



# Introduction to coding *with R and GitHub*

Becky Heath

IPB University

30<sup>th</sup> August 2024



UNIVERSITY OF  
CAMBRIDGE



# Selamat Pagi

Postdoctoral research associate working in the Museum of Zoology at the University of Cambridge in collaboration with IPB university

I work on the RERTA project with industrial and smallholder palm oil plantations to develop better sustainability strategies along waterways

@Becky\_Heath



UNIVERSITY OF  
CAMBRIDGE



rh862@cam.ac.uk

# Overview

- Introduction to coding and R
- Introduction to GitHub
- Plan for the rest of the session
- Where to find help

# Introduction to coding and R

- Introduction to coding and R
- Introduction to GitHub
- Plan for the rest of the session
- Where to find help

# What do we mean by coding?

Writing code is essentially writing a list of instructions for a computer to follow.

Computers operate using 1s and 0s so programming languages are used as a bridge between the user and the system.

Algorithms/scripts are just chunks of code that have been written to tell a computer how to accomplish certain tasks.



# Uses for coding in data science?

Clean, extract, or organise  
data



Automate tasks and do  
things more quickly

Create models and perform  
statistical analysis



Create reproducible  
analysis pipelines

Create professional data  
visualisations



Machine learning, big data,  
predictive modelling

# What about R?

R is the most commonly used programming language for data analysis in ecology and biosciences.

R is specifically designed for statistical analysis, so it's particularly powerful with data visualisation, data manipulation, modelling and visualisation

It's completely open source (free!)



## R vs. R studio?



Language

@Becky\_Heath



A software platform for  
writing and using R  
scripts

rh862@cam.ac.uk



# The R studio interface

The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and running code. The main editor area on the left shows a script titled 'Untitled1' with a single line of code: `print("hello world")`. The bottom-left pane is the Console, which shows the R version (4.2.2), copyright information, and the output of the `print("hello world")` command: `[1] "hello world"`. The bottom-right pane is the Environment pane, which is currently empty. The rightmost pane is the Packages pane, showing a list of installed and available packages.

This is the console, you can type instructions directly here.

Try:  
`print("hello world")`

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-5
askpass	Safe Password Entry for R, Git, and SSH	1.1
av	Working with Audio and Video in R	0.8.3
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.4.1
base64enc	Tools for base64 encoding	0.1-3
BH	Boost C++ Header Files	1.81.0-1
bit	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.5
bit64	A S3 Class for Vectors of 64bit Integers	4.0.5
blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBs')	1.2.3
broom	Convert Statistical Objects into Tidy Tibbles	1.0.3
bslib	Custom 'Bootstrap' Sass Themes for 'shiny' and 'rmarkdown'	0.4.2
cachem	Cache R Objects with Automatic Pruning	1.0.7
callr	Call R from R	3.7.3
car	Companion to Applied Regression	3.1-2
carData	Companion to Applied Regression Data Sets	3.0-5
cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
checkmate	Fast and Versatile Argument Checks	2.2.0
classInt	Choose Univariate Class Intervals	0.4-9
cli	Helpers for Developing Command Line Interfaces	3.6.0
clipr	Read and Write from the System Clipboard	0.8.0

# The R studio interface

The screenshot displays the RStudio interface with the following components:

- Console:** Shows the R version (4.2.2) and the execution of the command `a <- 3`. The output is `[1] "hello world"`.
- Environment:** Displays the variable `a` with the value `3`.
- Package Manager:** Lists installed and available packages, including `abind`, `askpass`, `av`, `backports`, `base64enc`, `Bi`, `bit`, `bit64`, `blob`, `broom`, `bslib`, `cachem`, `callr`, `car`, `carData`, `cellranger`, `checkmate`, `classInt`, `cli`, and `clipr`.

**Console Output:**

```
R 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

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.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from C:/Users/becky/Desktop/Github/RETA-Analysis-2017-2023/Data/.RData]
> print("hello world")
[1] "hello world"
> a <- 3
>
```

**Environment:**

values	
a	3

**Package Manager:**

Name	Description	Version
abind	Combine Multidimensional Arrays	1.4-5
askpass	Safe Password Entry for R, Git, and SSH	1.1
av	Working with Audio and Video in R	0.8.3
backports	Reimplementations of Functions Introduced Since R-3.0.0	1.4.1
base64enc	Tools for base64 encoding	0.1-3
Bi	Boost C++ Header Files	1.81.0-1
bit	Classes and Methods for Fast Memory-Efficient Boolean Selections	4.0.5
bit64	A S3 Class for Vectors of 64bit Integers	4.0.5
blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBs')	1.2.3
broom	Convert Statistical Objects into Tidy Tibbles	1.0.3
bslib	Custom 'Bootstrap' Sass Themes for 'shiny' and 'markdown'	0.4.2
cachem	Cache R Objects with Automatic Pruning	1.0.7
callr	Call R from R	3.7.3
car	Companion to Applied Regression	3.1-2
carData	Companion to Applied Regression Data Sets	3.0-5
cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
checkmate	Fast and Versatile Argument Checks	2.2.0
classInt	Choose Univariate Class Intervals	0.4-9
cli	Helpers for Developing Command Line Interfaces	3.6.0
clipr	Read and Write from the System Clipboard	0.8.0

**Text Overlay:**

We can also define variables here

try a <- 3

# The R studio interface

The screenshot displays the RStudio interface with the following components:

- Console:** Shows the R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting" and the workspace loaded from C:/Users/becky/Desktop/Github/RETA-Analysis-2017-2023/Data/.RData. The console output includes the command `print("hello world")` and the variable `a` assigned the value 3.
- Environment:** Displays the Global Environment with the variable `a` having the value 3.
- Package Manager:** Shows the User Library with a list of installed and available packages, including `abind`, `askpass`, `av`, `backports`, `base64enc`, `BH`, `bit`, `bit64`, `blob`, `broom`, `bslib`, `cachem`, `callr`, `car`, `carData`, `cellranger`, `checkmate`, `classInt`, `cli`, and `clipr`.

An orange box with the text "This shows what variables have been defined" is overlaid on the Environment pane.

