:00 E / 13:00 A / 13:30 N

Please take a moment to ensure that you have downloaded course materials for today, refresh your beverage, and / or network with us.

TDU

Introduction to R

For Public Health Investigations

Public Health Agency of Canada
Agence de la santé publique du Canada

Canada

https://native-land.ca

TERRITORIAL ACKNOWLEDGEMENT

Kate

Land of the Coast Salish Peoples - Squamish, Tsleil-Waututh & Musqueam First Nations (Vancouver, BC)

Joanne

Land of the Beothuk and Mi’kma’ki people (St. John’s, NL)

Temuulen

Land of the Syilx, Nlaka'pamux, and Secwépemc people (Kelowna, BC)

Eugene & Yuhui

Land of the Algonquin, Mohawk, Anishinabewaki, and Haudenosaunee people (Ottawa, ON)

Ben

Treaty One Territory, Land of the Anishinabe, Ininew, Oji-Cree, Dene, Dakota, and Métis people (Winnipeg, MB)

Acknowledgements

The materials used throughout this course have been developed and refined in collaboration with Emma Cumming (Canadian Public Health Service), Chris Mill (BC Centre for Disease Control), Naj Saqib (Data, Partnerships, and Innovation Hub), and the Canadian Field Epidemiology Program.

Their contributions to this course are appreciated!

HOUSEKEEPING

Etiquette

- Stepping away – use coffee cup, or just let us know via chat
- Device ringers off
- Mute if not speaking

File sharing platform: Github

- Pre-learning
- Virtual classroom materials
- Self-Study
- Additional resources

HOW THIS COURSE WORKS

- Virtual classroom – 12:00 pm ET for up to 3.5 hours
  - Learning activities and dedicated time to work on exercises
  - Opportunity for learners to ask questions and get help with troubleshooting

- Self-study
  - Revisit pre-learning resources as needed
  - Complete practical exercises

- Pre-learning
  - Required to build basic R skills for this course which focuses on application of skills in the context of public health investigations

Critical R Skills

- **Installation of R and RStudio**
  - Install necessary packages for analysis
- **Fundamentals of Data Analysis in R**
  - Import data from various formats
  - Clean and process data for analysis
  - Handle dates and date formats
  - Visualize results using ggplot
- **Resources for Troubleshooting and Learning**
  - Identify resources for troubleshooting R code
  - Learn more about R

How this course works

Philosophy:

- Learners can only truly gain these technical skills and proficiency through application (and possibly manufactured struggle)
- This is a shared journey; we are all on it
- If you are stressed, we've led you astray – take a break and make note of what the problem is and let us know
- Course facilitators are here to support you as you apply your skills in a safe, supportive learning environment

Course Learning objectives

Learners will be able to (in R):

1. Carry out data cleaning and processing, and descriptive epidemiological analyses (incl. commonly used data visualisations);

2. Create automated data products (e.g., epidemiologic summaries);

3. Design and carry out a data collation plan that is consistent with proposed analysis plan;

4. Explain when it is most appropriate to program analyses and automate tasks using R;

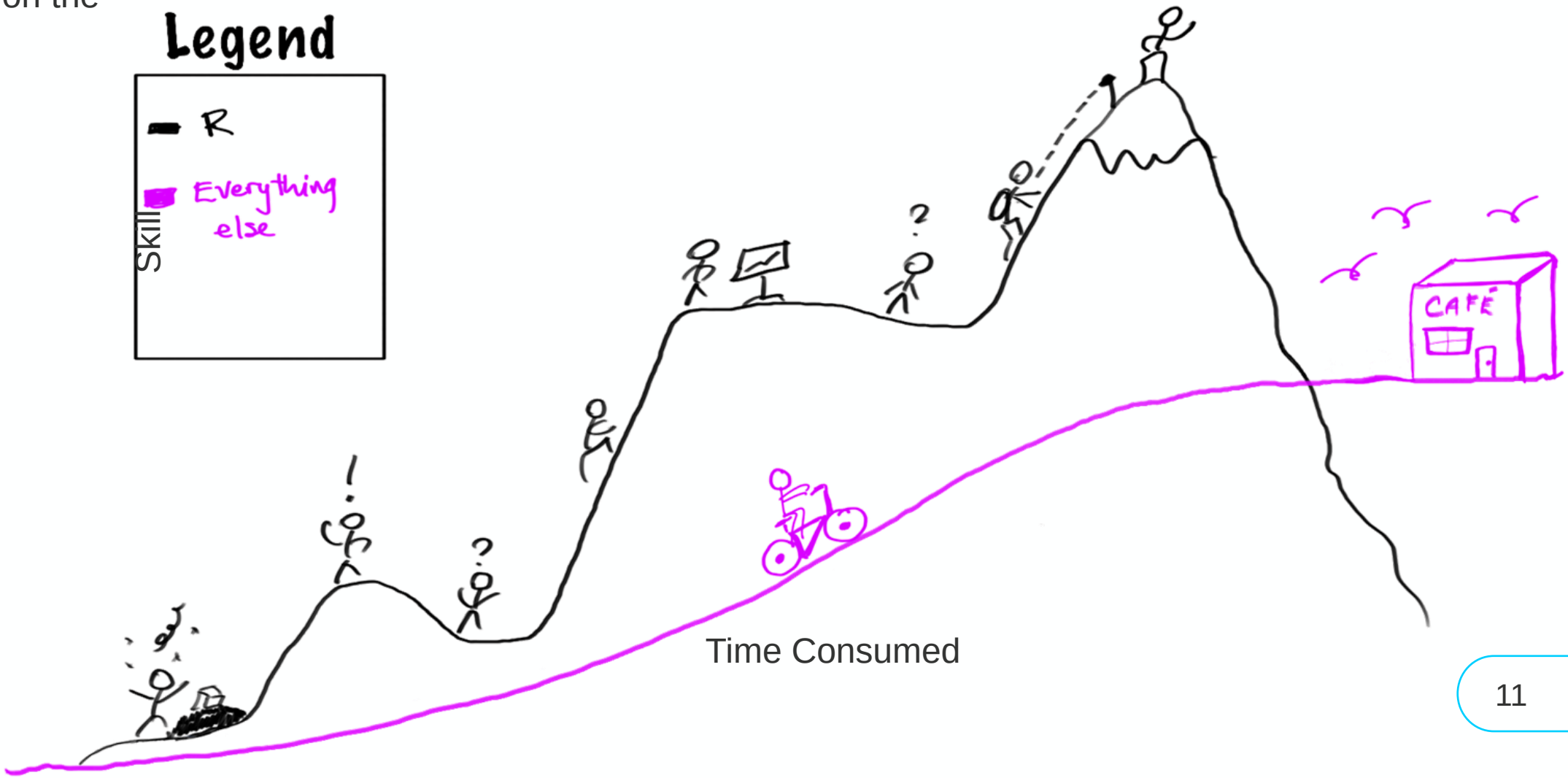
5. Find and appraise possible solutions to R programming challenges.

