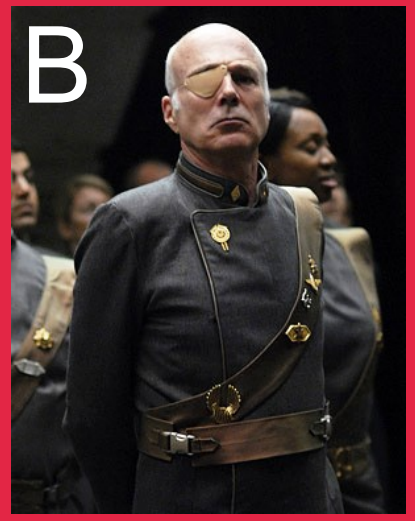
EXTREME MAKEOVER: EYEPATCH EDITION!



Who wore it best? In the chat, see if you can name the character in each picture, and indicate who you think deserves the award for: best eyepatch!

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!

TDU.

INTRODUCTION TO R

For Public Health Investigations

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

Canada

A MOMENT TO REFLECT ON THE FIRST PEOPLES ON THE LAND WE ARE JOINING FROM TODAY

Nevena: Granby, QC
Land of the
W8banakiak (Abenaki) peoples of
the W8banaki (Waban-Aki) Confederacy
(unceded)

Joanne: St. John's, NL
- NFLD: Mi'kmaq and Beothuk
- Labrador: Innu (Nitassinan), Inuit (Nunatisvut), Southern Labrador Inuit (NunatuKavut)

Emilia: Montreal, QC
Kanien'kehá:ka Nation, with ties to the Haundenosaunee Confederacy and Anishinaabeg peoples (unceded)

https://native-land.ca/

Lynn: Edmonton, AB
Treaty Six Territory, Land of Nêhiyaw / Cree, Dené, Anishinaabe / Saulteaux, Nakota Isga / Nakota Sioux, and Niitsitapi / Blackfoot, and Métis peoples.

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

Naj, Cass, Joëlle, Erin: Ottawa, ON
Algonquin Anishnaabe (unceded)

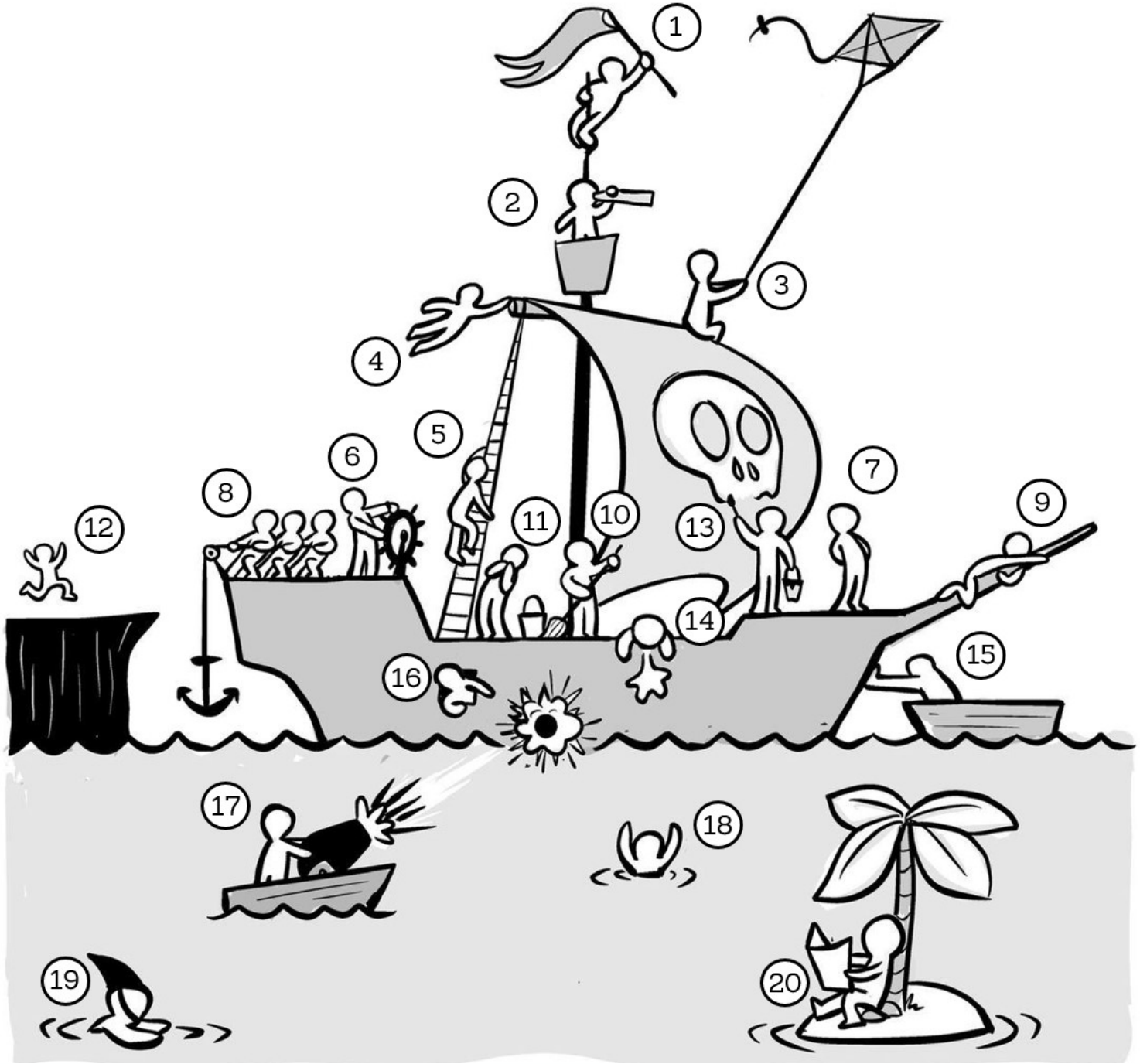
GETTING TO KNOW YOU

…And practicing zoom tricks at the same time.

- On your designated spot in the matrix, please share with the group:
  - City
  - Where you work/your position
  - Which character from our pirate ship (next slide) represents you best.
- Let’s try zoom polls with a have-you-ever question.

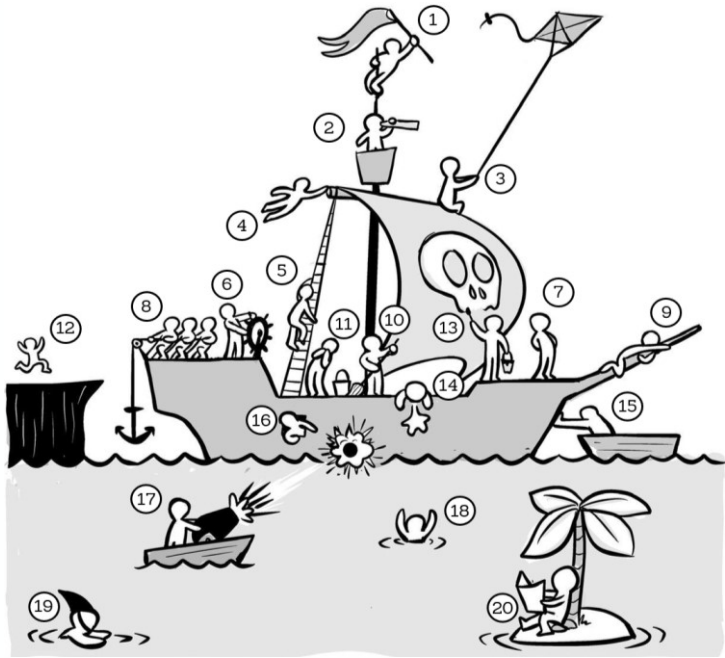
Designed by pch.vector on Freepik

- Take a minute to pinpoint which character suits you best (memorize the associated character number)



Brigitte Ho Mi Fane City: Work: Pirate #:	**Max Ferguson** City: Work: Pirate #:	**Diana Bark** City: Work: Pirate #:	**Cher Ghafari** City: Work: Pirate #:
Shane Randell City: Work: Pirate #:	**Mindy Lamer** City: Work: Pirate #:	**Renee Campbell** City: Work: Pirate #:	**Kristine Cooper** City: Work: Pirate #:
Cassandra Andrew City: Work: Pirate #:	**Penelope Gorton** City: Work: Pirate #:	**Jamie Imada** City: Work: Pirate #:	**Anton Maslov** City: Work: Pirate #:
Camilia Thieba City: Work: Pirate #:	**Wrency Tang** City: Work: Pirate #:	**Hilda Chan** City: Work: Pirate #:	**Stephanie Walsh** City: Work: Pirate #:
Carina Qiao City: Work: Pirate #:	**Erin Eales** City: Work: Pirate #:	**Zoe Alavi** City: Work: Pirate #:	**Robyn Mitchell** City: Work: Pirate #:

Use the Annotate text tool to fill in the grid!



