



THE UNIVERSITY  
*of* EDINBURGH | **U**usher institute

# Tutorial 2

Data Types and Structures in Python and R

Supported by



THE UNIVERSITY  
*of* EDINBURGH

**DDI** Data-Driven  
Innovation

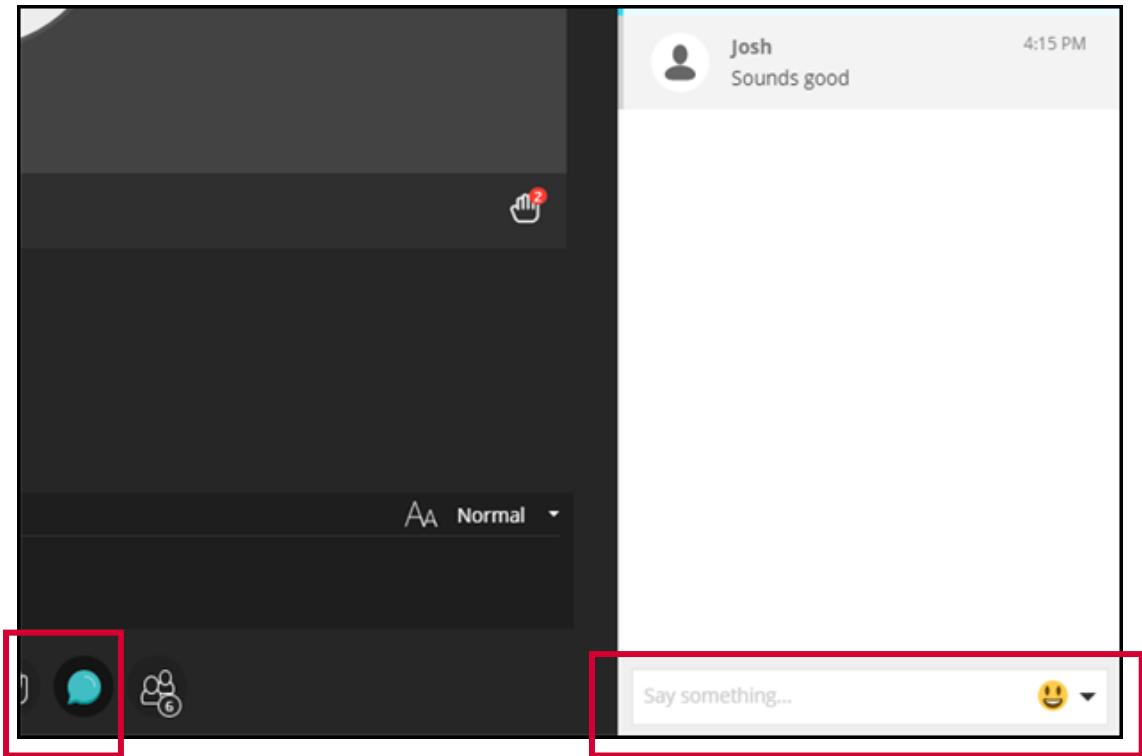
# Audio check

Can you hear the presenter talking?

Please type **yes** or **no** in the “Text chat area”

If you can't hear:

- Check your Audio/Visual settings in the Collaborate Panel
- Try signing out and signing back into the session
- Type into the chat box and a moderator will try to assist you



# Recording

This session will now be recorded. Any further information that you provide during a session is optional and in doing so you give us consent to process this information.

These sessions will be stored by the University of Edinburgh for one year and published for 30 days after the event. Schools or Services may use the recordings for up to a year on relevant websites.

By taking part in a session, you give us your consent to process any information you provide during it.

 Start Recording





THE UNIVERSITY  
*of* EDINBURGH | **U**usher institute

# Tutorial 2

Data Types and Structures in Python and R

Supported by



THE UNIVERSITY  
*of* EDINBURGH

**DDI** Data-Driven  
Innovation

How are you feeling today on this range of expressive animals?



