**## INTRODUCTION**

Workshop leaders + Ben if present + helpers

Research Platforms spiel?

**## MOTIVATION**

Why R/RStudio? Statistics, data visualisation, and reproducibility

Show some examples of end of analysis outputs. Leaflet map. Shiny app?

**## INTRODUCE COURSE STRUCTURE**

Project directory structures

Code script structures

Basic R

Load > manipulate data

Data visualisation

**## R VS RSTUDIO**

R is a free and open source programming language for statistical computing

RStudio is a free and open source integrated development environment (IDE) for R

Bottom left: This is the Console.

Top left: This is the Script Editor

Top Right: This is the Environment.

Bottom Right: This is the Files/Plots/Packages/Help/Viewer pane

Tools > Global Options

Code > Editing > Ctrl+Enter

Code > Display > Highlight function calls

Code > Completion > Tab for multiline completion

Code > Diagnostics > Check Args/Warn/Warn

Appearance

Pane Layout

**## PROJECT SET UP**

Before jumping into R, establish some programming best practices (you’ll thank us later)

“one folder, one project” mentality.

No “best” structure

Example:

Data folder, potentially split into raw and clean

Doc or reports for any manuscripts

Figs/images for graphs/figures

Outputs for outputs. Could include graphs figures, but also csvs, model objects etc

Scripts for scripts

**###################**

**### Challenge 1 ###**

**###################**

**# Build your own project directory structure for this workshop. For our purposes today you will require at least a `data`, `scripts`, and `images` folder**

R Projects

Relative file paths

Maintain your environment between sessions

Easy collaboration

Version control

**###################**

**### Challenge 2 ###**

**###################**

**# Add an `R Project` to the existing directory you created in Challenge 1.**

Code script structures

Again no set method, some examples:

A title and “script abstract”

Use headings/subheadings

Have any loaded packages called at the beginning of the script

Comment your code!

Code Style Guides

a series of self-imposed grammer/syntax rules to follow to write neat and human-readable code

Google vs Hadley

Naming conventions

<- instead of =

A line length of <80 characters

White space

Multi-line function calls

**## BASIC R**

Calculator

Order of operations

Parenthesis > exponents > Divide > multiply > addition > subtraction

**3 + 5 \* 2**

Comments

Scientific notation: **2/10000**

**5e3**

Mathematical functions: **log(1)**

Function > Parenthesis > exponents > Divide > multiply > addition > subtraction

**10 \* sum(1,2)**

**## Comparisons**

**1 == 1**

**1 != 2**

**1 < 2**

**1 <= 1**

**1 > 0**

**1 >= -9**

**## Variable assignment**

**x <- 1**

**x**

Environment tab

**Log(x)**

**x <- 100**

**x <- x + 1**

**## Vectorisation**

R is vectorised, meaning that variables and functions can have vectors as values

**1:5**

**2^(1:5)**

**x <- 1:5**

**2^x**

**###################**

**### Challenge 3 ###**

**###################**

**# Which of the following are valid variable names?**

**# min\_height**

**# max.height**

**# \_age**

**# .mass**

**# MaxLength**

**# min-length**

**# 2widths**

**# celsius2kelvin**

**###################**

**### Challenge 4 ###**

**###################**

**# What will be the value of each variable after each statement in the following program?**

**# mass <- 47.5**

**# age <- 122**

**# mass <- mass \* 2.3**

**# age <- age – 20**

**###################**

**### Challenge 5 ###**

**###################**

**# Run the code from the previous challenge, and write a command to compare mass to age. Is mass larger than age?**

**## Managing your environment**

**ls()**

**rm()**

**rm(list = ls())**

**rm(list <- ls())**

Errors vs warnings

**## Packages**

**install.packages("packagename")**

**library(packagename)**

**###################**

**### Challenge 6 ###**

**###################**

**# Clean up your working environment by deleting the mass and age variables.**

**###################**

**### Challenge 7 ###**

**###################**

**# Install the following package: `ggplot2`**

**## Seeking Help**

**?function\_name**

**help(function\_name)**

**function\_name + F1**

Run through an example

**Special operators: ?"+"**

**??plot fuzzy search**

Cheatsheets

**###################**

**### Challenge 8 ###**

**###################**

**# Look at the help for the `paste` function. You'll need to use this later. What is the difference between the `sep` and `collapse` arguments?**

**## Data Types**

Doubles/Numerics/Floats

Integers

Complex

Logical

Characters

**Typeof() vs class()**

**class(3.14)**

**class(1L)**

**class(1+1i)**

**class(TRUE)**

**class('banana')**

Type coercion:

logical > integer > numeric > complex > character

General programming definition of a vector: an ordered list of things

Vectors in R are **atomic vectors** in that they have the special constraint that every element must be the same data type

**c()**

**my\_vec <- c(2,6,3)**

Given what we’ve learned so far, what do you think the following will produce?