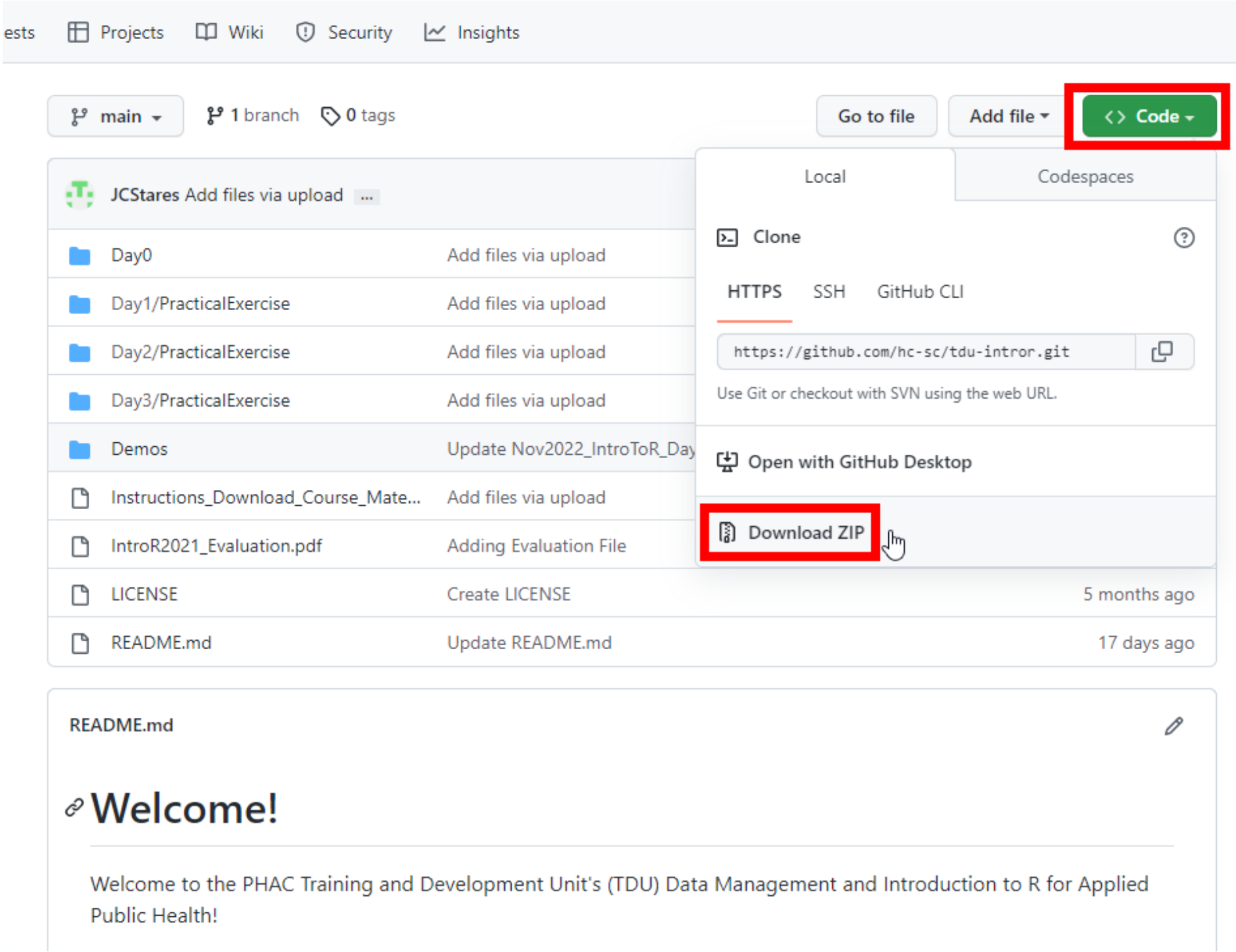
Course goals

- For novice users to have the confidence and basic skills required to start integrating R into their day-to-day work;
- To demystify and clearly communicate intricacies of R for those who want to use R and have been intimidated by the software in the past;
- To provide a brave place and relaxed environment for learning, practice, and discussion;
- To engage with and accommodate learning needs of individuals who are more experienced or advanced R users; and
- For all learners to practice writing legible, sharable code and set up efficient workspaces.

Learning Curve of Statistical Software

Use the annotate tool to place a stamp on where you feel you are on the R-learning curve





https://github.com/hc-sc/tdu-intror

https://github.com/hc-sc/tdu-intror/blob/main/Instructions_Download_Course_Materials.pdf

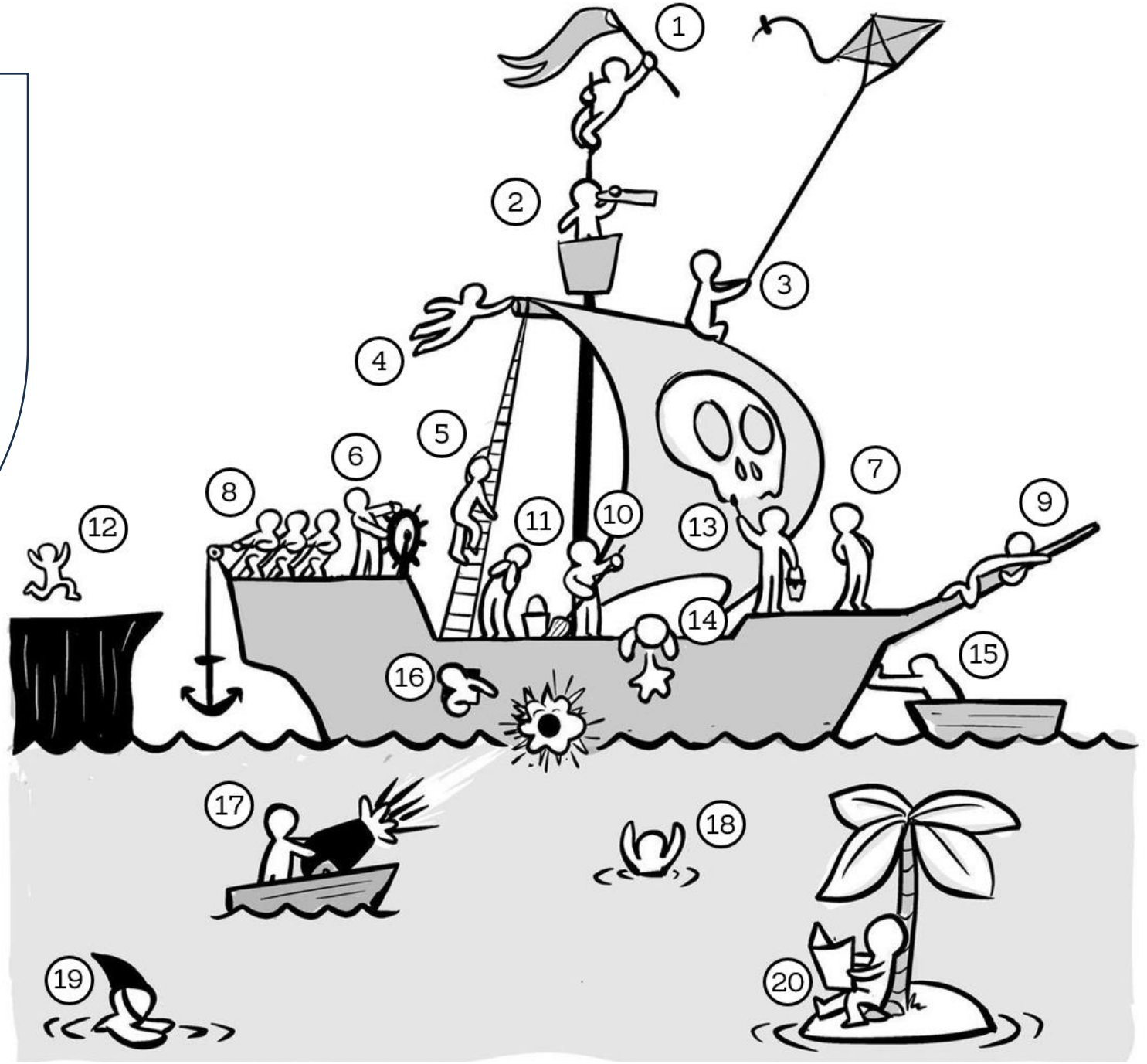
Designed by Wayhomestudio - Freepik.com

GETTING TO KNOW YOU

- When selected, please share your:
  - City where you live/work
  - What your job/role is and what team you work for
- Which character from our pirate ship (next slide) represents you best.