Designed by Wayhomestudio - Freepik.com

Photo by Christina Victoria Craft on Unsplash

HOUSEKEEPING

- Optimal set-up for Zoom:
  - Disconnect from VPN
  - Microphone on mute unless you’re speaking
  - Speaker view
  - Open chat window
  - Open participant window
  - If you are experiencing significant connectivity issues, please reach out

TDU.

ZOOM FOR VIRTUAL TRAINING

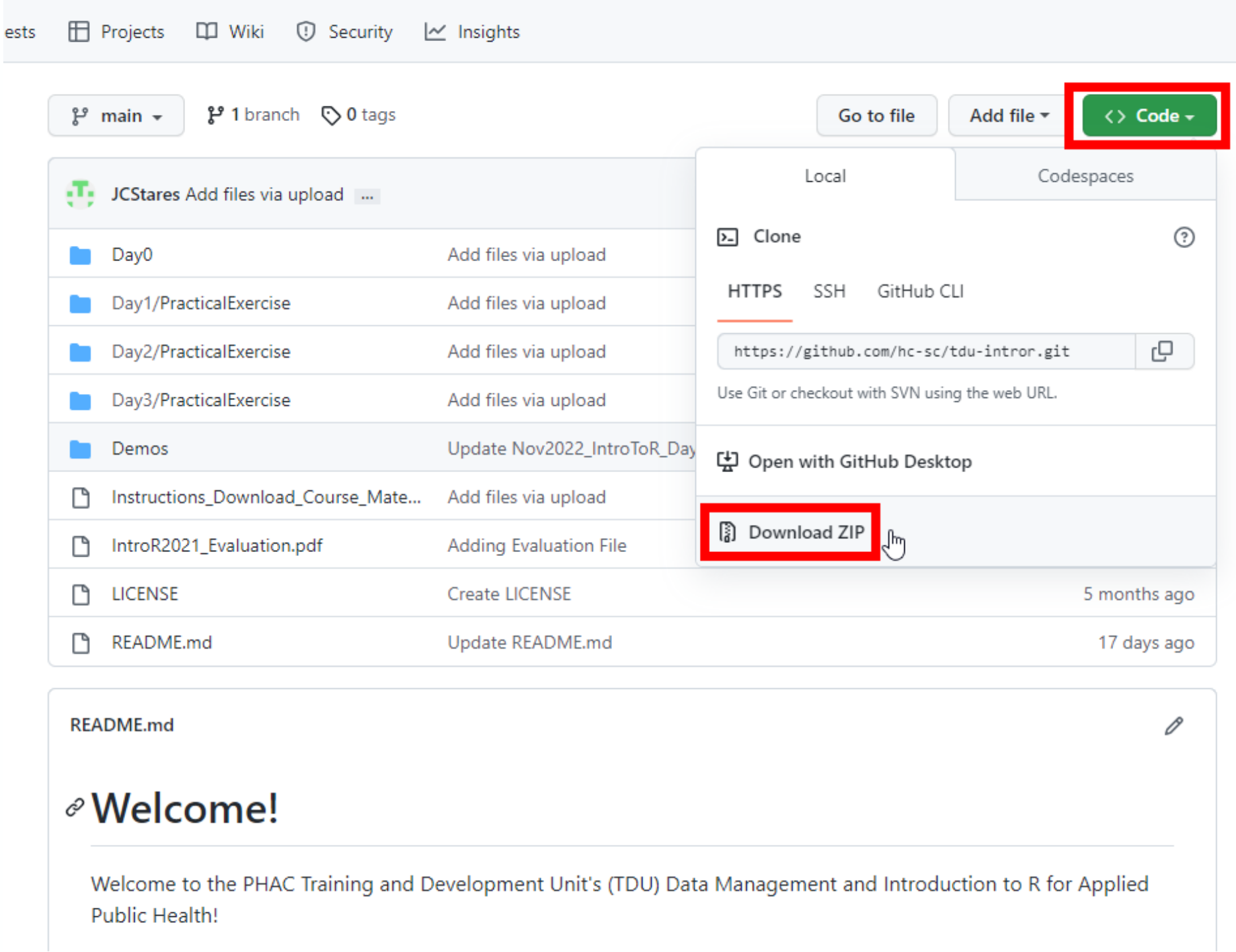
TIPS FOR SUBJECT MATTER EXPERTS, FACILITATORS, AND PARTICIPANTS

Zoom is a tool that the Training and Development Unit (TDU) will be using as a “virtual classroom” to deliver virtual training content. Zoom is accessible to persons inside or outside the Government of Canada provided that only unclassified, non-sensitive information is shared. For those joining from the Government of Canada, please be sure to disconnect from VPN prior to joining the virtual training session via Zoom.

HOUSEKEEPING

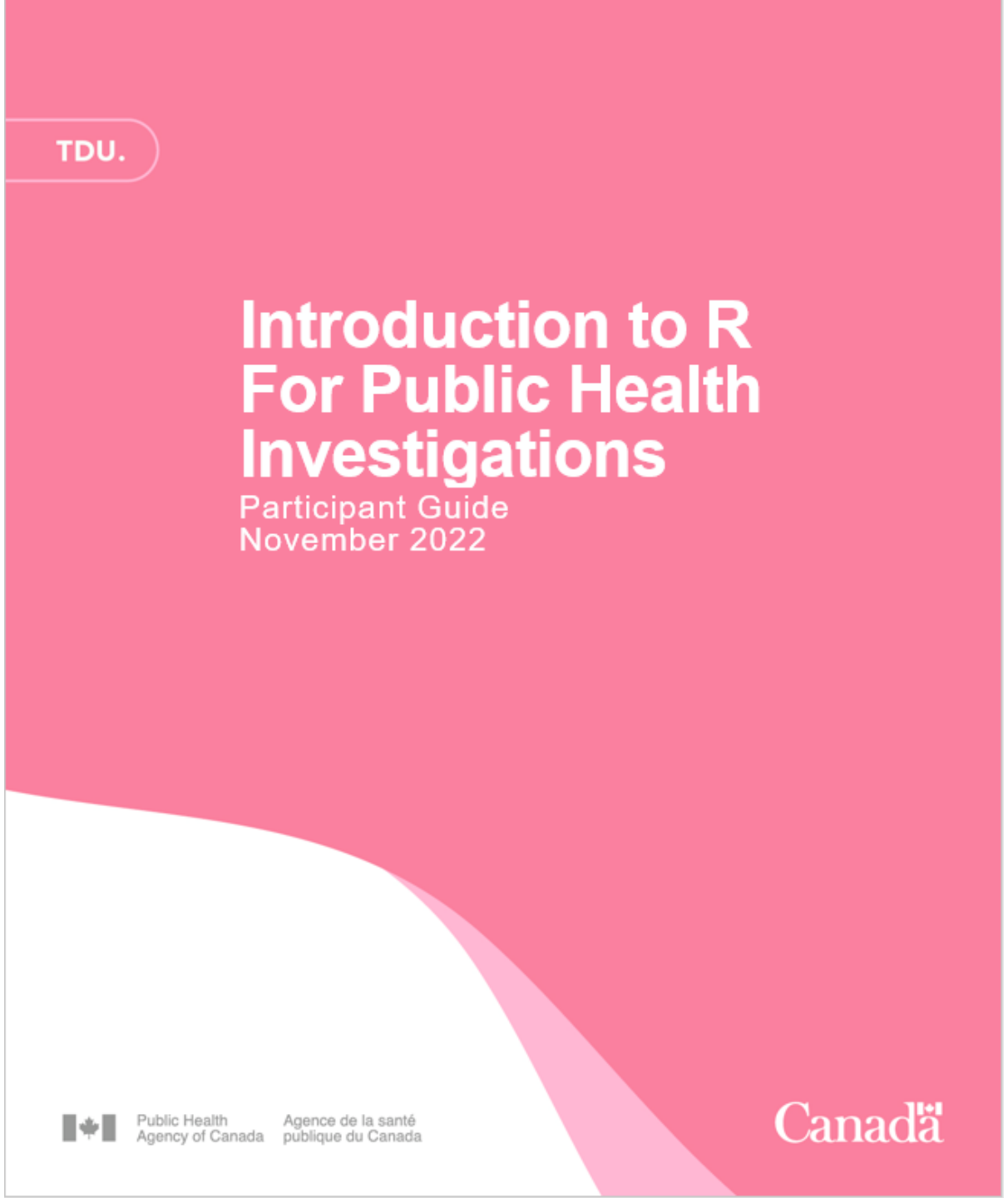
- Etiquette
  - Stepping away – use coffee cup, or just let us know
  - Device ringers off
  - Mute if not speaking