# The R studio interface

The screenshot displays the RStudio interface with the following components:

- Console:** Shows the R version 4.2.2 (2022-10-31 ucrt) -- "Innocent and Trusting" and the R startup message. It also shows the execution of the following code:

```
> print("hello world")
[1] "hello world"
> a <- 3
> b <- 5
> c <- a + b
```
- Environment:** Displays the Global Environment with the following values:

values	
a	3
b	5
c	8
- Package List:** A list of installed and available packages, including: cellranger, checkmate, classInt, cli, clipr, dplyr, ggplot2, gtable, grid, gridExtra, gtable, gtableExtra, gtableExtra2, gtableExtra3, gtableExtra4, gtableExtra5, gtableExtra6, gtableExtra7, gtableExtra8, gtableExtra9, gtableExtra10, gtableExtra11, gtableExtra12, gtableExtra13, gtableExtra14, gtableExtra15, gtableExtra16, gtableExtra17, gtableExtra18, gtableExtra19, gtableExtra20, gtableExtra21, gtableExtra22, gtableExtra23, gtableExtra24, gtableExtra25, gtableExtra26, gtableExtra27, gtableExtra28, gtableExtra29, gtableExtra30, gtableExtra31, gtableExtra32, gtableExtra33, gtableExtra34, gtableExtra35, gtableExtra36, gtableExtra37, gtableExtra38, gtableExtra39, gtableExtra40, gtableExtra41, gtableExtra42, gtableExtra43, gtableExtra44, gtableExtra45, gtableExtra46, gtableExtra47, gtableExtra48, gtableExtra49, gtableExtra50, gtableExtra51, gtableExtra52, gtableExtra53, gtableExtra54, gtableExtra55, gtableExtra56, gtableExtra57, gtableExtra58, gtableExtra59, gtableExtra60, gtableExtra61, gtableExtra62, gtableExtra63, gtableExtra64, gtableExtra65, gtableExtra66, gtableExtra67, gtableExtra68, gtableExtra69, gtableExtra70, gtableExtra71, gtableExtra72, gtableExtra73, gtableExtra74, gtableExtra75, gtableExtra76, gtableExtra77, gtableExtra78, gtableExtra79, gtableExtra80, gtableExtra81, gtableExtra82, gtableExtra83, gtableExtra84, gtableExtra85, gtableExtra86, gtableExtra87, gtableExtra88, gtableExtra89, gtableExtra90, gtableExtra91, gtableExtra92, gtableExtra93, gtableExtra94, gtableExtra95, gtableExtra96, gtableExtra97, gtableExtra98, gtableExtra99, gtableExtra100.

An orange overlay box contains the text: "See that we wrote a new variable 'b' and added it to 'a' to make 'c'"

# The R studio interface

The screenshot displays the RStudio interface with the following components:

- Console:** Shows the R startup message and the execution of the following code:

```
> print("hello world")
[1] "hello world"
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
```
- Environment:** Displays the global environment with the following values:

values	
a	3
b	5
c	8
- Package List:** A list of installed and available packages, including:

Package	Version
cellranger	1.1.0
checkmate	2.2.0
classInt	0.4-9
cli	3.6.0
clipr	0.8.0

An orange callout box contains the text: "We can also feed variables into other functions such as print... try: print(c)".

# The R studio interface

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains a script with the following code:

```
1 a <- 3
2 b <- 5
3 c <- a + b
4 print(c)
5
```
- Environment Pane:** Shows the current environment with the following values:

Variable	Value
a	3
b	5
c	8
- Console:** Shows the output of the script execution:

```
R 4.2.2 > C:/Users/becky/Desktop/Github/RETA-Analysis-2017-2023/Data/ >
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from C:/Users/becky/Desktop/Github/RETA-Analysis-2017-2023/Data/.RData]

> print("hello world")
[1] "hello world"
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
```
- User Library:** A list of installed R packages with their versions and descriptions.

**Text overlay:**

This is the script, or recipe that you can save!

Try file, save to save the code you have written.



# The R studio interface

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains a script with the following code:

```
1 a <- 3
2 b <- 5
3 c <- a + b
4 print(c)
5
```
- Environment:** Shows the current environment with the following values:

values
a
3
b
5
c
8
- Console:** Shows the output of the code execution:

```
[1] "hello world"
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
```
- Callout Box:** An orange box with an arrow pointing to the 'Run' button in the Source Editor. It contains the text: "You can press 'run' or ctrl+entr to run the code line by line, or highlight whole sections to run at once."

# The R studio interface

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code:

```
1 a <- 3
2 b <- 5
3 c <- a + b
4 print(c)
5
```
- Console:** Shows the output of the code:

```
R 4.2.2 - C:/Users/becky/Desktop/Github/RETA-Analysis-2017-2023/Data/ >
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from c:/Users/becky/Desktop/Github/RETA-Analysis-2017-2023/Data/.RData]

> print("hello world")
[1] "hello world"
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
> a <- 3
> b <- 5
> c <- a + b
> print(c)
[1] 8
```
- Environment Pane:** Shows the current workspace with variables 'a', 'b', and 'c'. A broom icon is located at the top right of this pane.
- Callout Box:** An orange box with white text pointing to the broom icon:

This little broom is used to clear the workspace... or delete all variables.

Make sure your final code runs after clearing to make sure you don't miss anything



# Data Types

## *using "typeof()"*

### Numeric

```
var1 <- 42  
var2 <- 9.81  
  
print(typeof(var1))  
print(typeof(var2))
```

### Integer

```
var1 <- 5L  
var2 <- 15L  
  
print(typeof(var1))  
print(typeof(var2))
```

### Data frame

```
var1 <- data.frame(  
  ID = 1:3,  
  Name = c("Alice", "Bob", "Charlie"),  
  Age = c(25, 30, 35))  
  
print(typeof(var1))
```

### Logical

```
var1 <- TRUE  
var2 <- FALSE  
  
print(typeof(var1))  
print(typeof(var2))
```

### Character

```
var1 <- "Hello, world!"  
var2 <- "405"  
  
print(typeof(var1))  
print(typeof(var2))
```

### Vector

```
var1 <- c(1,2,3,4,5)  
var2 <- c("1", "hi", "IPB")  
  
print(typeof(var1))  
print(typeof(var2))
```

# Common operators:

## *What do they mean?*

```
a <- 5  
b <- 3  
result <- (a + b) * b  
print(result)
```

```
p <- TRUE  
q <- FALSE  
print(p & q)
```

```
vec1 <- c(1, 2, 3)  
vec2 <- c(4, 5, 6)  
result <- vec1 + vec2  
print(result)
```

```
a <- 7  
b <- 4  
print(a > b)
```

```
x <- 10  
x <- x + 5  
print(x)
```