Designed by pch.vector on Freepik

Which character best suits you at this stage of your R journey?



Pre-Course Self-Assessment

The map

Day 1 landmarks:

- Data types and structures
- Base R versus Tidyverse
- Syntax and indexing
- Basic tips and tricks for programming analyses

Storage, Objects, and Syntax

Oh my!

R: Object-oriented programming

- R is different from other statistical software like SAS or STATA in that everything is stored as an object and there are so many different kinds!

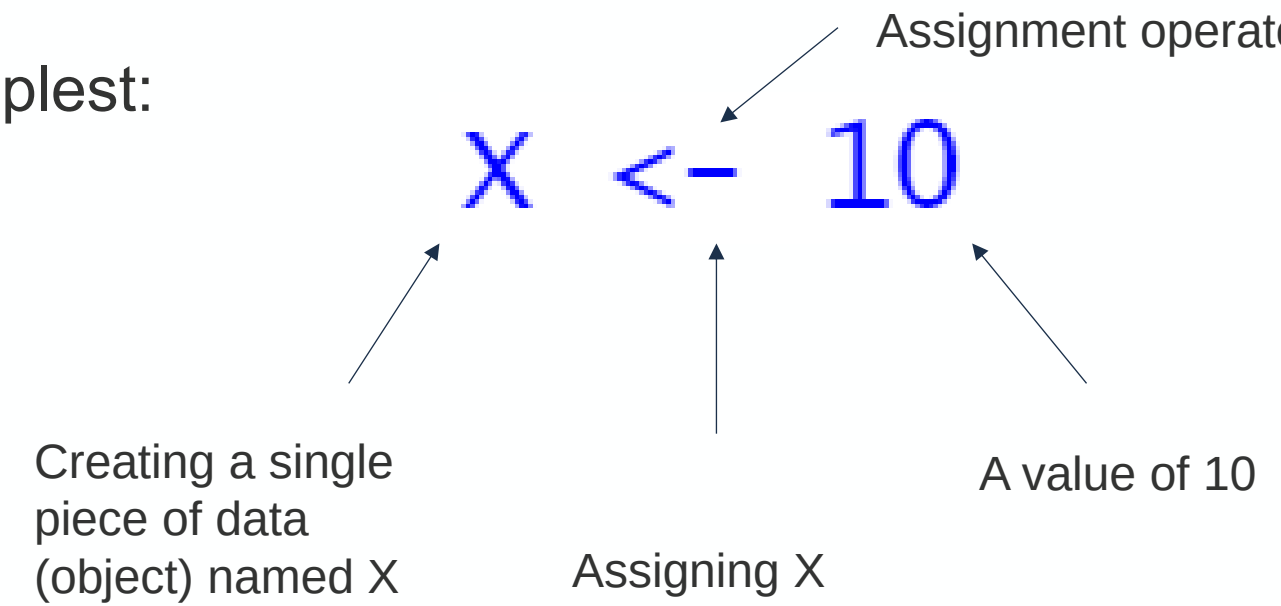
- This is one way that programming in R is a little counterintuitive for users of other statistical software

- Similar to other statistical software, we write programs to interact with these objects
  - Form and reshape them into useful objects
  - Summarise the contents of those objects
  - Present summaries in a meaningful way

Assigning objects in R

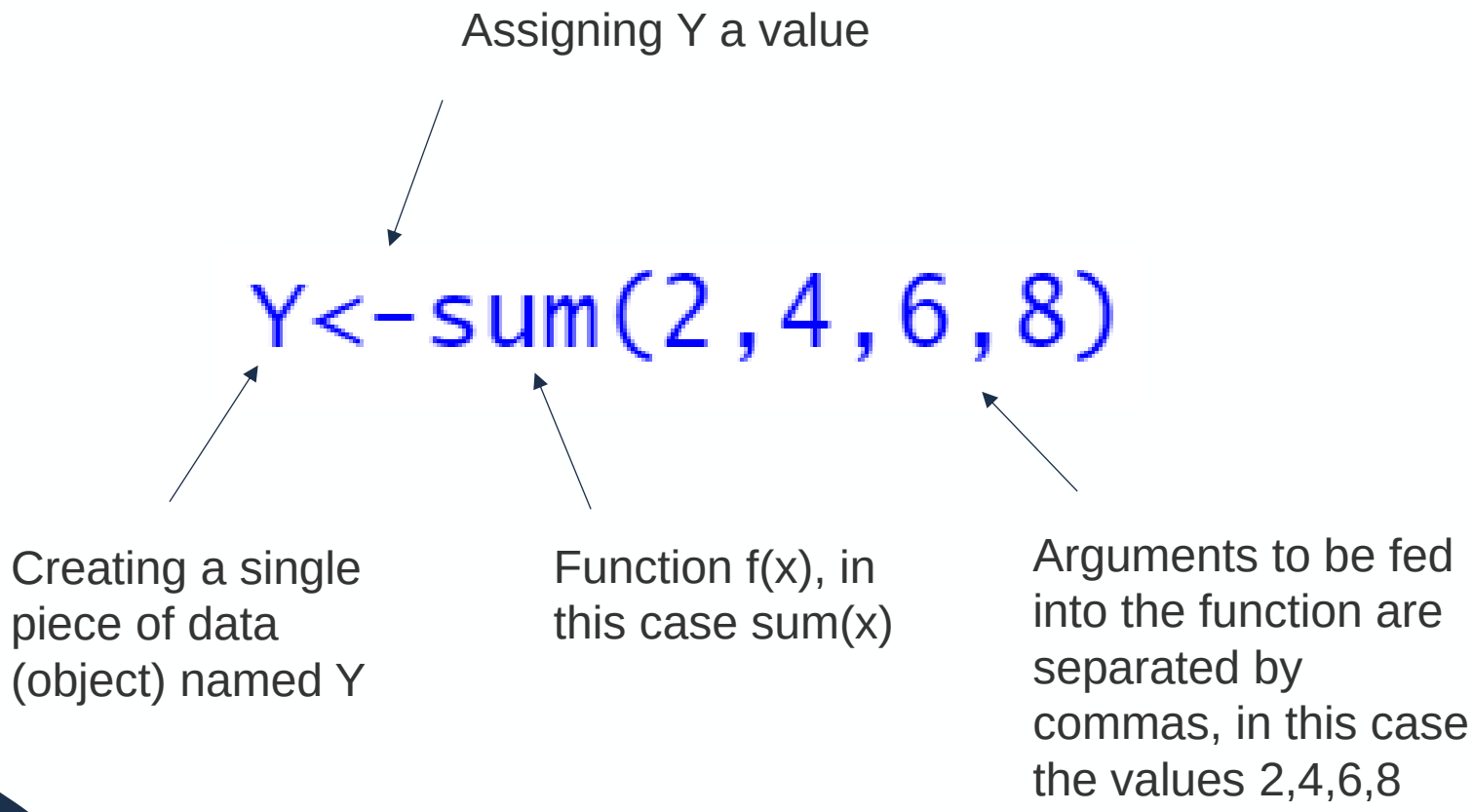
- We use an **assignment operator** (depicted as an arrow) to assign a "value" to a "name"

- At it's simplest:



Objects can be the result of functions

- Basic function structure:



Objects in R: Data types and data structures

DATA TYPES

- Character
- Numeric
- Integer
- Logical
- Complex

DATA STRUCTURES

- Vectors
- Matrix
- Data frame
- List
- Factors
- Arrays

Special data types: Dates

What about dates?

