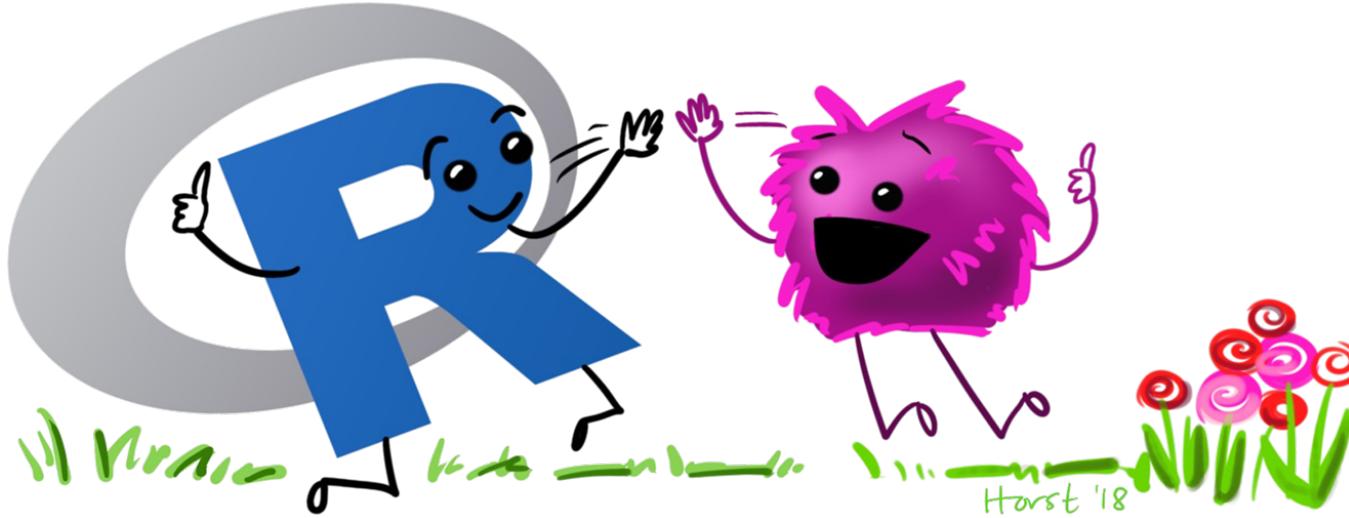


Convert your code for collaboration



Dr. Fonti Kar

Illustrations by Allison Horst

Workshop description:

Want to share your code alongside a publication or with your colleagues? How do you best format your scripts so others can use it? Come along to learn about the best practices for translating your curated scripts into more usable functions for collaborations. We will talk about functions, abstraction, modularisation, documenting your code using rendered reports to aid collaborations and reproducibility. This is a hands-on workshop, bring along your R script and computer.

Prerequisites:

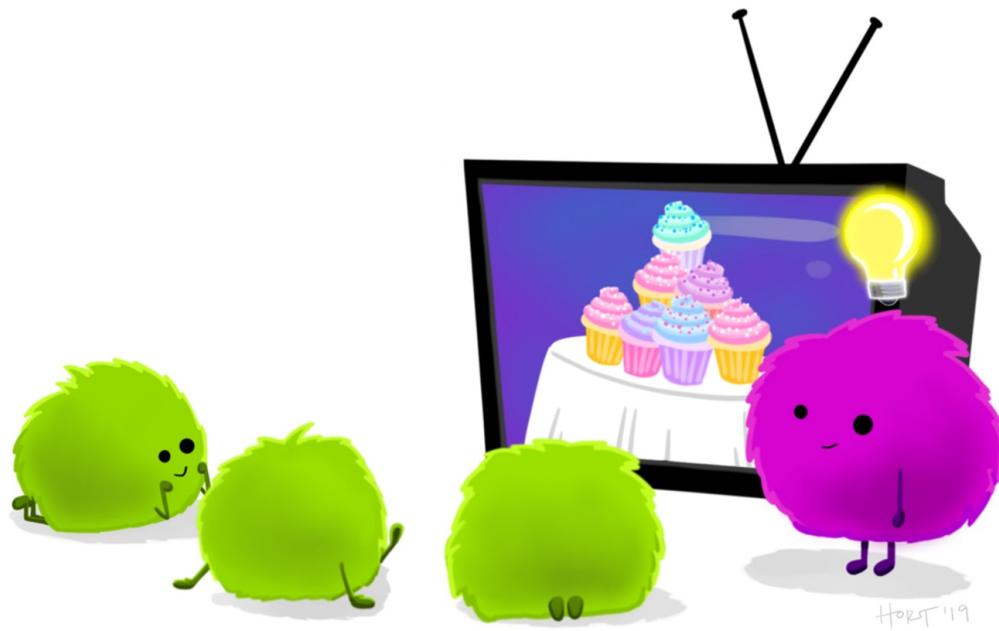
- You have written R functions before
- Install R/RStudio
- Install Rmarkdown
`install.packages("rmarkdown")`



Overview

- Functions
 - Why write them?
 - Some tips
- Abstraction
- Modularisation
- Documenting for future us
 - Roxygen
 - Rmarkdown

Hands on edits on your own script!