# Common operators:

## *Answers*

```
a <- 5  
b <- 3  
result <- (a + b) * b  
print(result)  
24
```

```
p <- TRUE  
q <- FALSE  
print(p & q)  
False
```

```
vec1 <- c(1, 2, 3)  
vec2 <- c(4, 5, 6)  
result <- vec1 + vec2  
print(result)  
c(5, 7, 9)
```

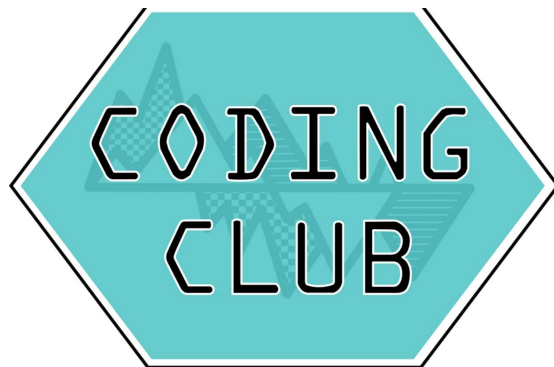
```
a <- 7  
b <- 4  
print(a > b)  
TRUE
```

```
x <- 10  
x <- x + 5  
print(x)  
15
```

## Now try: Loading in a dataset

Go to:

<https://ourcodingclub.github.io/tutorials/intro-to-r/>



# Introduction to GitHub

- Introduction to R
- Introduction to GitHub
- Plan for the rest of the session
- Where to find help

# GitHub

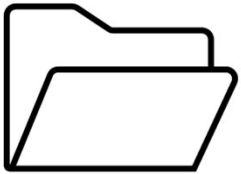
A platform for sharing,  
storing, and collaborating  
on code projects.



# Basics of what using GitHub is like



1) Create a repository



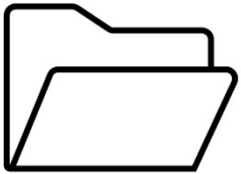
@Becky\_Heath

rh862@cam.ac.uk

# Basics of what using GitHub is like



1) Create a repository



2) Upload to GitHub



@Becky\_Heath

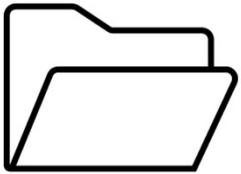
rh862@cam.ac.uk



# Basics of what using GitHub is like



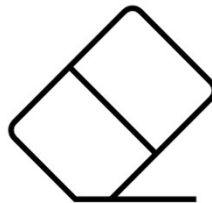
1) Create a repository



2) Upload to GitHub



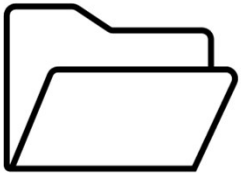
3) Make changes locally



# Basics of what using GitHub is like



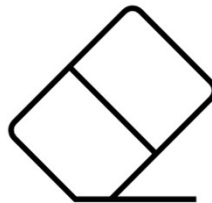
1) Create a repository



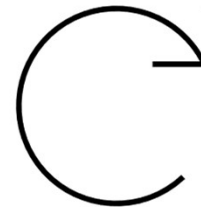
2) Upload to GitHub



3) Make changes locally



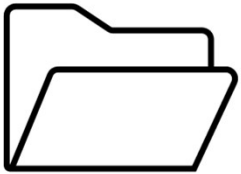
4) Send updates to GitHub



# Basics of what using GitHub is like



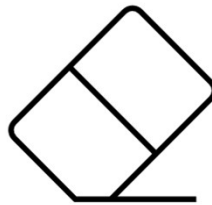
1) Create a repository



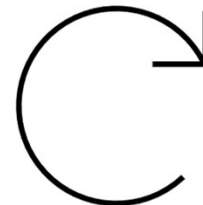
2) Upload to GitHub



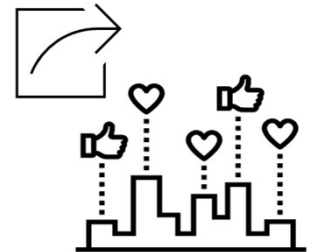
3) Make changes locally



4) Send updates to GitHub

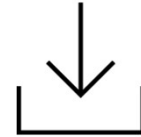


5) Share/collaborate



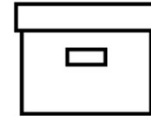
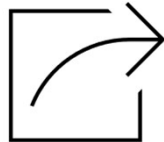
# Why bother using GitHub?

Keep a record of your work,  
revisit old versions.



Use other's work for your  
own projects

Share code and collaborate  
more easily.