- Dates are stored as integers
- R uses a default date of origin (Jan 1, 1970) and allows user to chose a different date of origin (or day # 0) if desired

```
> as.Date(0)
[1] "1970-01-01"
> as.Date(2)
[1] "1970-01-03"
> as.Date(2, origin = "2025-01-01")
[1] "2025-01-03"
```

- Allows for operations and calculations based on dates

```
> as.Date(0) + 365
[1] "1971-01-01"
```

- Dates stored as characters (i.e., “2013-09-20”) can be converted to a numeric date using as.Date()

```
> as.Date("2013-09-20")
[1] "2013-09-20"
```

Reading dates into R

- Dates are stored as integers and made readable by applying a format

- Some extra handling is usually involved with these data
  - as.Date() ← a function to recode character data to numeric dates with a default format

```
> x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")
> z <- as.Date(x, "%d%b%Y")
> z
[1] "1960-01-01" "1960-01-02" "1960-03-31" "1960-07-30"
> mode(z)
[1] "numeric"
```

Formatting dates in R (1)

- Formatting:

Symbol	Meaning	Example
%d	day as a number (0-31)	01-31
%a %A	abbreviated weekday unabbreviated weekday	Mon Monday
%m	month (00-12)	00-12
%b %B	abbreviated month unabbreviated month	Jan January
%y %Y	2-digit year 4-digit year	07 2007

Source: http://www.statmethods.net/input/dates.html

```
> x <- as.Date("1960-07-30")
> format(x, format="%b%d,%Y")
```

Type in the chat what the output for this command will produce

Formatting dates in R (2)

- Formatting:

Symbol	Meaning	Example
%d	day as a number (0-31)	01-31
%a %A	abbreviated weekday unabbreviated weekday	Mon Monday
%m	month (00-12)	00-12
%b %B	abbreviated month unabbreviated month	Jan January
%y %Y	2-digit year 4-digit year	07 2007

Source: http://www.statmethods.net/input/dates.html

```
> x <- as.Date("1960-07-30")
> format(x, format="%b%d,%Y")
[1] "Jul30,1960"
```

Datetime objects in R

- Date and time objects are a little different and not touched on in this course specifically.

- For reference:
  - POSIXct() and POSITXlt() ← functions to work with objects that include date and time
  - ISOdatetime() ← to create date-time from numeric data

Data structures in R

Data structures

DATA STORAGE: VECTOR

- 1D – single or multiple values
- Can store multiple data types:
  e.g., logical, numeric, character, etc.

```
> x <- c(1, 2, 3)
> str(x)
 num [1:3] 1 2 3
```

DATA STORAGE: DATAFRAME

- 2D – rows and columns
- Stores multiple data types:
  e.g., logical, numeric, character, etc.

```
> data.frame(x, y)
  x y
1 1 4
2 2 5
3 3 6
```

Data structures: Vectors

- Vectors are the simplest data structure
  - Storage of just a single value

```
Y<-sum(2,4,6,8)
```

  - Storage of multiple values

```
X <-c("apple", "orange", "pear", "grape")
```

  - Storage of any data type
    - character (text)
    - numeric and integer (real numbers)
    - logical (TRUE or FALSE)
    - complex (imaginary numbers)

```
> mode(X)
[1] "character"
> length(X)
[1] 4
```

```
> mode(Y)
[1] "numeric"
> length(Y)
[1] 1
```

Data frames and matrices

DATA STORAGE: DATAFRAME

- 2D – rows and columns
- Stores multiple data types: e.g., logical, numeric, character, etc.
- More functional than a matrix

DATA STORAGE: MATRIX

- 2D – rows and columns
- Stores data only of a single data type
- Less flexible than a dataframe

Data structures: Matrix

- Matrices are 2D structures that contain rows and columns of data
- Must contain all the same type of data (e.g., all integers, or all characters)

```
> new.matrix <- matrix(data=c(1,2,3,4,5,6,7,8,9), nrow=3, ncol=3, byrow=TRUE)
> new.matrix
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Data structures: Data frame

- Dataframes are also 2D structures that contain rows and columns of data
- Allows for combination of multiple data types
- A workhorse data structure, more flexible than matrices

```
> A <- c('Cat','Dog','Rabbit','Hamster')
> B <-  as.Date(c('2020-11-1','2020-3-25','2020-3-14', '2020-9-30'))
> C <- c(TRUE,FALSE,FALSE,TRUE)
> mydata <- data.frame(A,B,C)
> mydata
        A          B     C
1     Cat 2020-11-01  TRUE
2     Dog 2020-03-25 FALSE
3  Rabbit 2020-03-14 FALSE
4 Hamster 2020-09-30  TRUE
```

```
> str(mydata)
'data.frame':   4 obs. of  3 variables:
 $ A: chr  "Cat" "Dog" "Rabbit" "Hamster"
 $ B: Date, format: "2020-11-01" "2020-03-25" "2020-0
 $ C: logi  TRUE FALSE FALSE TRUE
```

Special data structures: Factors (1)

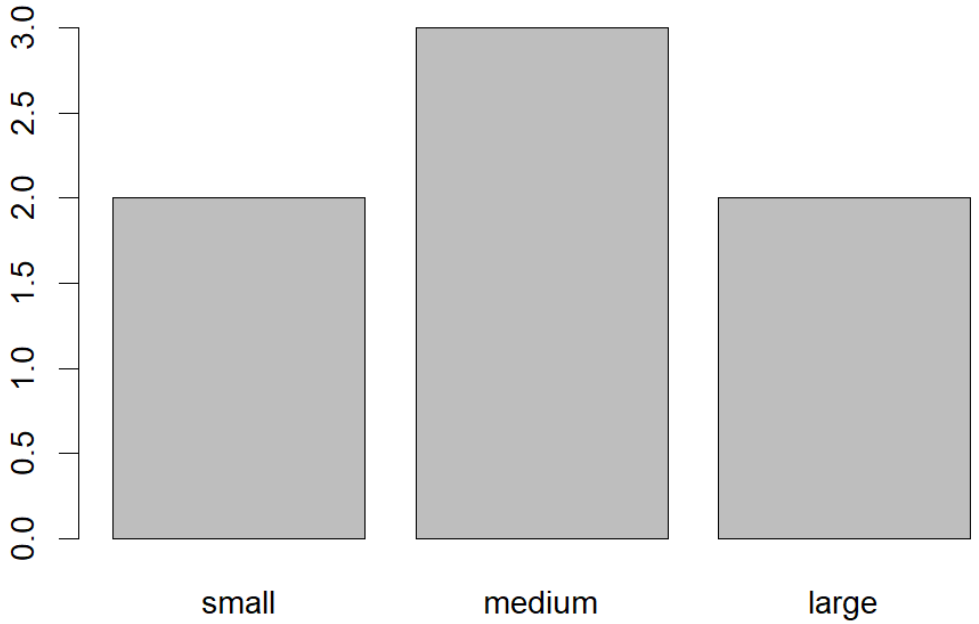
- Factors are data structures which are used to group values into categories or levels (ordered)