**vec1 <- c(2,6,'3')**

**vec2 <- c("apple", 2.1, TRUE)**

**vec3 <- c(2, 2.0, 2L)**

**as.numeric(c('0','2','4'))**

**as.logical(1)**

**as.logical(0)**

**as.logical(-0.5)**

**as.logical("house")**

**## Data Structures: Vectors**

**ab\_vector <- c('a', 'b')**

**concat\_example <- c(ab\_vector, 'SWC')**

**1:10**

**1.1:9.9**

**seq(from = 1,**

**to = 10,**

**by=0.1)**

Subsetting by index

**my\_vec <- c(1,3,5,6,10)**

**my\_vec[3]**

**my\_vec[c(2,4)]**

Subsetting by criteria

**my\_vec <- 1:10**

**my\_vec[my\_vec > 8]**

**my\_vec[my\_vec %% 2 == 0]**

**my\_vec <- seq(0, 100, 0.1)**

**length(my\_vec)**

**head(my\_vec)**

**tail(my\_vec)**

**name\_vec <- 5:9**

**names(name\_vec) <- c("a", "b", "c", "d", "e")**

**name\_vec**

**name\_vec["a"]**

**name\_vec[c("a", "b")]**

**###################**

**### Challenge 9 ###**

**###################**

**# Given the following lines of code:**

**# x <- 1:5**

**# names(x) <- letters[1:5]**

**# x**

**# Find at least five different commands to come up with the following subset:**

**# b c d**

**# 2 3 4**

**# Fictional bonus points for anyone who figures out the %in% operator!**

**## Data Structures: Lists**

While everything in a vector has to be the same data type, a list is a really useful data structure to know since you can fill it with anything (thus a programming vector!).

**list\_example <- list(1, "a", TRUE, 1+4i)**

**another\_list <- list(title = "Research Bazaar", numbers = 1:10, data = TRUE )**

**[] vs [[]]**

**another\_list[1]**

**another\_list[[1]]**

**another\_list$title**

**####################**

**### Challenge 10 ###**

**####################**

**# Using the following code:**

**# challenge\_list <- list(words = c("alpha", "beta", "gamma"),**

**# numbers = 1:10,**

**# letter = letters)**

**# challenge\_list**

**# Extract the following things:**

**# - The word "gamma"**

**# - The letters "a", "e", "i", "o", and "u"**

**# - The numbers less than or equal to 3**

**# More fictional bonus points if you use a different methods!**

**## Data Structures: Data Frames**

Let’s start by making a toy dataset in your data/ directory, called feline.csv. Copy the following lines of data, open a new text file in RStudio with File > New File > Text File, paste the data, and save it to the appropriate directory.

**coat,weight,likes\_string**

**calico,2.1,TRUE**

**black,5.0,FALSE**

**tabby,3.2,TRUE**

We can load this into R via the following:

**cats <- read.csv(file = "data/feline.csv")**

**cats**

Each column in a data frame is a vector (same data type), and each row is a list (different data types).

**Str()**

**cats$weight**

**cats$weight[1]**

**cats$weight[2]**

**cats$weight[1] + cats$weight[2]**

[,] notation

**cats[1, ]**

**cats[ , 2]**

**cats[2, 3]**

Lets try to add a new cat

**garfield <- c("marmalade", 20, FALSE)**

**Garfield # why wont this one work? Type coercion**

**garfield <- list("marmalade", 20, FALSE)**

**garfield**

**cats2 <- rbind(cats, garfield) # why wont this work? Factors!**

**## Factors**

Look like character data but are stored as integers with a look-up table

**str(cats$coat)**

**levels(cats$coat)**

**levels(cats$coat) <- c(levels(cats$coat), "marmalade") # add a new level**

**levels(cats$coat)**

While factors are essential for statistical modelling they can be a nuisance in other instances. R will load all character data as factors by default, but we can tell it not to