# Agenda

- !? Error of the week
  -  Understanding and navigating folder structures in code
  -  Errors! Oh my ... how to read them
-  Data types overview
-  Data types activity
- ? Any questions



# !? Error of the week !?

- 📁 Understanding and navigating folder structures in code
- 📖 How to read error messages



# How to find your current path

## Python

- OS module to work with paths independent of operating systems
- import os  
os.getcwd()
  - to get the current working directory

## R

- getwd()
  - base R function to get the current working directory
- When using R Projects, your current working directory automatically points to the root folder where that .Rproj file is saved
- When using an RMD, it is where the RMD file is located



# Navigating folder structures in code

**File path** = location of a file on a computer's file system structure

**Absolute path** = specified from the root directory (which is the first or topmost directory)

- AKA “full file paths”
- ~ commonly used to represent user’s home directory
- e.g., C:\Users\bblankin\Teaching\AY2024–25\Data Types and Structures in Python and R\Tutorial Slides or ~\Teaching\AY2024–25\Data Types and Structures in Python and R\Tutorial Slides

**Relative path** = path relative to the current directory

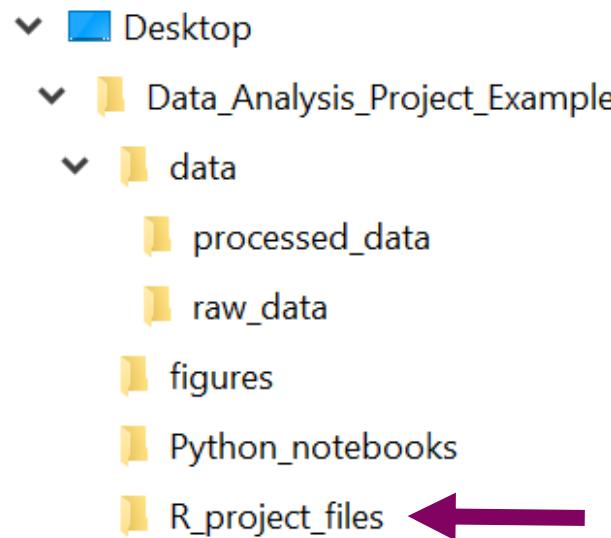
- Reproducible if you share your code with someone who has the same file and folder set up!
- Single dot (.) indicates current directory & double dot (..) represents parent directory
- e.g., .\Data Types and Structures in Python and R\Tutorial Slides

Path separators formatting note: Mac & Linux use / whereas Windows uses \

- URLs follow a standard format always using forward slash / regardless of operating system



# Navigating folder structures in code



In Noteable, your home directory  
(\home\joyyan\ or ~\ ) = Jupyter Notebook  
File Browser (what you see when you open a  
Standard Python 3 server)

The absolute path to a RMD file in “R\_project\_files” is location:

~\Desktop\Data\_Analysis\_Project\_Example\R\_project\_files\R\_file.RMD

To navigate **up** the folder tree, use the “..” prefix.

To navigate 2 levels up, repeat the up prefix twice “..\..\”

For example to go up to figures from R\_project\_files it would be:

..\figures\figure\_1.png

To get to raw\_data from within R\_project\_files:

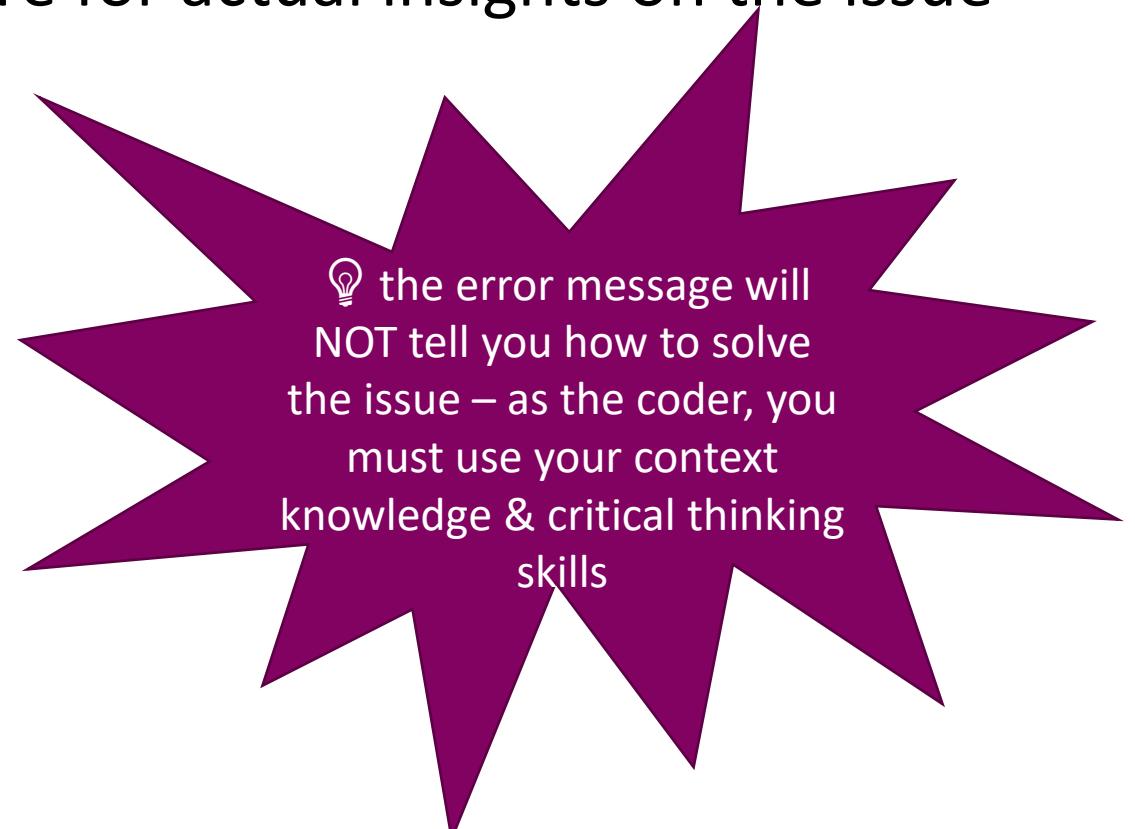
..\data\raw\_data\raw\_data.csv

# How to read errors

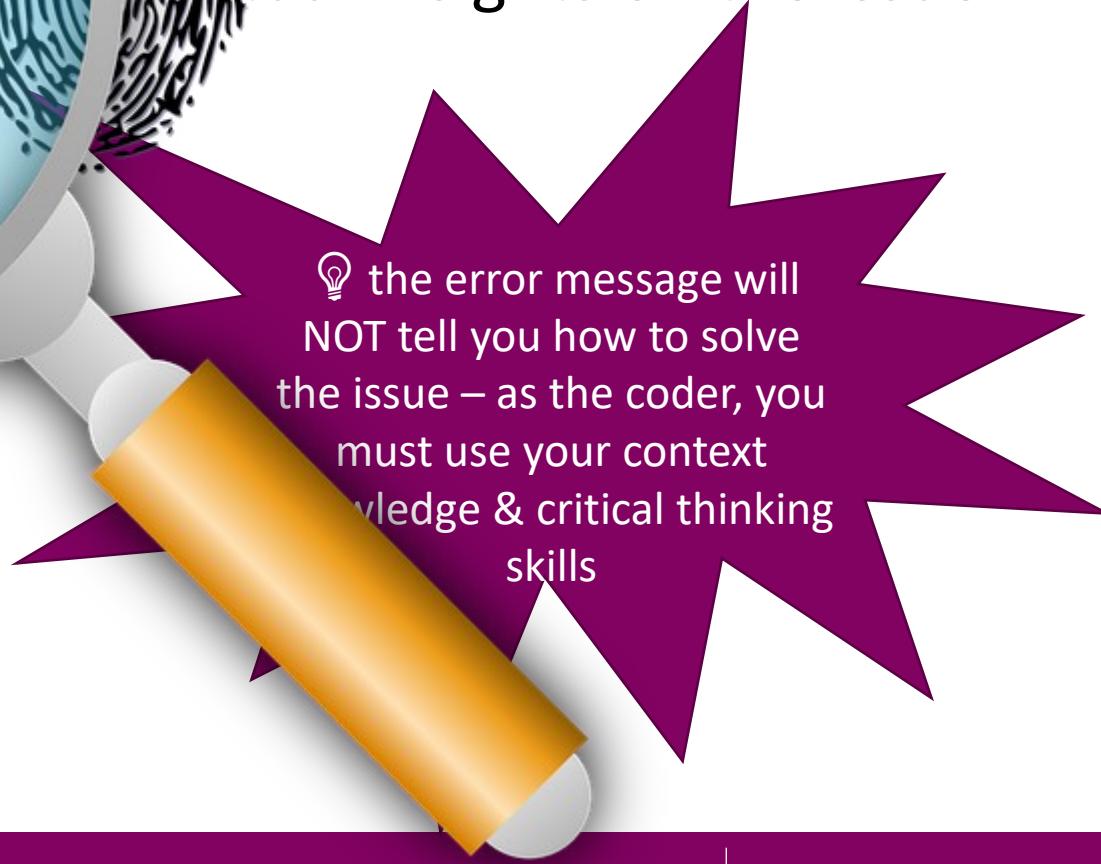
- Step 1: Take a deep breath
- Step 2: Go to the bottom and read there for actual insights on the issue

↑ Read from bottom to top!

- Key components:
  - Error location (if applicable)
  - Error type (more likely in Python)
  - Error message



# How to read errors

- Step 1: Take a deep breath
  - Step 2: Go to the bottom and read there for actual insights on the issue
-  **↑ Read from bottom to top!**
- Key components:
    - Error location (if applicable)
    - Error type (more likely in Python)
    - Error message
-  **💡** the error message will NOT tell you how to solve the issue – as the coder, you must use your context knowledge & critical thinking skills

# Some examples – Python

```
[1]: X
```

-----

NameError Traceback (most recent call last)  
Cell In[1], line 1  
----> 1 X  
NameError: name 'X' is not defined

Error type

Error message

Error location

# Some examples – Python

Error type

Error message

Error location

```
[5]: from IPython.display import Image
      Image(filename = "figures/PythoLogo.png")

-----
FileNotFoundError                                     Traceback (most recent call last)
Cell In[5], line 2
      1 from IPython.display import Image
----> 2 Image(filename = "figures/PythoLogo.png")