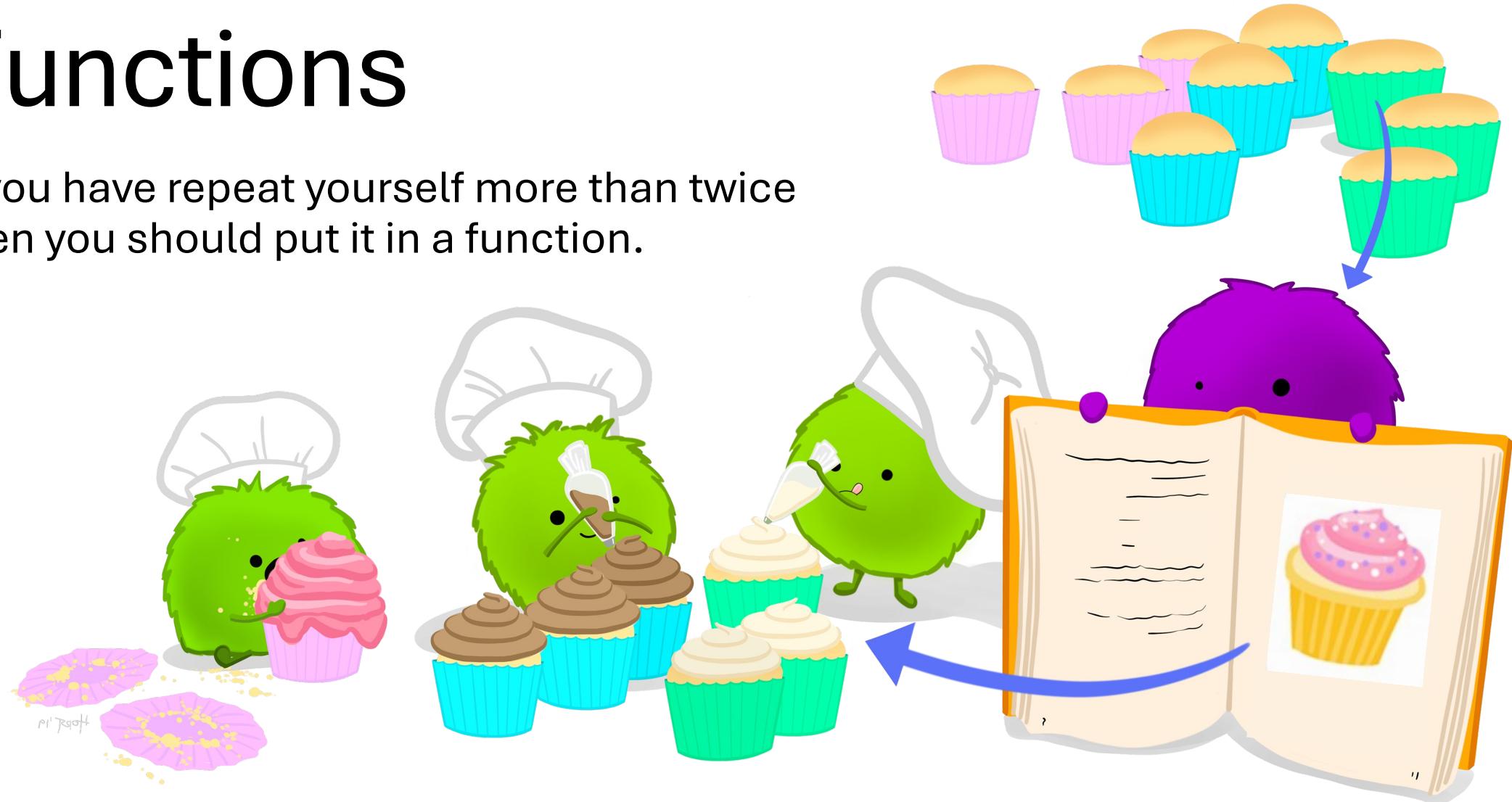
Why write functions

- A quick group discussion 3 mins



Functions

If you have repeat yourself more than twice
then you should put it in a function.



Why write functions

1. Improve coding skills
2. More readable code
 - Contain the messiness in an organised manner
3. Tidy R workspace
4. You only need to update code in one place
 - No more search and replace!
 - Reduce incidental mistakes when you copy and paste
 - i.e. updating a variable name in one place, but not in another.
5. Reuse work from project-to-project 