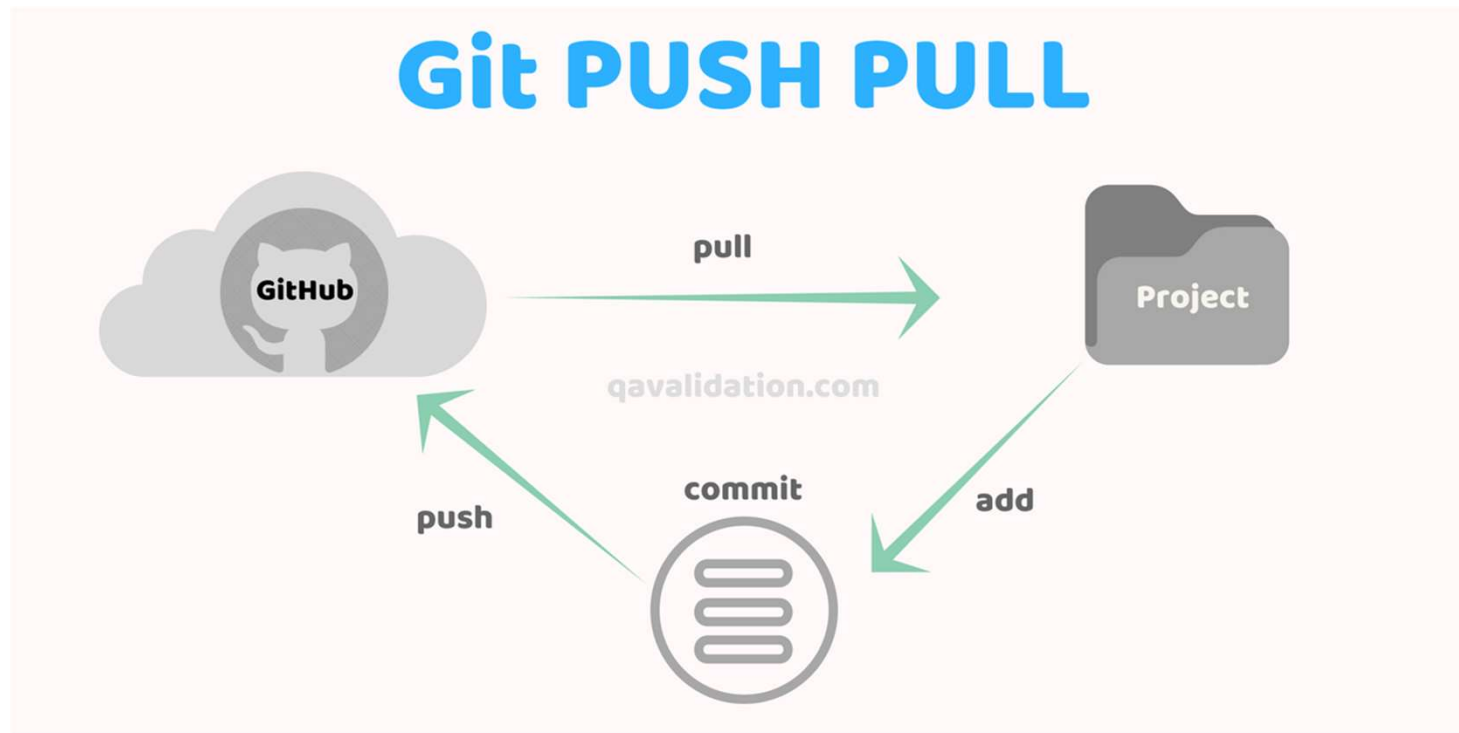


Online backup/ archive

Publish code in journals.



# GitHub Core Concepts

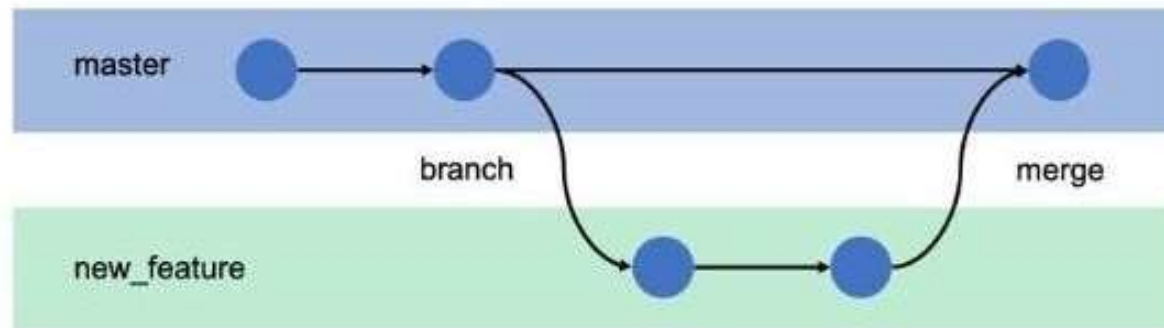


@Becky\_Heath

rh862@cam.ac.uk

## GitHub Core Concepts

# GIT BRANCHES



# GitHub Desktop



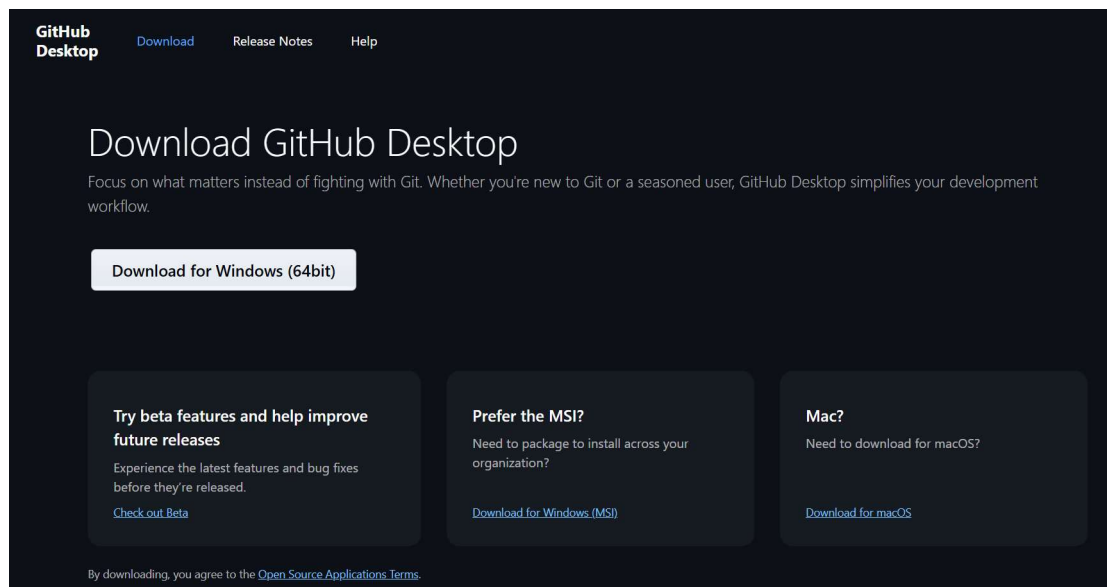
GitHub Desktop is a friendly interface for working with GitHub repos

Here you can create a new repo locally, clone repos, and push to your GitHub account online

# Getting started with GitHub

1) Download GitHub Desktop (this will also download Git)

<https://docs.github.com/en/desktop/installing-and-authenticating-to-github-desktop/setting-up-github-desktop#>