File /opt/conda/lib/python3.11/site-packages/IPython/core/display.py:970, in Image.__init__(self, data, url, filename, format, embed, width, height, retina, unconfined, metadata, alt)
    968 self.unconfined = unconfined
    969 self.alt = alt
--> 970 super(Image, self).__init__(data=data, url=url, filename=filename,
    971                         metadata=metadata)
    973 if self.width is None and self.metadata.get('width', {}):
    974     self.width = metadata['width']

File /opt/conda/lib/python3.11/site-packages/IPython/core/display.py:327, in DisplayObject.__init__(self, data, url, filename, metadata)
  324 elif self.metadata is None:
  325     self.metadata = {}
--> 327 self.reload()
  328 self._check_data()

File /opt/conda/lib/python3.11/site-packages/IPython/core/display.py:1005, in Image.reload(self)
  1003 """Reload the raw data from file or URL."""
  1004 if self.embed:
-> 1005     super(Image, self).reload()
  1006     if self.retina:
  1007         self._retina_shape()

File /opt/conda/lib/python3.11/site-packages/IPython/core/display.py:353, in DisplayObject.reload(self)
  351 if self.filename is not None:
  352     encoding = None if "b" in self._read_flags else "utf-8"
--> 353     with open(self.filename, self._read_flags, encoding=encoding) as f:
  354         self.data = f.read()
  355 elif self.url is not None:
  356     # Deferred import

FileNotFoundError: [Errno 2] No such file or directory: 'figures/PythoLogo.png'
```