```
> sizes <- c(1, 2, 3, 2, 1, 3, 2)
> sz.factor <- factor(sizes,
+                     levels = c(1, 2, 3),
+                     labels = c("small",
+                                "medium",
+                                "large")
+ )
```

Special data structures: Factors (2)

- Factors are data structures which are used to group values into categories or levels (ordered)

```
> plot(sz.factor)
```



Data structures: Lists

- Lists are one of the most flexible data structures and thus can be the most complex to work with

- It is possible to have a list of individual values (i.e., vector), a list of several vectors, a list of matrices, data frames, or even a list of lists!

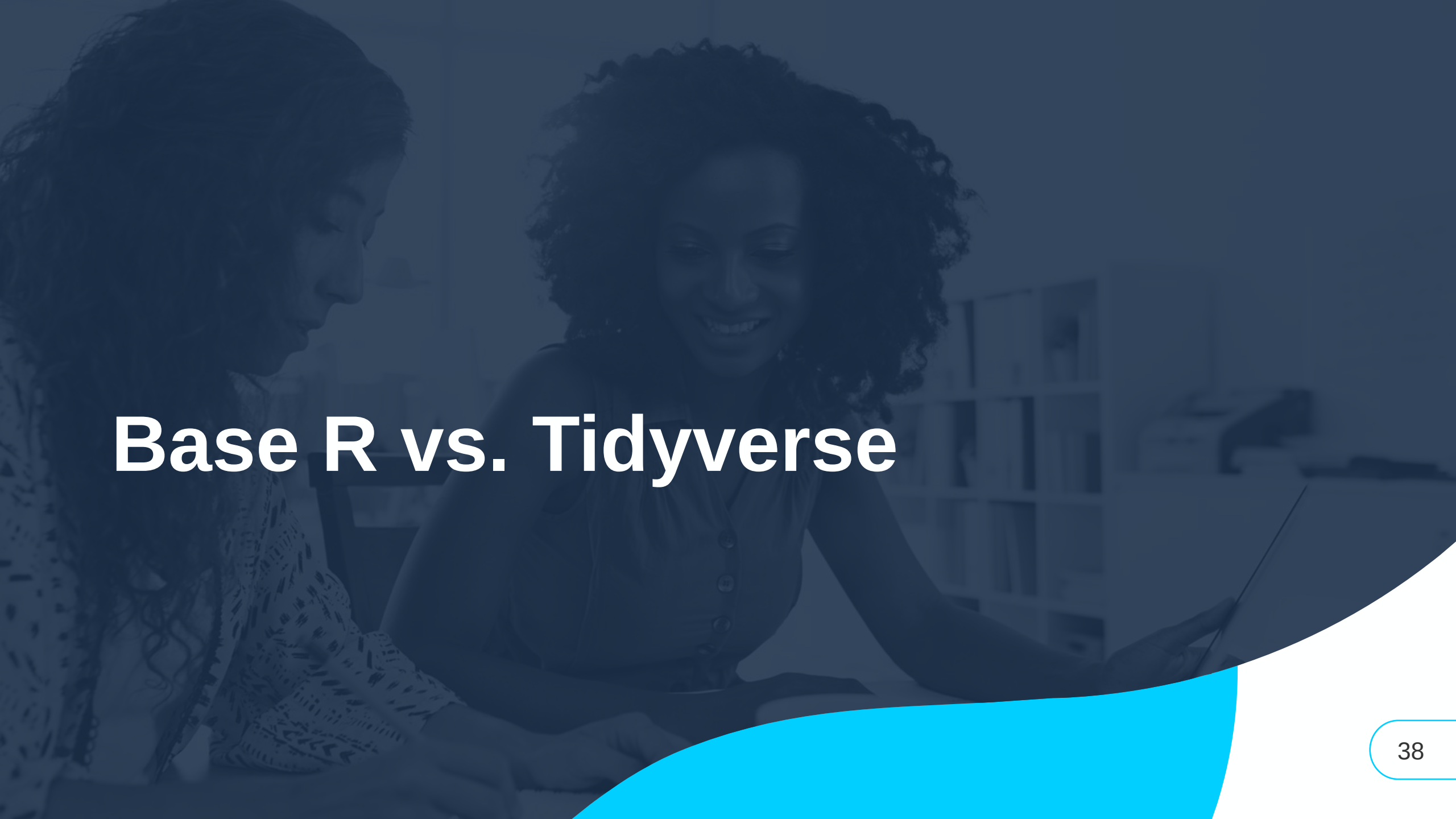
```
> D <- c('Mushroom', 'Lichen', 'Moss')
> E <- matrix(c(10,22,26,14,5,63), nrow=3)
> F <- data.frame(D,E)
> G <- list(D,E,F)
> G
[[1]]
[1] "Mushroom" "Lichen"   "Moss"

[[2]]
     [,1] [,2]
[1,]   10   14
[2,]   22    5
[3,]   26   63

[[3]]
         D X1 X2
1 Mushroom 10 14
2   Lichen 22  5
3     Moss 26 63
```

Data structures: Arrays

- Arrays are data structures that can store data in multiple dimensions
- Very useful under certain circumstances
- Will not touch on this type of object in this course



Base R vs. Tidyverse

R Tidyverse

- A data-science “philosophy” developed by scientists at Posit (formerly Rstudio)
- Provides a **unified approach** to data manipulation, visualization, and analysis
- Promotes organizing data in a consistent format:
  - Each **column** represents a **variable**
  - Each **row** represents a unique **observation**
  - Each **cell** represents a **single value**
- Collection of R packages that function together, using a common approach to syntax:

%>%

Comparing base R and Tidyverse

BASE R

- Basic operations
  - E.g., Use R like a calculator
- Basic, built in functions (no need to install packages):
  - To see a list of Base R functions, run library(help="base")
- Chaos – different data structure and sometimes syntax across packages and functions, at times limiting reproducibility

TIDYVERSE

- A collection of packages for data science and analytics
- Consistency in syntax, data structure, etc.
- Flow across packages and functions
- Minimum set of functions that are sufficient for data science

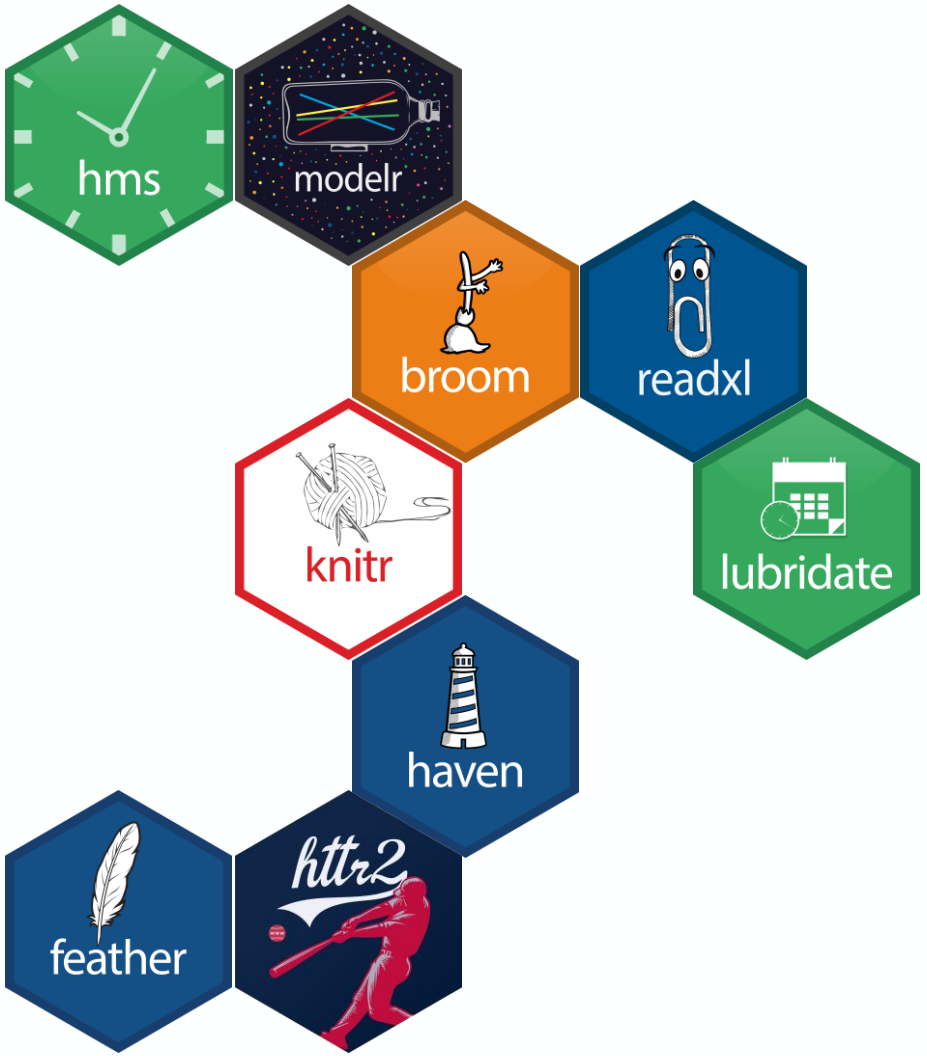
Tidyverse core packages:

- dplyr – a package for manipulating and reshaping data
- tidyr – a package to help you keep your data nice and tidy
- readr – a package to load your data tables (i.e., csv, tsv, etc.)
- purr – a package for working with functions and vectors
- tibble – a package for working with dataframes
- forcats – a package for working with factors
- stringr – a package for working with strings (character/text)
- ggplot2 – a package for creating graphics

https://www.tidyverse.org/

Tidyverse extra packages:

1. Working with specific types of vectors:
   - * *lubridate*, for working with dates
   - *hms*, for times.
2. Importing other types of data:
   - *feather*, for sharing with Python and other languages.
   - *haven*, for SPSS, SAS and Stata files.
   - *httr*, for web apis.
   - *jsonlite* for JSON.
   - *readxl*, for .xls and .xlsx files.
   - *rvest*, for web scraping.
   - *xml2*, for XML.
3. Modelling
   - *modelr*, for modelling within a pipeline
   - *broom*, for turning models into tidy data



R syntax comparison – BASE R (nested)

- Why know both? How are they different?

Base R (nested, un-nested) vs Tidyverse (%>%)