rh862@cam.ac.uk



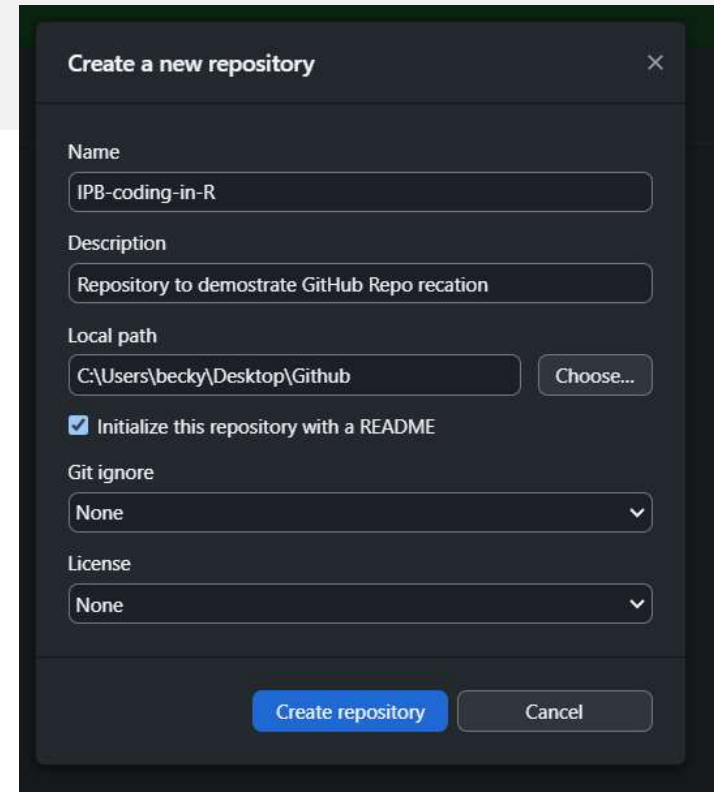
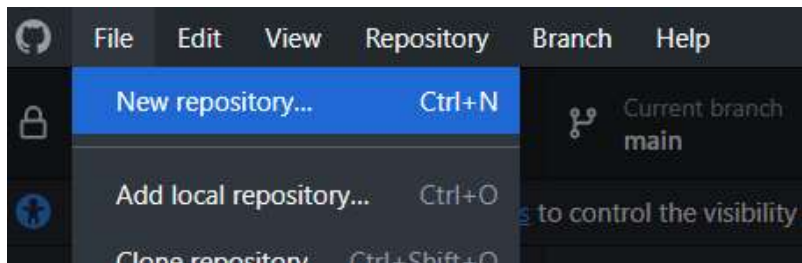
## Getting started with GitHub

2) Authenticate by signing in (you'll need to create an account if you don't have one)  
<https://docs.github.com/en/desktop/installing-and-authenticating-to-github-desktop/authenticating-to-github-in-github-desktop>