- File sharing platform: Github
  - Pre-learning
  - Virtual classroom materials
  - Self-Study
  - Additional resources

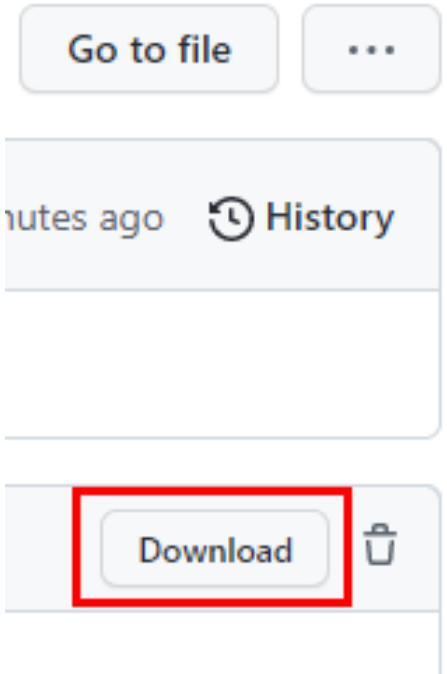


https://github.com/hc-sc/tdu-intror

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



https://github.com/hc-sc/tdu-intror/blob/main/Oct2023_IntroToR_ParticipantGuide.pdf



HOW THIS COURSE WORKS

- Virtual classroom – 11:30 am ET for up to 4 hours
  - Review of materials covered in self-study or practical exercises
  - Opportunity for learners to ask questions

- Self-study
  - Knowledge check
  - Required self-learning

- Practical exercises
  - Feel free to reach out to your facilitators and others engaged in this course via Slack!
  - Dedicated time set aside on Friday set aside for practice and support

HOW THIS COURSE WORKS

- Main pathway for learning and support - Slack:

  Slack channel for this training session

  - Phacrusers.slack.com then find the channel → **intro_r_oct2023cohort**

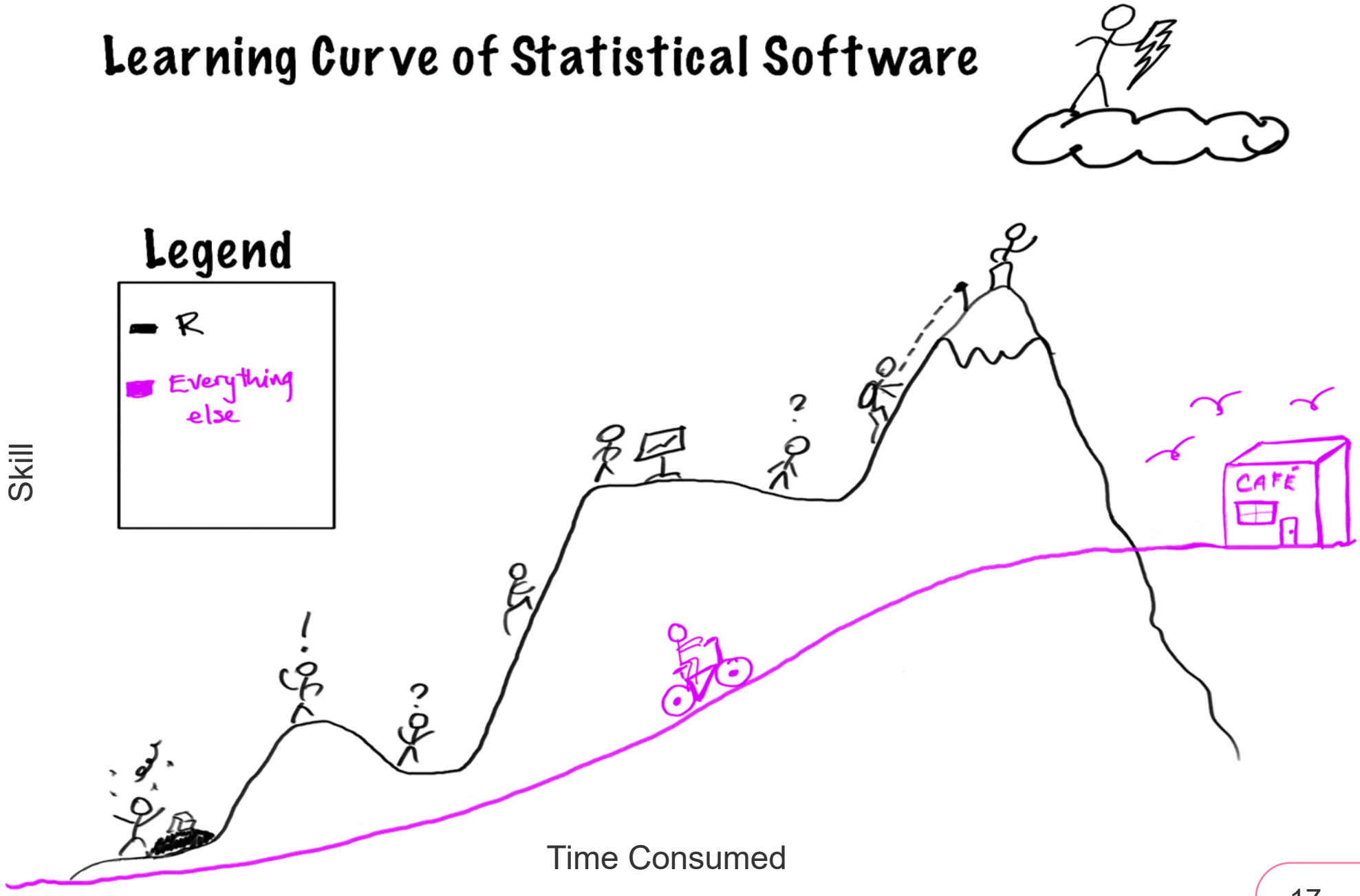
- Please use this slack channel to:
  - Connect with your peers
  - Find materials shared throughout the course
  - Ask questions about the materials/exercises to your peers, facilitators and subject matter experts
  - Help others who have questions if you know the answer!

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

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



COURSE LEARNING OBJECTIVES

- By the end of this course, learners will be able to:
  - Carry out data cleaning and processing, and descriptive epidemiological analyses (incl. commonly used data visualisations);
  - Create automated data products (e.g., epidemiologic summaries);
  - Design and carry out a data collation plan that is consistent with proposed analysis plan;
  - Explain when it is most appropriate to program analyses and automate tasks using R;
  - Find and appraise possible solutions to R programming challenges.

COURSE GOALS

- For novice users to have the confidence and basic skills required to integrate 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;
- To provide a safe 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.


Image: Designed by rawpixel.com / Freepik

TDU.