```
#Base R (nested):
bread <- bake(x=let_rise(x = knead(x = combine_ingredients(x = bowl, flour=TRUE,
water=TRUE, yeast=TRUE, salt=TRUE, sugar=TRUE), mixer = TRUE, byhand = FALSE,
addflour=TRUE), nhours = 4, rep = 2), temp = 400, minutes = 45)
```

R syntax comparison – BASE R (un-nested)

- Why know both? How are they different?

Base R (nested, un-nested) vs Tidyverse (%>%)

```
#Base R (nested):
bread <- bake(x=let_rise(x = knead(x = combine_ingredients(x = bowl, flour=TRUE,
water=TRUE, yeast=TRUE, salt=TRUE, sugar=TRUE), mixer = TRUE, byhand = FALSE,
addflour=TRUE), nhours = 4, rep = 2), temp = 400, minutes = 45)
```

```
#Base R (unnested):
mixture <- combine_ingredients(x = bowl, flour=TRUE, water=TRUE, yeast=TRUE,
salt=TRUE, sugar=TRUE)
dough <- knead(x = mixture, mixer = TRUE, byhand = FALSE, addflour=TRUE)
proofed <- let_rise(x = dough, nhours = 4, rep = 2)
bread <- bake(x=proofed, temp = 400, minutes = 45)
```

R syntax comparison – Tidyverse

- Why know both? How are they different?

Base R (nested, un-nested) vs Tidyverse (%>%)

```
#Base R (nested):
bread <- bake(x=let_rise(x = knead(x = combine_ingredients(x = bowl, flour=TRUE,
water=TRUE, yeast=TRUE, salt=TRUE, sugar=TRUE), mixer = TRUE, byhand = FALSE,
addflour=TRUE), nhours = 4, rep = 2), temp = 400, minutes = 45)
```

```
#Base R (unnested):
mixture <- combine_ingredients(x = bowl, flour=TRUE, water=TRUE, yeast=TRUE,
salt=TRUE, sugar=TRUE)
dough <- knead(x = mixture, mixer = TRUE, byhand = FALSE, addflour=TRUE)
proofed <- let_rise(x = dough, nhours = 4, rep = 2)
bread <- bake(x=proofed, temp = 400, minutes = 45)
```