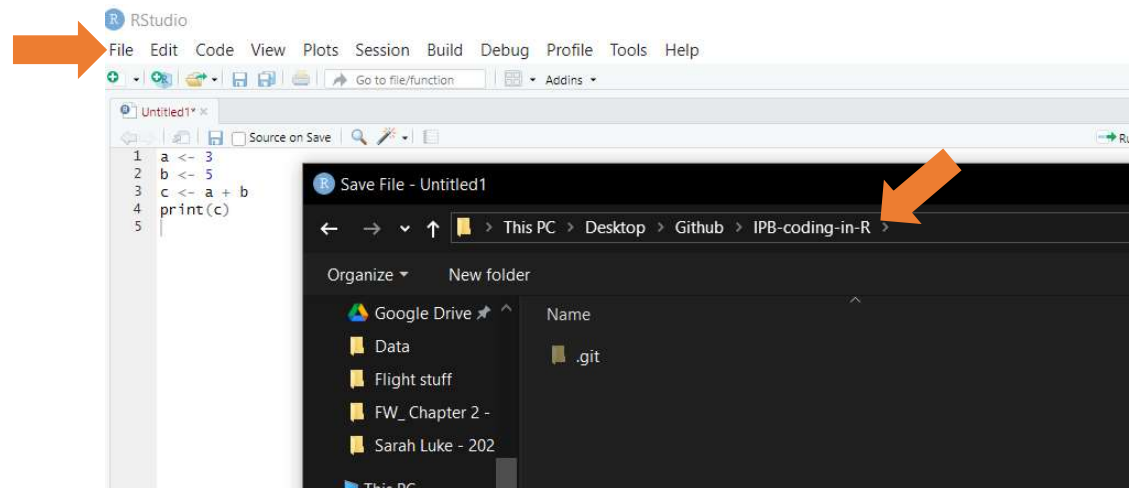
# Getting started with GitHub

## 3) Create a new repository



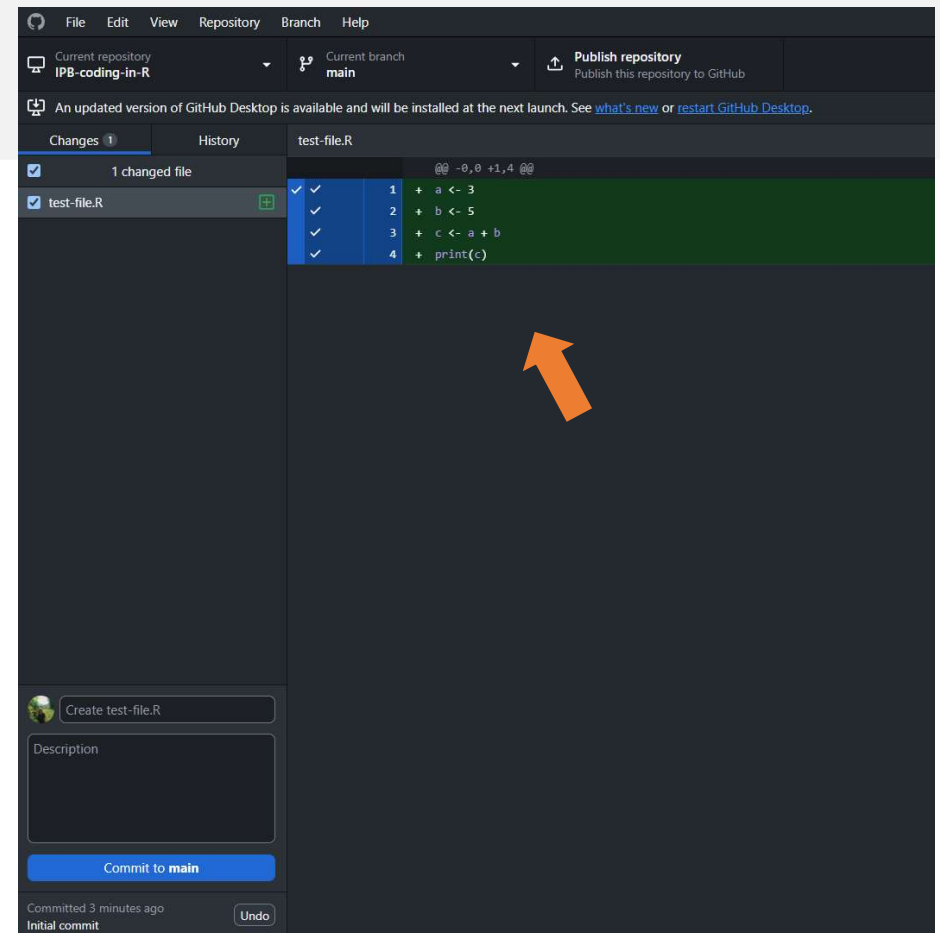
# Getting started with GitHub

## 4) Save R file to new repo



# Getting started with GitHub

5) Check that the new file/ edits have been picked up by github desktop

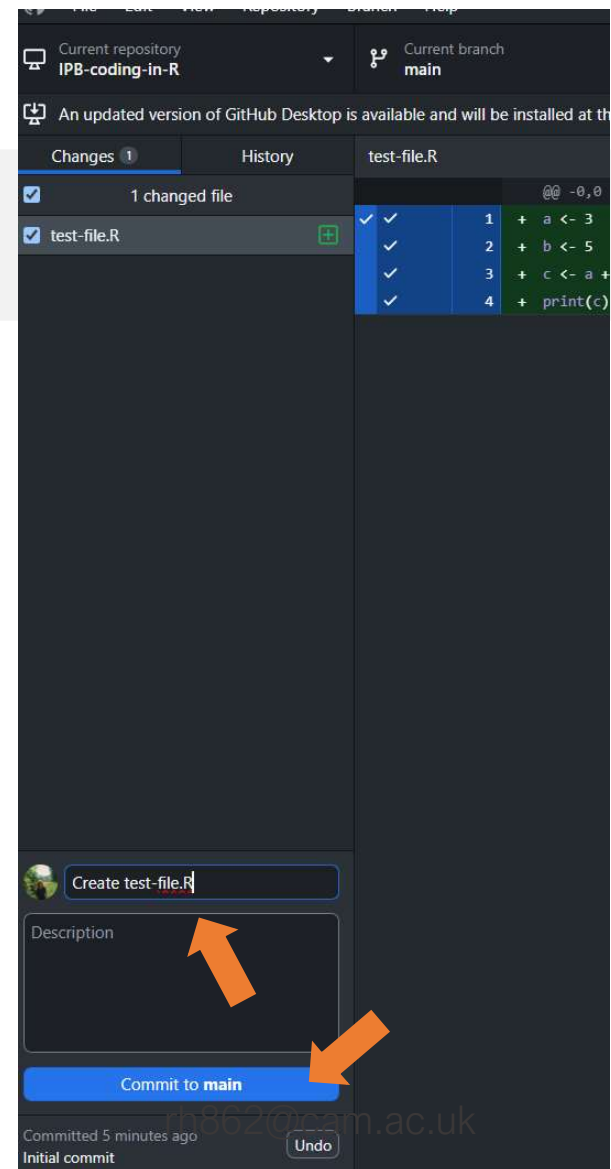


@Becky\_Heath

rh862@cam.ac.uk

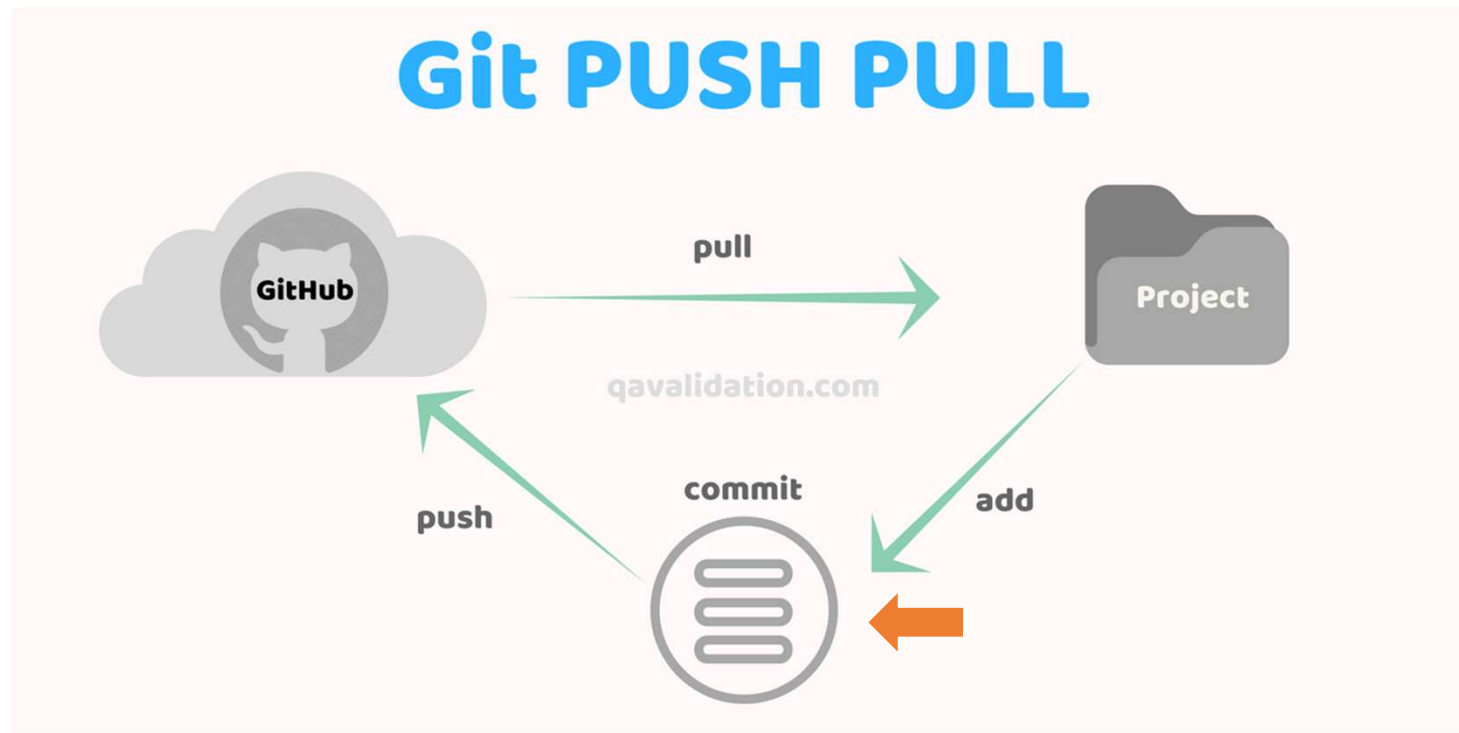
# Getting started with GitHub

6) Write a description for the commit  
(try to be informative!)