INTRODUCTION TO R

Syntax, Data Storage, and Other Essentials

Day 1

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

Canada

THE MAP

- Ten considerations for data analysis
- Data types and storage
- Syntax and indexing
- Basic tips and tricks for programming analyses

DAY 1: PRE-COURSE LEARNING

Activity	Description
Refresher	Pre-Course Self-Study Module
Foundational Concepts	10 Considerations for Data Analysis
Foundational Concepts	TDU Introduces R
Foundational Concepts	TDU Walks Through The Basics

A FRAMEWORK FOR DATA ANALYSIS

Eating an elephant (or a very large cake), one bite at a time

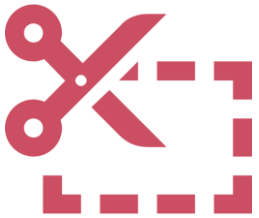
10 Considerations for Data Analysis

1. Read data into R

Consider (1) the type of data, (2) how they are stored, and (3) the size of the dataset.

2. Format the data

Long or wide format? Plan the analysis keeping in mind what your needs are.



3. Select data

Decide what data you'll need in advance from the dataset: subsets vs. filtered dataset. Consider pros and cons of the approach.

4. Modify data

Consider (1) what variables you will need, (2) calculations you will perform, and (3) algorithms needed to create the variables.

5. Link data

Assign a key variable(s) that you will link datasets on. Consider the join type you will need: inner, left, right, full, semi, or anti join.

6. Analyse data

Consider grouping data meaningfully in performing calculations and obtaining descriptive statistics. Evaluate outputs and interpret summaries.

7. Visualise data

Use tables or charts, and consider the best way to display the information.

8. Report findings

Communicate findings to those who need the information created from your work. Consider the audience you are communicating with.

9. Celebrate!

Preferably with cake!

10. Review

Review, revise, and re-do as needed. Often we can go back and simplify/optimise. This is important especially if the code will be used again.

HEAVE AWAY:
DRAW OR NAME SOMETHING YOU WOULD FIND HARD TO LIVE WITHOUT WHILE FAR FROM HOME

STORAGE, OBJECTS, AND SYNTAX

Oh my!

OBJECTS: DATA TYPES AND DATA STRUCTURES

DATA TYPES

- Character
- Numeric
- Integer
- Logical
- Complex

DATA STRUCTURES

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

OBJECTS: DATA TYPES AND DATA STRUCTURES

DATA TYPE: LOGICAL

- Values:
  - True or False
  - T or F
  - 1 or 0

DATA TYPE: COMPLEX

- Values:
  - Imaginary numbers (i, $\sqrt{-1}$)

OBJECTS: DATA TYPES AND DATA STRUCTURES

- What about dates?
  - Dates are stored as numbers
  - R allows user to chose date of origin (or day # 0)

- Allows for operations and calculations based on dates

- Dates stored as characters (i.e., “20-Oct-2002”) can be converted to a numeric date variable using a special function for this purpose

- Date is displayed by applying a format to those numbers

OBJECTS: DATA TYPES AND DATA STRUCTURES

DATA STORAGE: VECTOR

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

DATA STORAGE: DATAFRAME

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

OBJECTS: DATA TYPES AND DATA STRUCTURES

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

OBJECTS AND SYNTAX

- R is different from other programs like SAS or STATA in that **everything** is 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

- Syntax is often a challenging part of learning a new programming language

R SYNTAX

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/

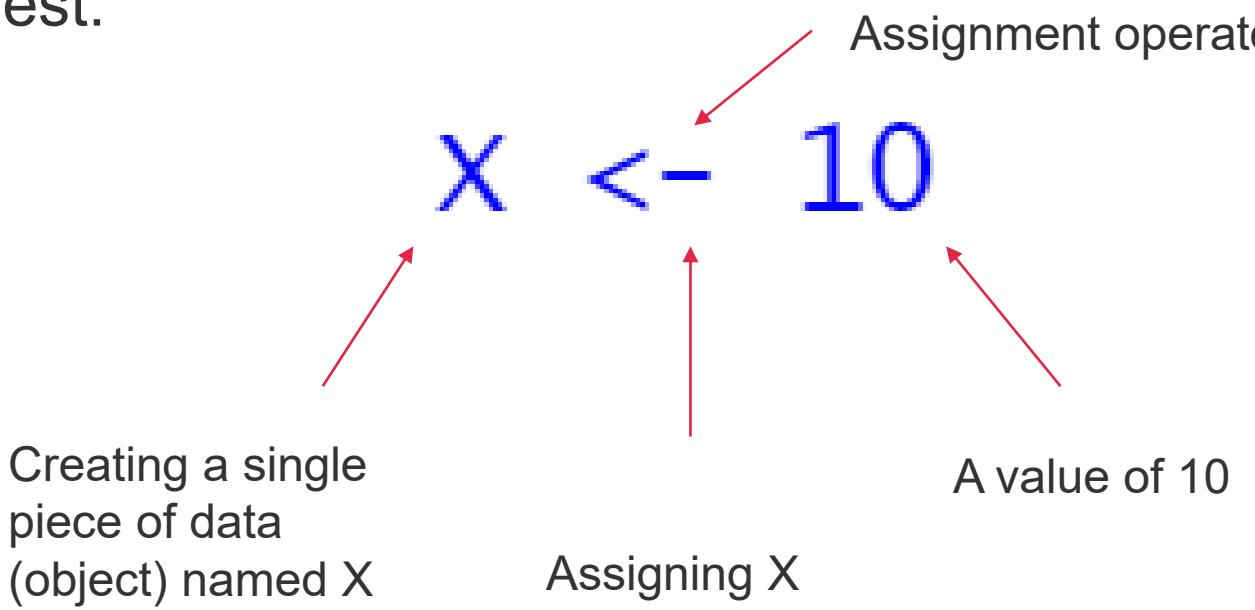
TIDYVERSE EXTRA PACKAGES:

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: INTERACTING WITH OBJECTS

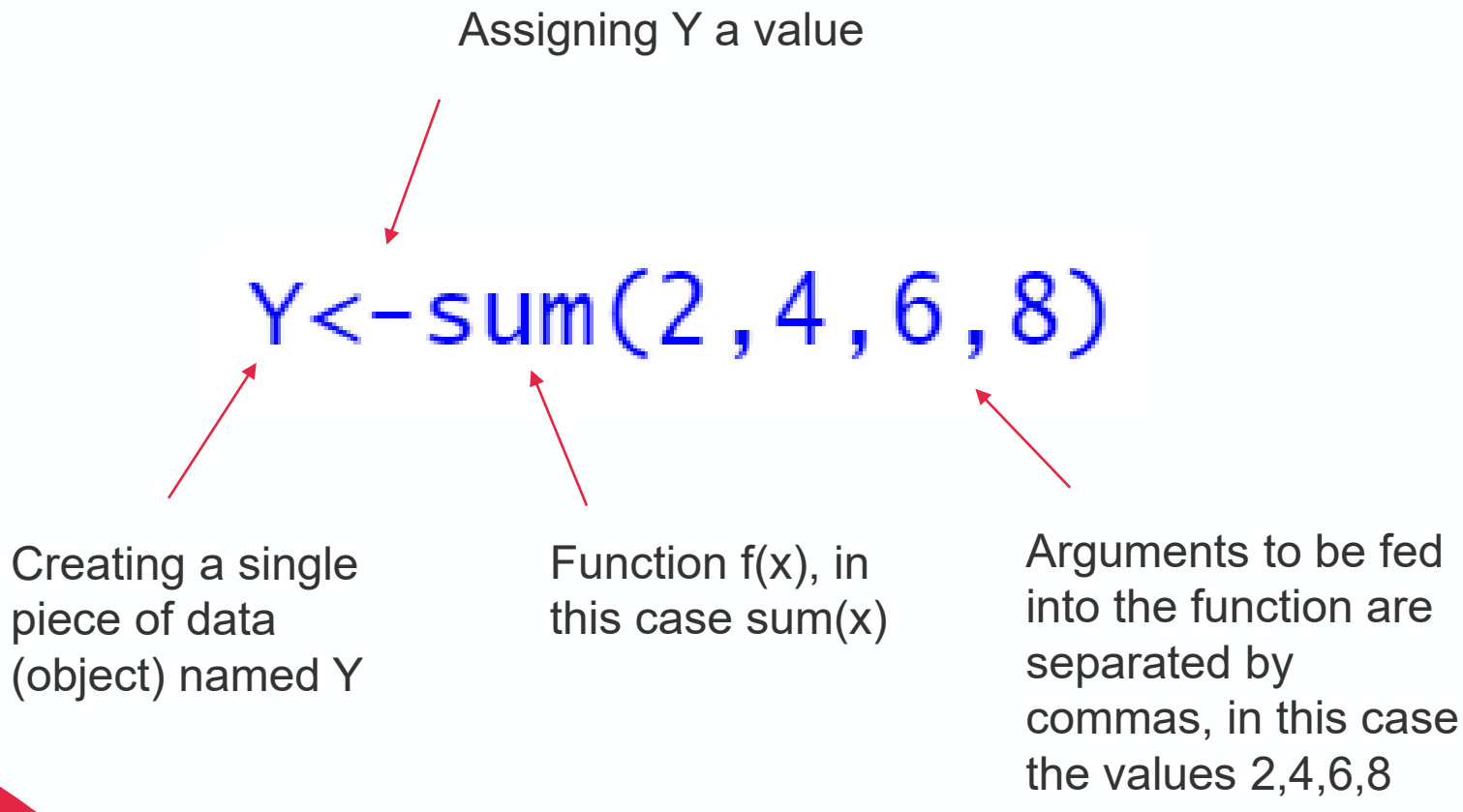
- We use an **assignment operator** (depicted as an arrow) to assign a "value" to a "name"