**####################**

**### Challenge 11 ###**

**####################**

**# Look thorugh the help file for the read.csv() command to find an argument to stop character data from being loaded as factors. Hint: Characters are sometimes referred to as strings.**

**# Reload the cats data frame from file without factors**

**# Add the new row of Garfield data to the data frame**

**####################**

**### Challenge 12 ###**

**####################**

**# Create a list of length two containing a character vector for each of the sections in this part of the workshop:**

**# - Data types**

**# - Data structures**

**# Populate each character vector with the names of the data types and data structures we've seen so far.**

**## LUNCH BREAK HERE**

**##Data frame Manipuation**

Quick recap of basic [,] notation

We have previously covered some simple subsetting methods using $ and [ , ] for extraction of specific elements, but what if we’re only after elements that meet a set criteria?

**titanic <- read.csv("https://goo.gl/4Gqsnz", header=TRUE)**

**mean(titanic$Age)**

**anyNA(titanic$Age)**

**mean(titanic$Age, na.rm = TRUE)**

**mean(titanic$Survived, na.rm = TRUE)**

But the two values above can’t be fairly used in conjunction with each other. Why is that? Well it turns out that while we don’t know the age of some of the passengers, we do know if each of them survived. This means we are summarising values from different data sets: the survival rate of all passengers, and the mean age of passengers with known ages. Unless the missing values have an identical distribution to the known values their absence will change the mean value. What if all of the unknowns are children? What if they are all >50? This may seem somewhat trivial here, but if you are performing more elaborate statistical analyses like fitting linear models then it is crucial to ensure your datasets are identical.

**titanic\_clean <- na.omit(titanic)**

# Single criteria

**titanic\_cheap <- titanic\_clean[titanic\_clean$Fare < 10, ]**

**dim(titanic\_cheap)**

**####################**

**### Challenge 13 ###**

**####################**

**# Create a new data frame called titanic\_35 that includes only the passengers aged 35 or under.**

# Multiple Criteria

**&**

**|**

**&& / ||**

**titanic\_rich <- titanic\_clean[titanic\_clean$Pclass == 1 & titanic\_clean$Fare > 200, ]**

**dim(titanic\_rich)**

**####################**

**### Challenge 14 ###**

**####################**

**# Create a new data frame that includes only the passengers aged < 18 or >= 50.**

**####################**

**### Challenge 15 ###**

**####################**

**# Create a new data frame that includes only female, third class passengers.**

**####################**

**### Challenge 16 ###**

**####################**

**# Create a new data frame that includes only male passengers aged either 18, 23, 31, or 46.**

**# Hint: Rememberthe %in% operator!**

**## Data Visualisation**

**install.packages("ggplot2")**

**library(ggplot2)**

grammar of graphics concept

idea that any plot can be expressed from the same set of components:

A data set

A coordinate system

A set of geoms - the visual representation of data points

The key to understanding ggplot2 is thinking about a figure in layers

**ggplot(data = titanic\_clean,**

**aes(x = Age,**

**y = Fare))**

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare)) **+**

**geom\_point()**

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare**,**

**col = as.factor(Pclass)**)) +

geom\_point()

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare,

col = as.factor(Pclass))) **+**

**geom\_line()**

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare,

col = as.factor(Pclass))) **+**

**geom\_point() +**

**geom\_line()**

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare,

col = as.factor(Pclass))) +

geom\_point(**col="black"**) +

geom\_line()

**####################**

**### Challenge 17 ###**

**####################**

**# Switch the order of the point and line layers from the previous example. What happened?**

**ggplot(data = titanic\_clean,**

**aes(x = Age,**

**y = Fare,**

**col = as.factor(Pclass))) +**

**geom\_point()**

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare,

col = as.factor(Pclass))) +

geom\_point() **+**

**geom\_smooth(method = "lm")**

ggplot(data = titanic\_clean,

aes(x = Age,

y = Fare,

col = as.factor(Pclass))) +

geom\_point() +

geom\_smooth(method = "lm") **+**

**scale\_y\_log10()**

**####################**

**### Challenge 18 ###**

**####################**

**# Using the following base code:**

**# ggplot(data = titanic\_clean,**

**# aes(x = Age,**

**# y = Fare,**

**# col = as.factor(Pclass))) +**

**# geom\_point() +**

**# geom\_smooth(method = "lm") +**

**# scale\_y\_log10()**

**# Experiment with the additional options you can use to customise the `geom\_point()` and `geom\_smooth()` layer. There is no set challenge, just experiment with different options. Consider changing the point/line size, transparency, linetype, etc.**