@Becky\_Heath

# GitHub Core Concepts



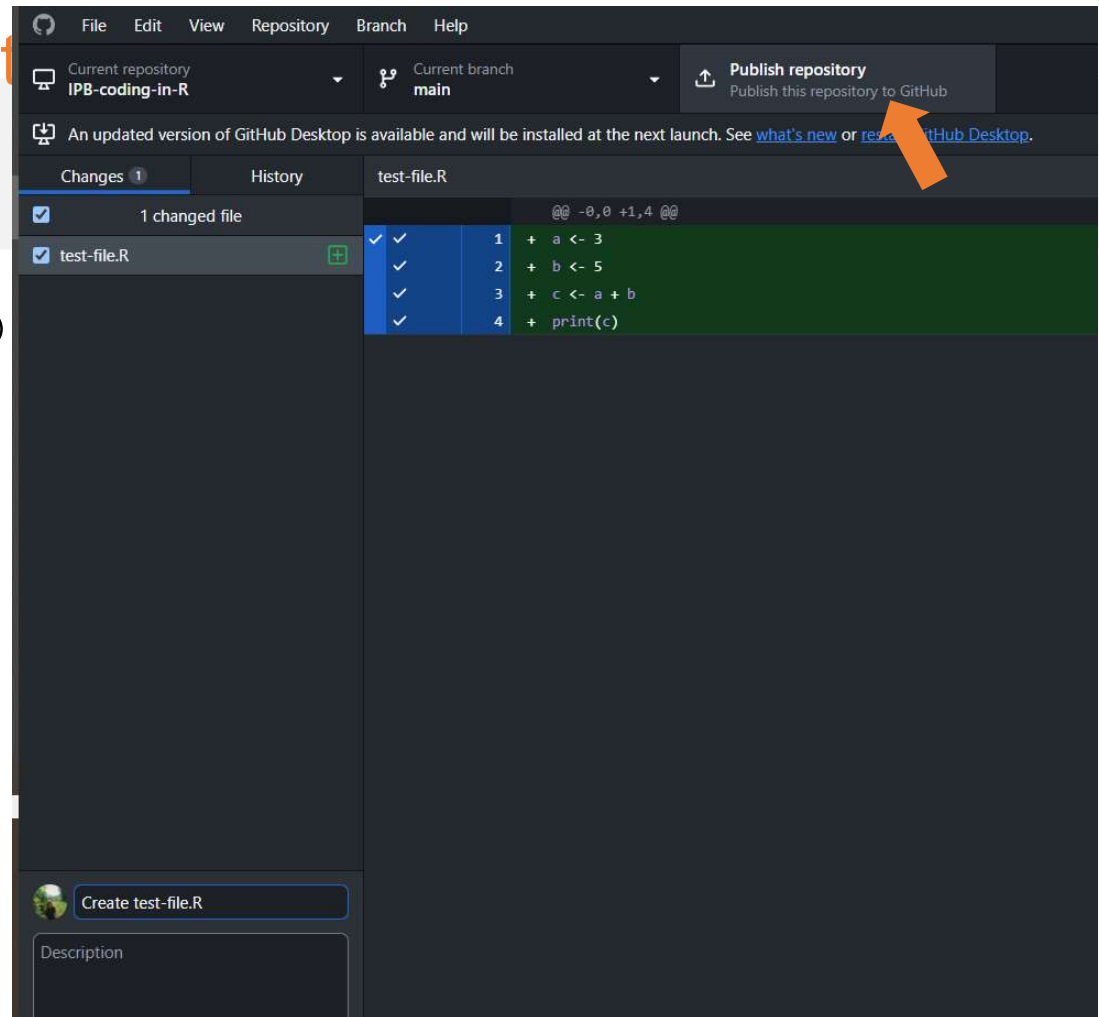
@Becky\_Heath

rh862@cam.ac.uk

## Getting started with Git

7) Push the commit to GitHub ☺

NB: it says publish the first time  
but it will say push after

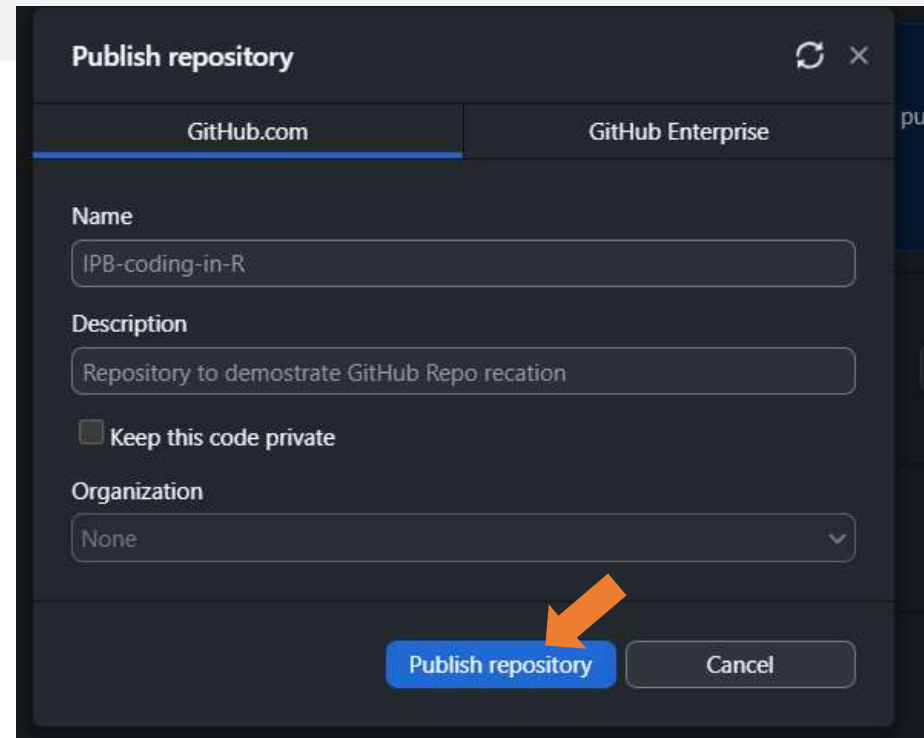


## Getting started with GitHub

6) only the first time:

check details are correct and  
decide if you want the repo to be  
public or not

.... then publish



The screenshot shows a 'Publish repository' dialog box with a dark theme. At the top, there are two tabs: 'GitHub.com' (selected) and 'GitHub Enterprise'. Below the tabs, the 'Name' field contains 'IPB-coding-in-R'. The 'Description' field contains 'Repository to demonstrate GitHub Repo recation'. There is a checkbox labeled 'Keep this code private' which is currently unchecked. The 'Organization' dropdown menu is set to 'None'. At the bottom right, there are two buttons: 'Publish repository' (highlighted with a blue background and an orange arrow pointing to it) and 'Cancel'.