- At it's simplest:



R SYNTAX: INTERACTING WITH OBJECTS

- Basic function structure:



R SYNTAX: INTERACTING WITH OBJECTS

- Vectors are the simplest data structure
  - Storage of just a single value
  - Storage of multiple values
  - Storage of any data type
    - character (text)
    - numeric and integer (real numbers)
    - logical (TRUE or FALSE)
    - complex (imaginary numbers)

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

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

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

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

R SYNTAX: INTERACTING WITH OBJECTS

- 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"
```

R SYNTAX: INTERACTING WITH OBJECTS

- 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

```
"30jul1960" = "%d%b%Y" =  "1960-07-30"
```

R SYNTAX: INTERACTING WITH OBJECTS

- 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

R SYNTAX: INTERACTING WITH OBJECTS

- 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
```

R SYNTAX: INTERACTING WITH OBJECTS

- 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
```

R SYNTAX: INTERACTING WITH OBJECTS

- Lists are the most flexible data structure and thus often the most complex to work with

- Can have a list of individual values (i.e., vector), a list of several vectors, matrices, data frames, or 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
```

R SYNTAX: INTERACTING WITH OBJECTS

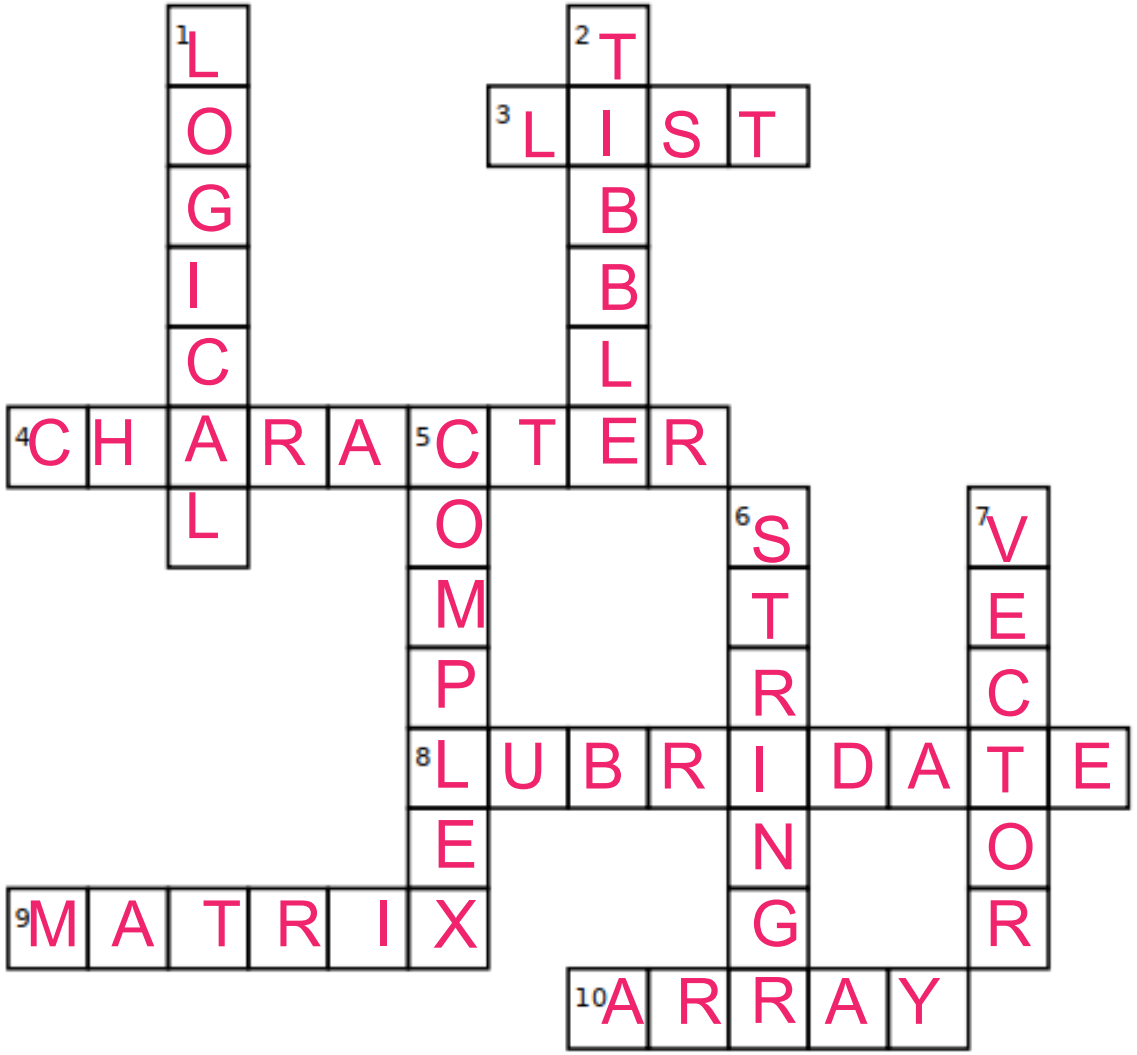
- Factors are data structures which are used to group variables into categories or levels (ordered)

R SYNTAX: INTERACTING WITH OBJECTS

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

LET’S REVIEW

Crossword puzzle



Down:

1 – A data type that handles data with only two possible values

2 – A core tidyverse package designed for working with dataframes

5 – A data type that handles imaginary numbers

6 – A core tidyverse package designed for working with character/text data

7 – The simplest data structure

Across:

3 – The most flexible data structure

4 – A data class for handling text

8 – A tidyverse package for working with dates

9 – A data structure for two dimensional data of a single type (e.g., numeric variables only)

10 – A data structure that can handle three dimensional data

BREAK TRIVIA - FAMOUS SHIPWRECKS

Description	Options
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 man 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 storm in 1942.	D – Terror

Share your answers in the chat box!

STORAGE, OBJECTS, AND SYNTAX

Oh my!