```
#Tidyverse:
bread<-
 bowl %>%
  combine_ingredients(flour=TRUE, water=TRUE, yeast=TRUE, salt=TRUE, sugar=TRUE) %>%
  knead(mixer=TRUE, byhand=FALSE, addflour=TRUE) %>%
  let_rise(nhours=4, rep=2) %>%
  bake(temp=400, minutes=45)
```

Break - Trivia: Famous Shipwrecks

1. This ship was wrecked during a storm on Lake Superior in 1975 and was later immortalised in a Gordon Lightfoot song. — A - Valencia
2. This ship was one of two long lost and long searched for British ships in the Canadian Arctic. It was finally located by searchers in Nunavut in 2016 following a tip from local Inuk Sammy Kogvik. — B - Truxtun
3. This ship was wrecked off the west coast of Vancouver Island in 1906 and the tragedy led to the development of a rescue trail for shipwreck survivors. In the present day, this trail is known as one of BC's most iconic hiking trails, the West Coast Trail. — C - Edmund Fitzgerald
4. This ship was one of two American Navy vessels run aground and lost on the south coast of Newfoundland during a winter storm in 1942. — D – Terror

Share your answers in the chat!

Please return by:
10:10 Pacific / 11:10 Mountain / 12:10 Central / 13:10 Eastern / 14:10 Atlantic / 14:40 NFLD

TDU

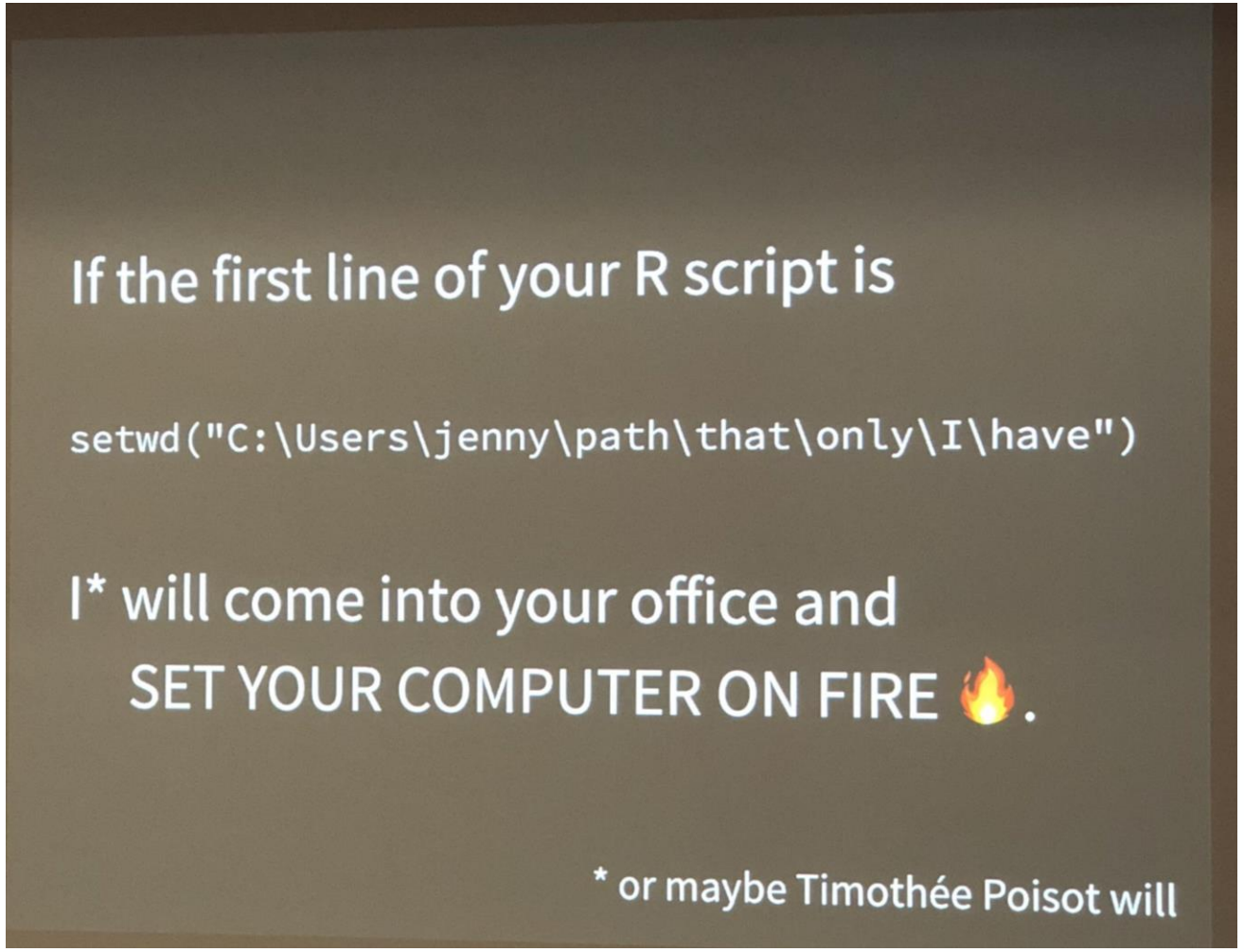
Setting yourself up for success: Demo

R-Projects && here()

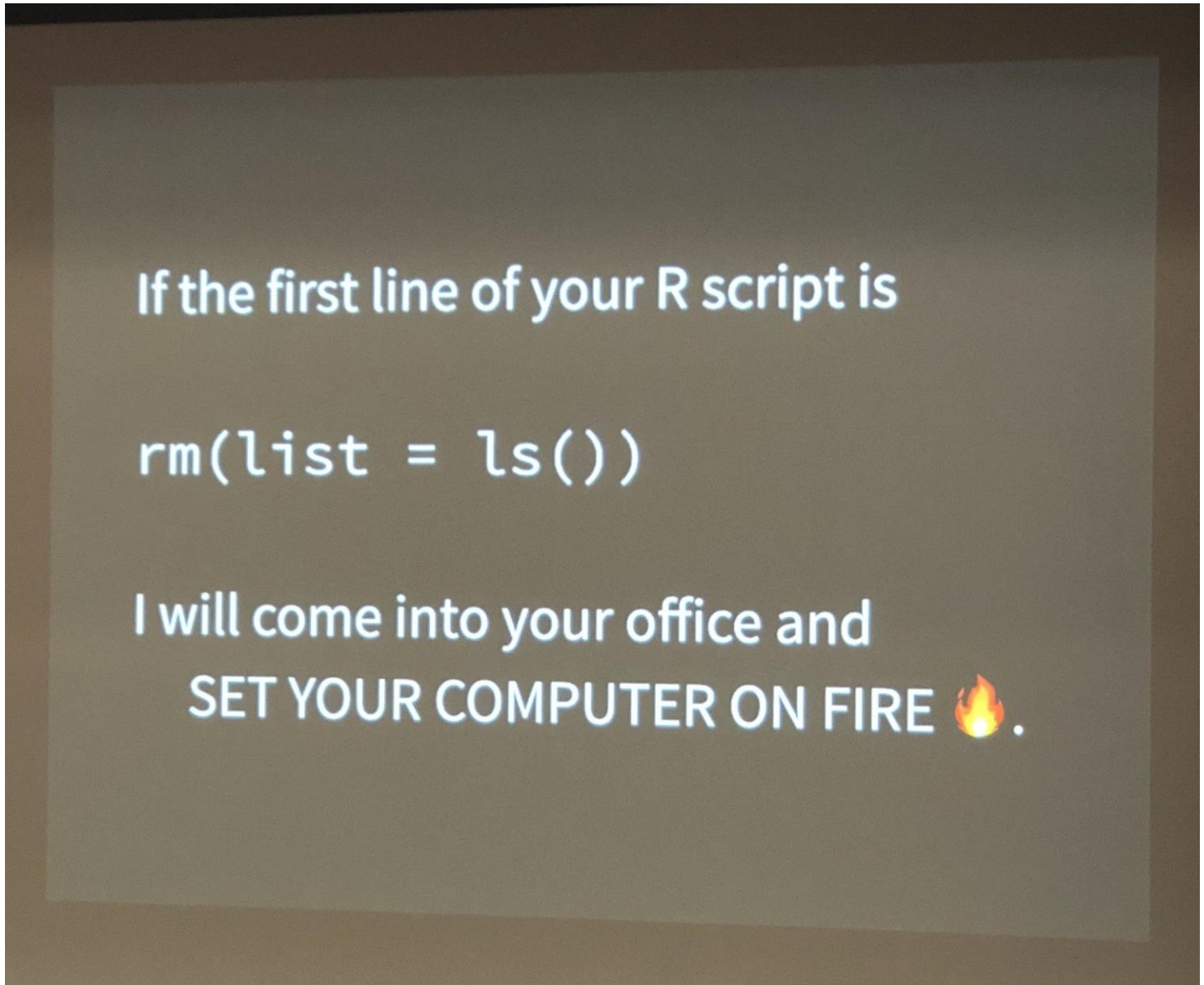
Public Health Agency of Canada
Agence de la santé publique du Canada

Canada

A special message from the data scientists at Rstudio:



A special message from the data scientists at Rstudio:



Tips to avoid a flammable computer:

1. Don’t use absolute file pathways

2. Don’t set the first line of your script to rm(list = ls())

Why not??

Tips to avoid a flammable computer:

1. Don’t use absolute file pathways

2. Don’t set the first line of your script to `rm(list = ls())`

Why not??

1. Don’t use absolute file pathways
   a) Different system folder names based on user
   b) Different operating systems
   c) Different working directories
2. Don’t set the first line of your script to `rm(list = ls())`
   a) What if someone wants to source() your script in the middle of their analysis?

Tips to avoid a flammable computer:

Solutions?

1. Use a system that allows for relative file paths, e.g., here() package

2. Set up your R projects using … R Projects!

Tips to avoid a flammable computer:

Solutions?

1. Use a system that allows for relative file paths, e.g., here() package
   a) Uses file pathway relative to a ‘root’ location on your computer
   b) Similar to file.path, allows you to ignore nuances related to operating system (e.g., agnostic to Windows, MacOS, Linux distros)
   c) Results in much easier portability between project stakeholders

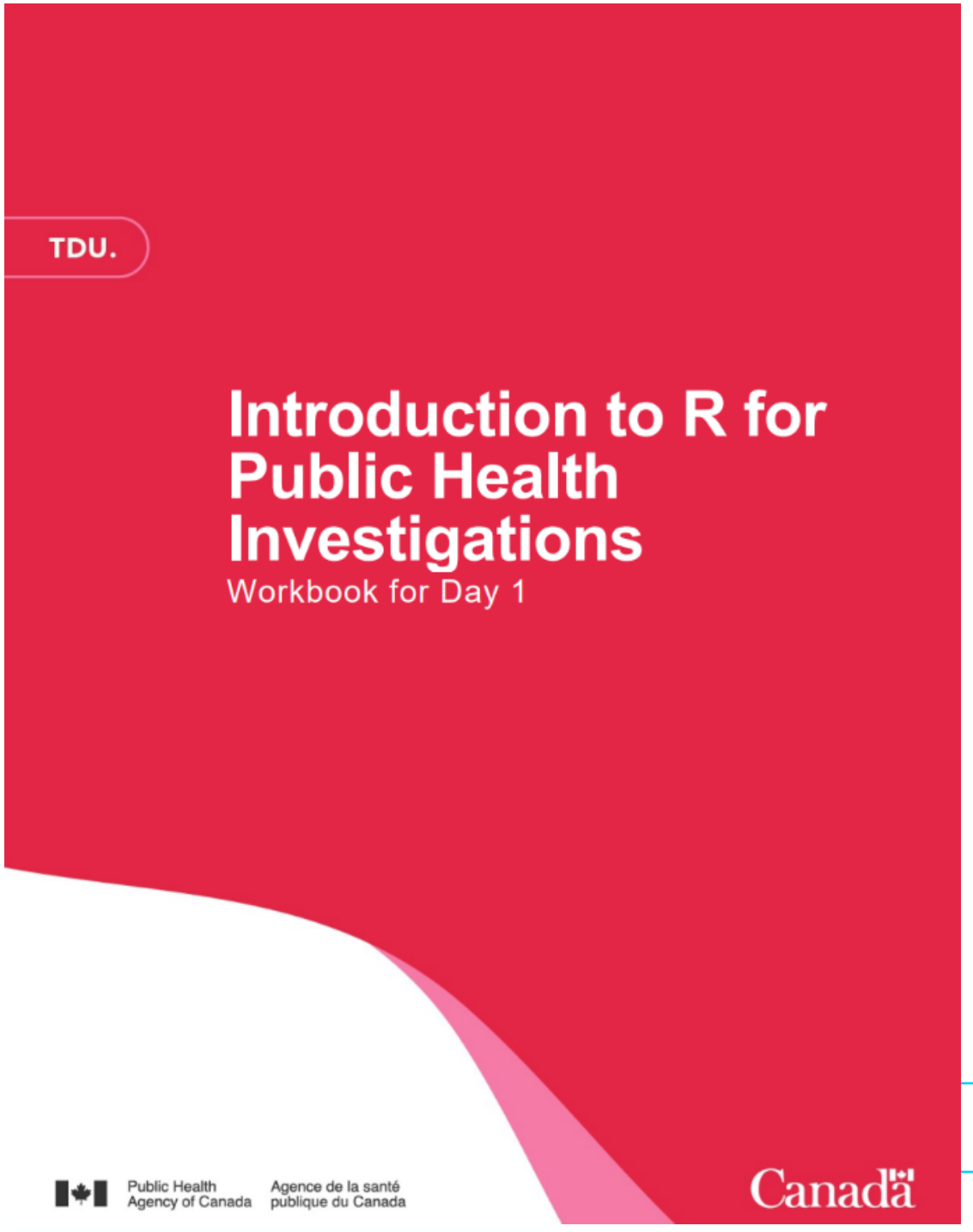
2. Set up your R projects using … R Projects!
   a) RStudio has a built-in project manager: .Rproj
   b) R projects set your working directory for you automatically, and provide the anchor point for the here() package