**ggplot(data = titanic\_clean,**

**aes(x = Sex,**

**fill = as.factor(Survived))) +**

**geom\_bar(position = "dodge")**

ggplot(data = titanic\_clean,

aes(x = Sex,

fill = as.factor(Survived))) +

geom\_bar(position = "dodge") **+**

**facet\_grid(~ Pclass)**

ggplot(data = titanic\_clean,

aes(x = Sex,

fill = as.factor(Survived))) +

geom\_bar(position = "dodge") +

facet\_grid(~ Pclass) **+**

**theme\_bw()**

ggplot(data = titanic\_clean,

aes(x = Sex,

fill = as.factor(Survived))) +

geom\_bar(position = "dodge") +

facet\_grid(~ Pclass) +

theme\_bw() **+**

**xlab("Passenger age (in years)") +**

**ylab("Number of passengers survived") +**

**ggtitle("Titanic passenger survival rates")**

ggplot(data = titanic\_clean,

aes(x = Sex,

fill = as.factor(Survived))) +

geom\_bar(position = "dodge") +

facet\_grid(~ Pclass**,**

**labeller = labeller(Pclass = c(`1` = "First Class",**

**`2` = "Second Class",**

**`3` = "Third Class")))** +

theme\_bw() +

xlab("Passenger age (in years)") +

ylab("Number of passengers survived") +

ggtitle("Titanic passenger survival rates")

ggplot(data = titanic\_clean,

aes(x = Sex,

fill = as.factor(Survived))) +

geom\_bar(position = "dodge") +

facet\_grid(~ Pclass,

labeller = labeller(Pclass = c(`1` = "First Class",

`2` = "Second Class",

`3` = "Third Class"))) +

theme\_bw() +

xlab("Passenger age (in years)") +

ylab("Number of passengers survived") +

ggtitle("Titanic passenger survival rates") +

**scale\_fill\_discrete(name = "Survival",**

**label = c("Died", "Survived"))**

**ggsave("images/My\_Plot.jpg")**

**ggsave("images/My\_Plot.jpg",**

**height = 10,**

**width = 15,**

**units = "cm")**

**pdf("images/My\_Plot.pdf",**

**width = 10,**

**height = 6)**

**ggplot(data = titanic\_clean,**

**aes(x = Sex,**

**fill = as.factor(Survived))) +**

**geom\_bar(position = "dodge") +**

**facet\_grid(~ Pclass,**

**labeller = labeller(Pclass = c(`1` = "First Class",**

**`2` = "Second Class",**

**`3` = "Third Class"))) +**

**theme\_bw() +**

**xlab("Passenger age (in years)") +**

**ylab("Number of passengers survived") +**

**ggtitle("Titanic passenger survival rates") +**

**scale\_fill\_discrete(name = "Survival",**

**label = c("Died", "Survived"))**

**dev.off()**

**pdf("images/My\_Plot.pdf",**

**width = 10,**

**height = 6)**

**my\_plot**

**dev.off()**

**####################**

**### Challenge 19 ###**

**####################**

**# This is the final challenge of the course and the ONLY assessed component.**

**#**

**# Plot the titanic\_clean dataset in any way you wish. Customise the plot to your heart's content! If you're lacking inspiration try some of these:**

**#**

**# - Change the variables on the axis**

**# - Change line/point widths/clours/transparency**

**# - Change the title and axis labels**

**# - Use a theme!**

**# - Facet the data**

**# - Change the colours!**

**#**

**# If you have ideas and don't know how to do it ask a helper!**

**#**

**# Save the plot to a pdf file in your images/ directory**

**#**

**# Show the plot in the file to a helper to get a tick next to your name on the assessment list. We don't care what it looks like as long as the pdf has a plot in it! Make it as pretty or ugly as you wish!**