SYNTAX AND INDEXING

```
breakfast_foods <- c('hot cereal', 'cold cereal', 'eggs', 'fruit', 'toast', 'smoothie')
```

- What does the following function do?

```
X <- length(breakfast_foods)
```

- What value will the following command return?

```
breakfast_foods[5]
```

R INDEXING

- R uses the square bracket convention (e.g. x[*i*] will return the i^{th} item in x) to access data stored within a variable
- For vectors this is simple:

```
> cupboard <- c('tea', 'nutmeg', 'oatmeal')
> cupboard[3]
[1] "oatmeal"
```

INDEXING AND SUBSETTING:

- For **matrices** and **data.frames**, you must indicate a **row** and **column** that you wish to return, with [1,1] referring to the **top left** of your data:

my.mat =

4	7	8
2	5	6
1	3	9

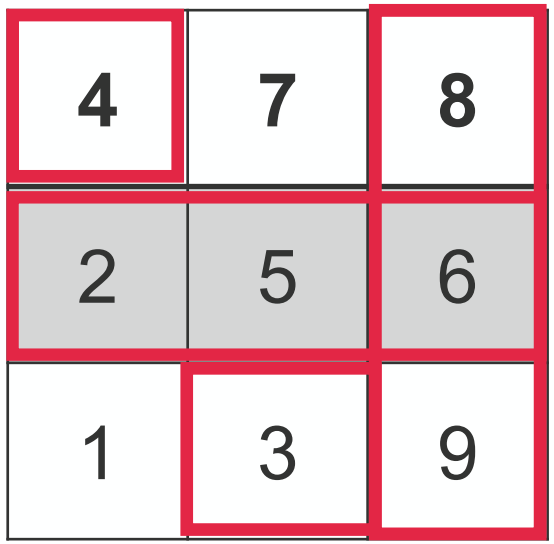
row *column*

my.mat[1, 1] = 4
my.mat[3, 2] = ?
my.mat[2,] = 2 5 6
my.mat[, 3] = ?

INDEXING AND SUBSETTING:

- For **matrices** and **data.frames**, you must indicate a **row** and **column** that you wish to return, with [1,1] referring to the **top left** of your data:

my.mat =



row *column*

my.mat[1, 1] = 4
my.mat[3, 2] = 3
my.mat[2,] = 2 5 6
my.mat[, 3] = 8 6 9

INDEXING AND SUBSETTING:

- Additionally, for **data.frames**, you can access the columns by their column names using the `$` operator:

my.df =

col1	col2	col3
4	**7**	**8**
2	5	6
1	3	9

my.df**$col1** = **4 2 1**

my.df**$col2** = **?**

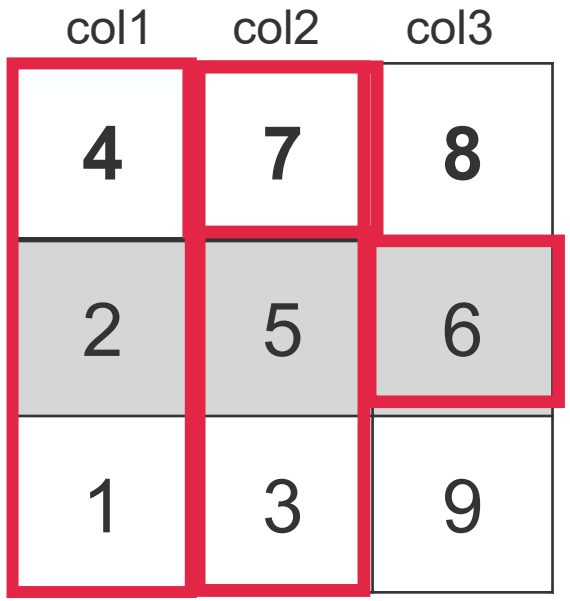
my.df**$col3[2]** = **6**

my.df**$col2[1]** = **?**

INDEXING AND SUBSETTING:

- Additionally, for **data.frames**, you can access the columns by their column names using the `$` operator:

my.df =



my.df**$col1** = **4 2 1**

my.df**$col2** = **7 5 3**

my.df**$col3[2]** = **6**

my.df**$col2[1]** = **7**

R SYNTAX – BASE R VS. TIDYVERSE

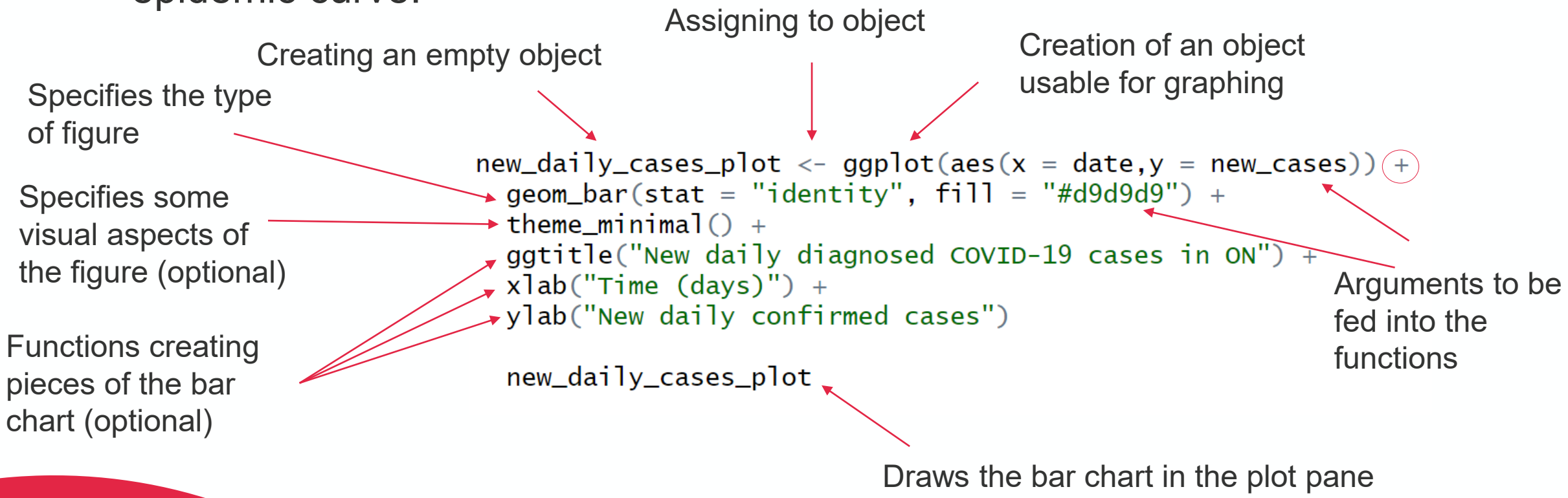
- Why know both? How are they different?