   c) R projects can be customized with options including (1) saving your history between sessions; (2) retaining your global environment; and much more!

Independent Study and Drop-In Office Hours

Practice makes perfect

Exercise 1

- Scenario: It is December 2020, and you are working in the Office of the Provincial Health Officer in British Columbia. The Provincial Health Officer wants a repeat analysis from the PHAC open access COVID-19 line list to inform situational awareness and decision making.
- For this request you will:
  - Setup your workspace
  - Load data
  - Clean the data including dates
  - Create new variables
  - Visualize and summarise the data
  - Automate the results by writing a script



Approach

- We recommend:
  - **Novice users:** Use the workbook and R script(s) provided on GitHub as a guide. Run the available scripts and prioritize your understanding what each chunk of code and functions used are doing. Do not worry about being able to write or debug code.
  - **Beginner/Intermediate users:** R code is provided as a screen capture image in the workbook. You should have sufficient understanding of coding to get a general sense of what the code is doing by reading it or doing a little research. We would like you write the code out from the guide as you progress through the scenario. Cross reference to the R script(s) provided on GitHub if you encounter any tangly problems.
  - **Advanced users:** We encourage you to try writing your own code where you like and contrast it with the code used for the exercise, and to help your peers as questions arise. Cross reference to the R script(s) provided on GitHub if you encounter any tangly problems.

Independent Study and Drop-In Office Hours

- We will provide a short demo of basic functions and walk through the first steps of the exercise to ensure that everyone is able to get started (optional attendance)
- You are free to stay in the virtual classroom or leave while you work through the exercise
- We'll be here in the virtual classroom to answer your questions as they arise
- Please return by 12:15 Pacific / 13:15 Mountain / 14:15 Central / 15:15 Eastern / 16:15 Atlantic / 16:45 NFLD
- Remember to take a 15-minute break before you return

Demo

Trivia: Public health at sea

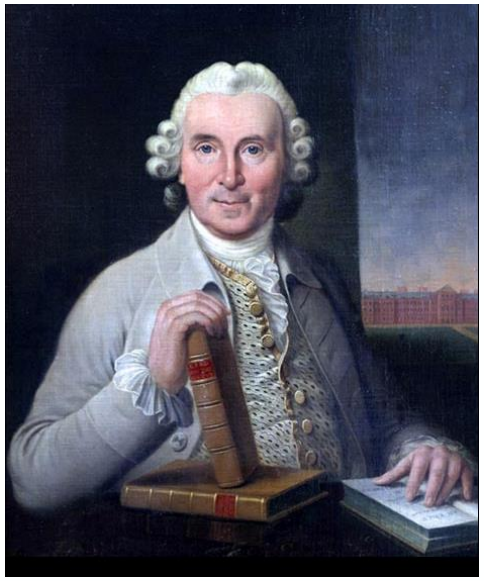
The first recorded controlled clinical trial was established in 1747 by ship surgeon James Lind to identify a treatment for which illness?

A - Yellow fever

B – Lead toxicity

C - Scurvy

D – Typhus



Share your answer in the chat!

Please return by:

12:15 Pacific / 13:15 Mountain / 14:15 Central / 15:15 Eastern / 16:15 Atlantic / 16:45 NFLD

Questions?

Wrap-up

Reminder

- Complete exercise 1 as we will debrief when we meet tomorrow

- Please reach out to your course facilitators if you have questions

Feedback for day 1

- Please take a moment to answer a few questions
- https://www.slido.com/
- #IntroR2025