# Some examples – R (in RMD/console)

A screenshot of an RStudio interface. In the code editor pane, lines 29, 30, and 31 are visible. Line 30 contains the code `~ ~ {r}`. Line 31 has a cursor at the beginning of the letter 'X' in the word 'X'. Line 32 shows the continuation of the code block. A red rectangular box highlights the character 'X' on line 31. In the bottom right corner of the RStudio window, there is an error message box with a purple border containing the text "Error: object 'X' not found".

Error type

Error message

Error location

# Some examples – R (knitting)

Error type

Error message

Error location

The screenshot shows the RStudio interface with two main panes: a Source editor and a Console pane.

**Source Editor:** The file is named "Tutorial01-R.Rmd". It contains R Markdown code. A warning message at the top states: "Package gapminder required but is not installed. [Install](#) [Don't S](#)". The code includes a YAML front matter block and several R code chunks. One chunk uses `knitr::opts\_chunk\$set(echo = TRUE)`.

**Console Pane:** The output shows the R environment and the results of the knitting process. It starts with "processing file: Tutorial01-R.Rmd". An error message is highlighted in a pink box: "Error in `library()` : ! there is no package called 'gapminder'". Below the error is a "Backtrace" showing the call stack from line 1 to line 19. The trace ends with "evaluate (local) fun(base::quote(`<package>`))". At the bottom, it says "Quitting from lines 63-69 [load-packages] (Tutorial01-R.Rmd)" and "Execution halted".

# Some examples – R (knitting) ... what lines 63-69 look like when code is run in the RMD document

Error type

Error message

Error location

The screenshot shows an RStudio interface with an RMD file open. The code editor displays the following lines:

```
61 62  ``{r load-packages}
63 Sys.setenv(TZ="Europe/London") # set timezone (TZ argument) to whatever timezone you
64 may be in, I am in the UK so am using the TZ identifier for London. This allows the to
65 packages to work as intended
66 # for a list of TZ identifiers see here:
67 https://en.wikipedia.org/wiki/List\_of\_tz\_database\_time\_zones
68 library(tidyverse)
69 library(gapminder) # for our data
70 ````
```

An orange box highlights line 69, which is the source of the error. A green box highlights the error message:

Error in library(gapminder) : there is no package called 'gapminder'

The error message is also displayed in the console area:

```
2. stop(packageNotFoundError(package, lib.loc, sys.call()))
1. library(gapminder)
```