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)
```

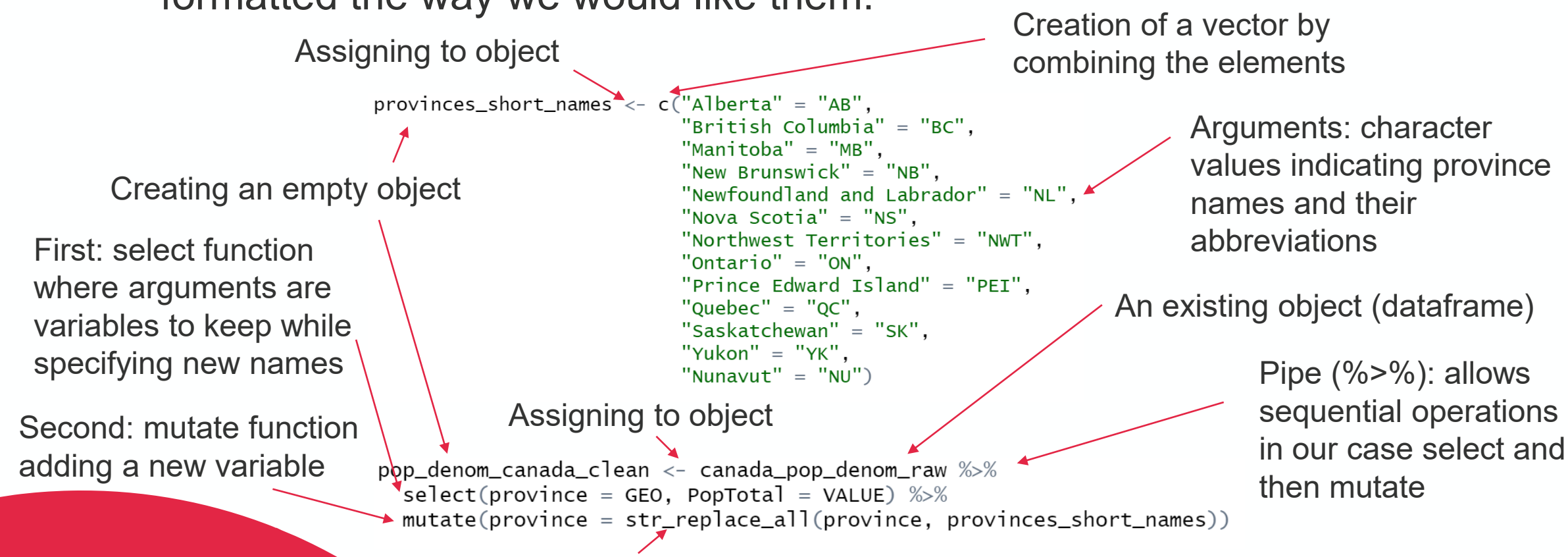
R SYNTAX - TIDYVERSE

- Example of a set of functions working together to create a bar chart or epidemic curve:



R SYNTAX - TIDYVERSE

- Example of functions working together to create an object with data formatted the way we would like them:



CONSIDERATIONS FOR PROGRAMMING

Save yourself from yourself!

PROGRAMMING TIPS

Do's	Do Not's
Create a header that includes: • Date the code was written and date it was last modified • The author and person who last modified the code • Purpose of the code • Associated datasets and location • Pertinent notes (e.g., decisions)	Leave code and syntax documents in such a way that it is unclear what they are for, when they were created, or who created them (this is especially important for tasks which are run regularly but not frequently)
If using one script document, chunk the code by various stages of the project (e.g., loading data, processing data, analysing data, summarising data, visualising data, reporting and data export)	Leave code in such a state that it is unclear when to run which code block
Write brief comments throughout explaining what code blocks or complicated code is doing	Leave code in such a state that is unclear what they are meant to do
Revise code such that it can be run all at once, especially if automating tasks	Leave code in your script that does not work or that is not needed

AUTOMATION: WHEN IS IT APPROPRIATE?

Appropriate:

- Conducting an analysis that will be repeated regularly
- Conducting an analysis that will be revisited at a later date
- Other?

Not appropriate

- Conducting an analysis which will be run once only
- Handing over a “one-off” data analysis to another individual
- Other?

GETTING THE MOST FROM GOOGLE

- It is an extraordinary human that can sit and write R code like they are writing a letter

- Google is your friend!
  - As are stackoverflow.com, tidyverse.org, rdocumentation.org, cookbook-r.com, etc.

- Tip in framing web search: software + version + key terms

If you can’t tie knots…

Photo by Miguel A. Amutio on Unsplash

Photo by Pascal van de Vendel on Unsplash

…tie lots!

DEMO: FUNCTIONS TO EXPLORE YOUR DATA

Base R Cheat Sheet

Getting Help

Accessing the help files

```
?mean
```

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5	
seq(2, 3, by=0.5)	2.0 2.5	

for (variable

Data Import : : CHEAT SHEET

R's **tidyverse** is built around **tidy data** stored

Read Tabular Data - These functions share the common arguments:

Data types

nctions guess
s of each column and
types when appropriate (but will NOT
strings to factors automatically).

ge shows the type of each column in the

```
sed with column specification:
s(
ge = col_integer(),
ex = col_character(),
arn = col_double()
```

age is an integer

sex is a character

arn is a double (numeric)

se problems.
roblems(x)

e parsing.

double()
') Also
l_time(format = "")
d = FALSE)

Data Transformation with dplyr : : CHEAT SHEET

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1**)** Extract column values as a vector. Choose by name or index.

Dates and times with lubridate : : CHEAT SHEET

Date-times

2017-11-28 12:00:00

A **date-time** is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC

dt <- ***as_datetime****(1511870400)*
"2017-11-28 12:00:00 UTC"

2017-11-28

A **date** is a day stored as the number of days since 1970-01-01

d <- ***as_date****(17498)*
"2017-11-28"

Round Date-times

String manipulation with stringr : : CHEAT SHEET

The **stringr** package provides a set of internally consistent tools for working with character strings, i.e. sequences of characters surrounded by quotation marks.

1L, end = -1L) Extract
racter vector.
.sub(fruit, -2)

ttern) Return only the
pattern match.

ttern) Return the first
n each string, as a vector.
o return every pattern
it, "[aeiou]")

tern) Return the first
n each string, as a
for each ( ) group in
ch_all.
"(a|the) ([^]+)")

Manage Lengths

str_length(string) The width of strings (i.e. number of code points, which generally equals the number of characters). *str_length(fruit)*

str_pad(string, width, side = c("left", "right", "both"), pad = " ") Pad strings to constant width. *str_pad(fruit, 17)*

str_trunc(string, width, side = c("right", "left", "center"), ellipsis = "...") Truncate the width of strings, replacing content with ellipsis. *str_trunc(fruit, 3)*

str_trim(string, side = c("both", "left", "right")) Trim whitespace from the start and/or end of a string. *str_trim(fruit)*

Order Strings

str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Return the vector of indexes that sorts a character vector. *x[str_order(x)]*

str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1] Sort a

llapse = NULL) Join
o a single string.
RS)

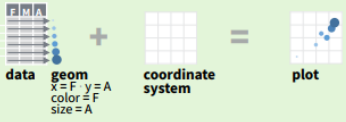
llapse = ""**)** Collapse
nto a single string.
se = "")

Data Visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

required

```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
```

Not required, sensible defaults

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1),curvature=1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

TWO VARIABLES

continuous x , continuous y

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

```
f <- ggplot(mpg, aes(class, hwy))
```

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))
```

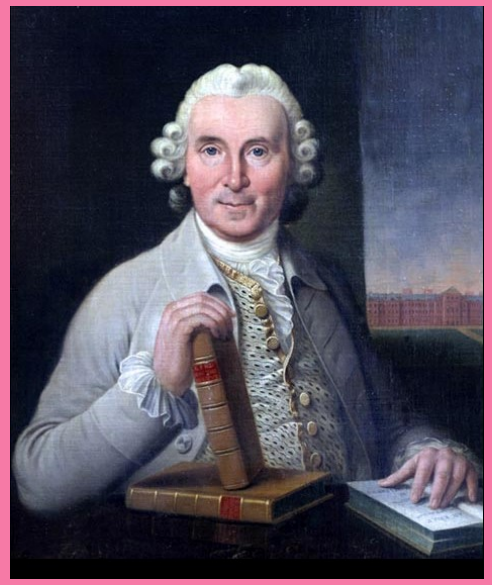
j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also

https://rstudio.com/resources/cheatsheets/

BREAK 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 box!

TDU.