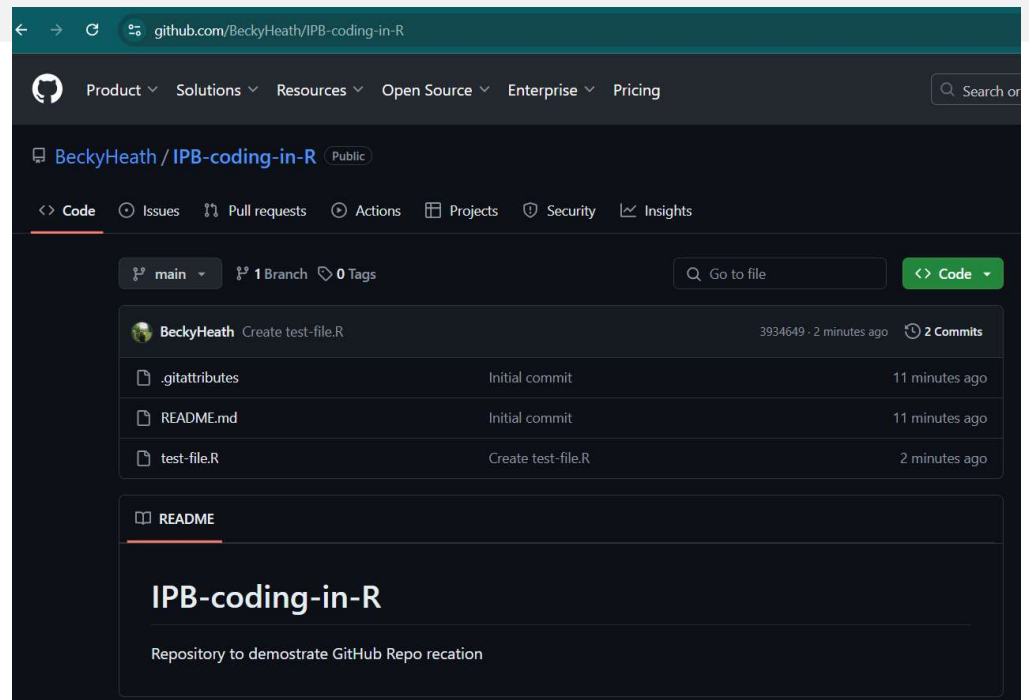


# Getting started with GitHub

6) Check the repo is online!!

To check, go to

`github.com/[USERNAME]/[REPO-NAME]`



@Becky\_Heath

rh862@cam.ac.uk

# Rest of the session

- Introduction to coding and R
- Introduction to GitHub
- Plan for the rest of the session
- Where to find help

# Plan for the rest of the session



Work through tutorials  
online at your own pace

@Becky\_Heath



Try to find solutions  
online (next section).



Ask for help if you can't  
find a solution

rh862@cam.ac.uk

# Selected Tutorials



UNIVERSITY OF  
CAMBRIDGE

Code Club (University of Edinburgh)

@Becky\_Heath

rh862@cam.ac.uk

# Where to find help

- Introduction to coding and R
- Introduction to GitHub
- Plan for the rest of the session
- Where to find help

# Using error messages?

The fastest and easiest way to debug code in R is to type the error message into google and test out solutions

Best to use public forums such as “cross check”, “stack overflow” and “stack exchange”.

```
4
5 df <- data.frame(id = 1, source_id = 1)
6
7 df %>% left_join(source %>% select(aaaaa))
8
9
10
```

R code execution error

7:41 (Top Level) ↕

Console ~/

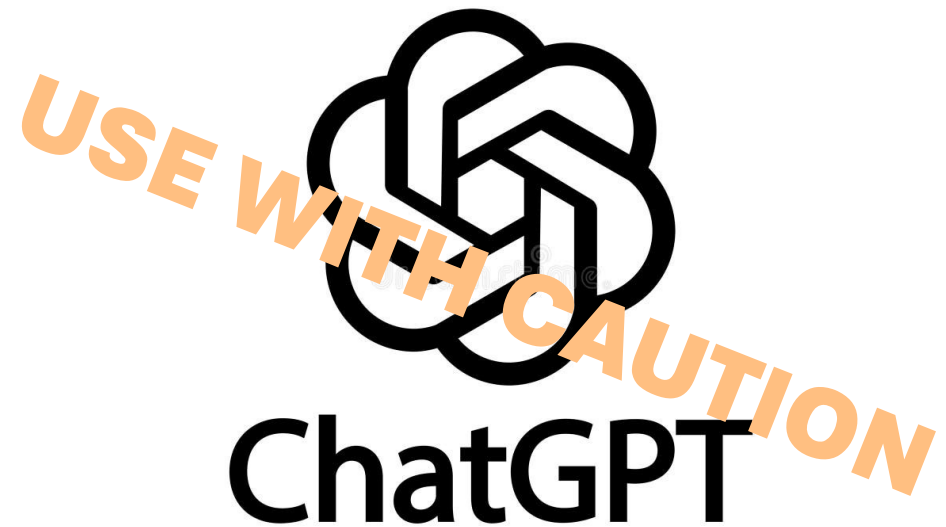
```
> df <- data.frame(id = 1, source_id = 1)
Error in leftData[completions] :
  object of type 'closure' is not subsettable
Error in leftData[completions] :
  object of type 'closure' is not subsettable
Error in leftData[completions] :
  object of type 'closure' is not subsettable
Error in leftData[completions] :
  object of type 'closure' is not subsettable
Error in leftData[completions] :
  object of type 'closure' is not subsettable
Error in leftData[completions] :
  object of type 'closure' is not subsettable
Error in leftData[completions] :
  object of type 'closure' is not subsettable
> |
```

# Using AI for debugging?

Chat GPT is an algorithm that is really good at picking out patterns in human language.

It is very good at sounding correct, and is very often good at finding simple solutions, but not complex issues.

Chat GPT gets stuff wrong a *lot*.



# How do I get good at coding?

Practice

@Becky\_Heath

rh862@cam.ac.uk



# How do I get good at coding?

Practice

... sorry

@Becky\_Heath

rh862@cam.ac.uk

# Selected Tutorials



UNIVERSITY OF  
CAMBRIDGE

Code Club (University of Edinburgh)