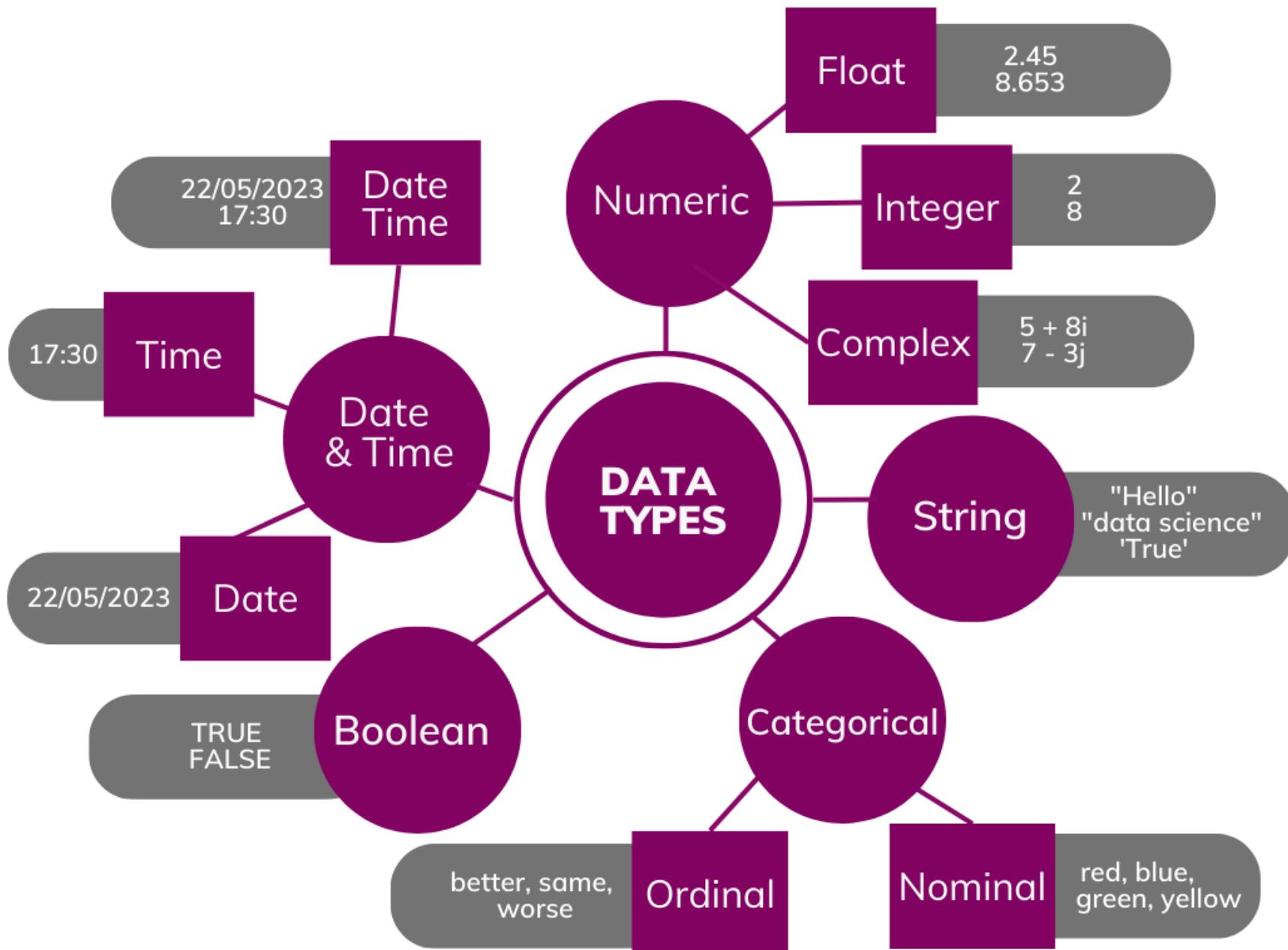
# What are data types



- Type of data or values an object contains
  - Internal construct that a programming language uses to understand how to store and manipulate data
- Determines:
  - What kind of mathematical, relational, or logical operation can be applied
  - Which operations can be performed to create, transform, and use the variable in further computation

## DATA TYPES





# Data types produced in data generating scenarios

- Emergency service call outs and transfers (e.g., ambulance)
- Hospital waiting times for non-elective surgeries
- Smoking behaviours in the community
- At-home carers in a local council area (local region)
- Any other health and social care situation you can think of!



# Data types produced in data generating scenarios

-  35 min: Discuss the data generated in your selected scenario
  - What types of data they are
  - What are the possible range of values
  - Could the data type vary depending on your analytic use case? If so, how?
- Share with everyone what you spoke about in your groups
- Be sure to save your document so you can post it on the discussion boards after!