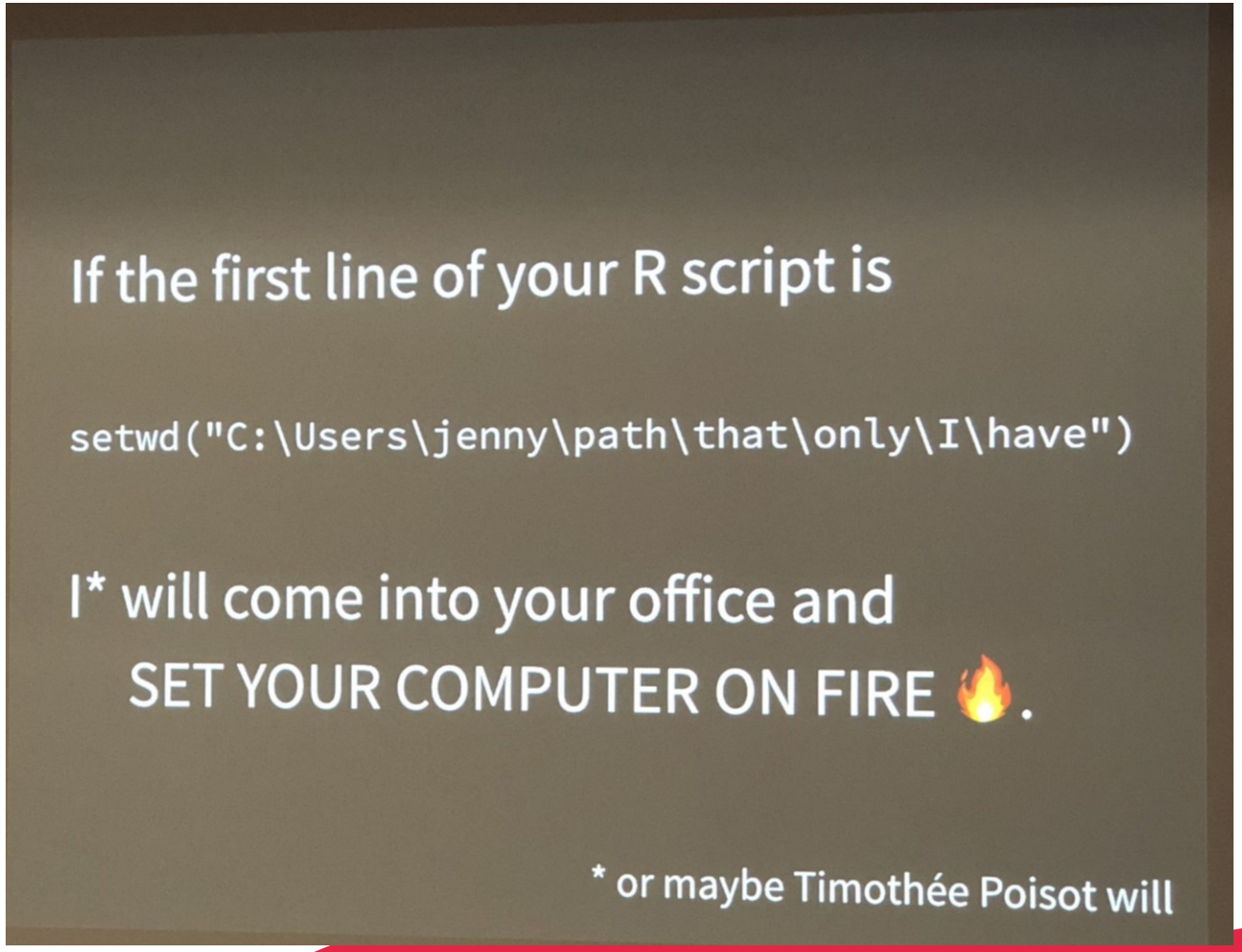
SETTING YOURSELF UP FOR SUCCESS: DEMO

R-PROJECTS && HERE()

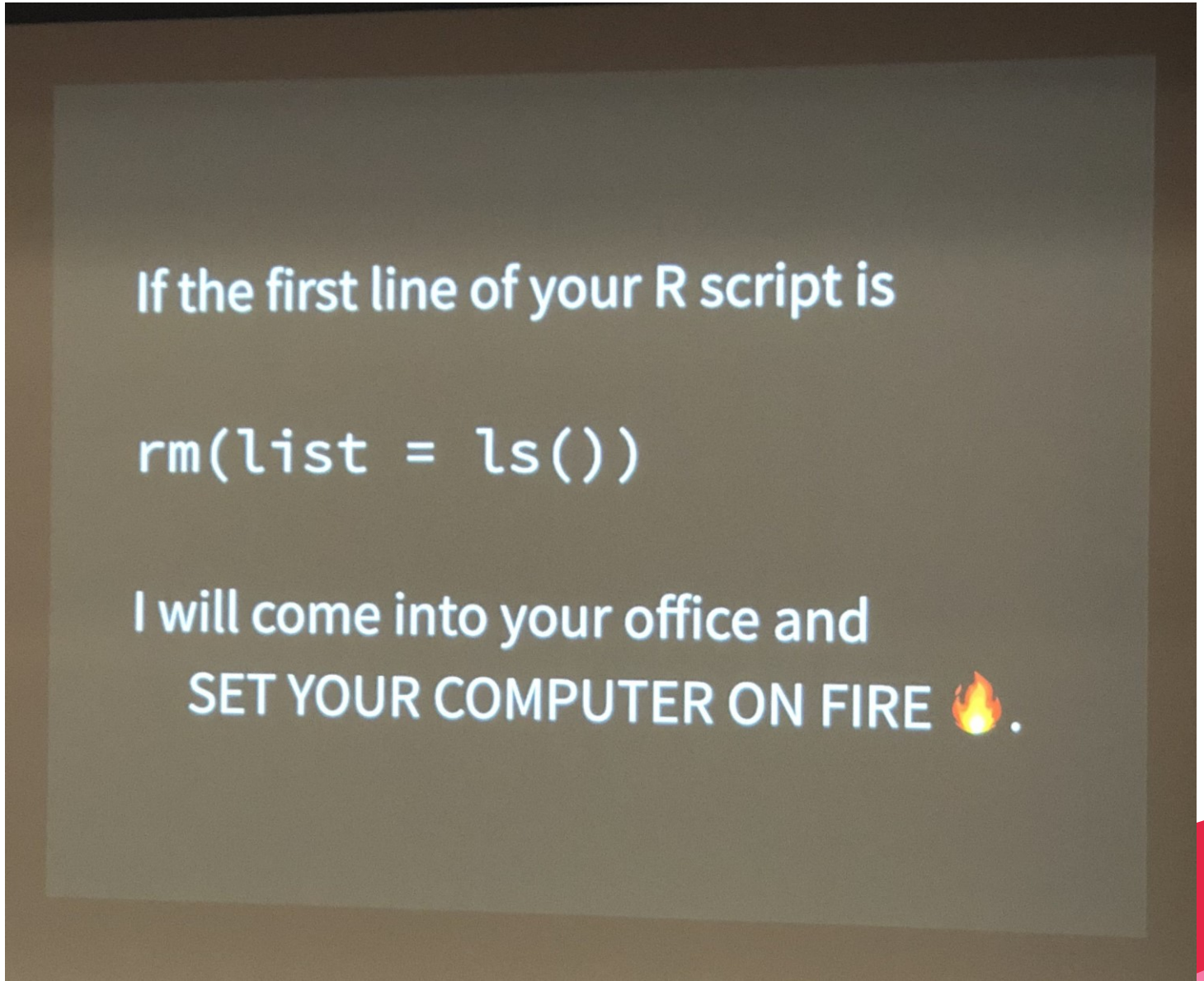
Public Health Agency of Canada
Agence de la santé publique du Canada

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!

WRAP-UP

DAY 1 - EXERCISE

CONCEPTUAL

- CMOH wants a repeat analysis from PHAC open access COVID-19 linelist:
  - Manipulating dates
  - Reshaping data
  - Creating new variables
  - Creating the plot
  - Automating the results by writing a script

TECHNICAL

- Preparations to start your work in R
  - Load installed packages
  - Import data
  - Open scripts
  - Workspace setup
- Cleaning data
- Creating new variable
- Summarising data
- Plotting data (epidemic curve)
- Automating the analysis

PRE-LEARNING FOR DAY 2

- Details can be found in the Participant Guide:

1. Handout: TB 101
2. Webpage: Tidygraph
3. Webpage: Flextable
4. From pre-course self-study module:
   - Video: Data Manipulation Tools and Dplyr (20 mins)
   - Video: R Markdown (7 minutes)

REMINDER

- Download course materials for Day 2

- Demo on:
  - Here() package and setting up your R working environments
  - Getting started on exercise 1

Questions?

NEED HELP OR CLARIFICATION?

Remember: Slack channel for this training session

- Phacrusers.slack.com -> intro_r_oct2023cohort

- **Use the Slack channel to:**
  - Connect with your peers
  - Find materials shared throughout the course
  - Ask questions about the materials/exercises to your peers, facilitators and subject matter experts
  - Help others who have questions if you know the answer!

FEEDBACK FROM DAY 1

- Please answer a few questions to provide us some feedback!

https://www.slido.com/

#ROct2023

EXERCISE DEMO

Getting Started