Function writing tips

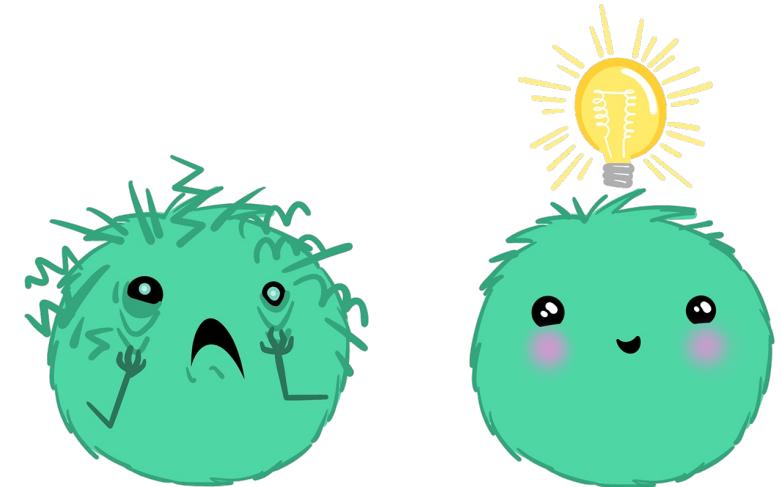
- A good **declarative** name
- Input names + defaults are **intuitive**
- **Comments** within function
- Dependencies are **explicit** using ::



```
get_latest_download <- function(path_to_galah_data = 'data/galah/'){  
  # Grab all files in folder  
  all_downloads <- list.files(path_to_galah_data)  
  
  # Grab the dates only  
  dates <- stringr::str_extract(all_downloads,  
                                pattern = regex("[0-9]{4}--[0-9]{2}--[0-9]{2}"))  
  
  # Determine the most recent date  
  most_recent <- max(lubridate::ymd(dates)) %> as.character()  
  
  # Identify the file name that is the most recent  
  most_recent_file <- stringr::str_subset(all_downloads, most_recent)  
  
  # Get path to most recent file  
  paste0(path_to_galah_data, most_recent_file)  
}
```

Abstraction

- Identifying patterns, giving them a name to simplify the wider problem
- E.g.
 - Draw a square
 - Draw a rectangle with sides of equal length
 - Draw a triangle
 - Draw a house
- **Feature or task focused**
 - NOT details about your project, or data or variables
 - Creates more readable + reusable code



Abstraction

I want to add 2 to my values!

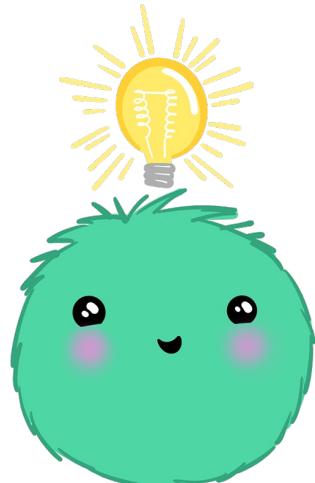
```
> add_2 <- function(y) {  
+   2 + y  
+ }  
> add_2(1:10)  
[1] 3 4 5 6 7 8 9 10 11 12
```

Actually...I need to also add 7
to some other values

```
> add_7 <- function(y) {  
+   7 + y  
+ }  
> add_7(1:10)  
[1] 8 9 10 11 12 13 14 15 16 17
```

Write a function for adding
things!

```
> add_x <- function(y, x) {  
+   y + x  
+ }  
> add_x(1:10, 7)  
[1] 8 9 10 11 12 13 14 15 16 17  
> add_x(1:10, 30)  
[1] 31 32 33 34 35 36 37 38 39 40
```

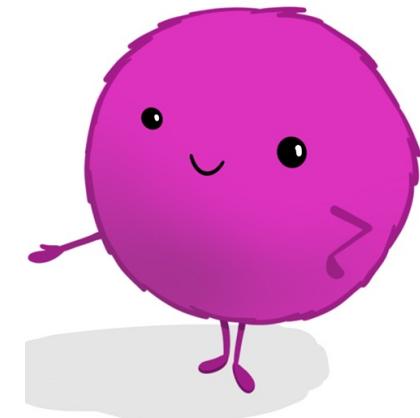


Tips on being more abstract

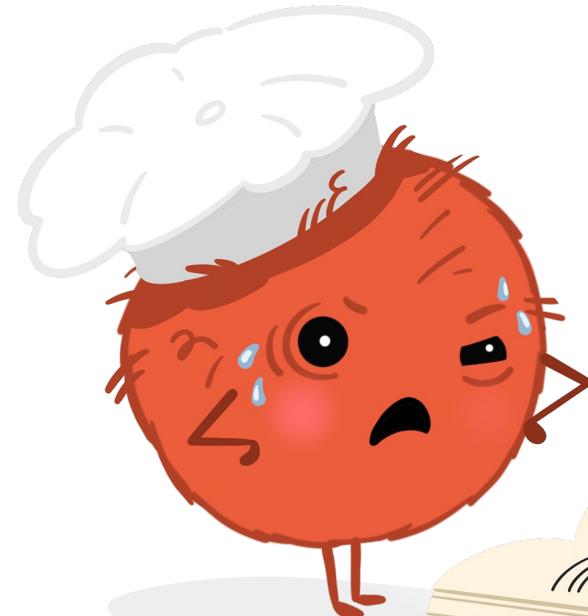
- Keep it high level
- Zoom out, **generalise**
- Focus on the tasks/verbs
 - Creating a variable
 - Excluding values
 - Summarise by X
- Explain to someone what your function is doing without any context
 - Nothing about your question/methods/data

Try it yourself!

Look at your script, what are some patterns you can identify?
Can you pull these out and generalise it with a function?

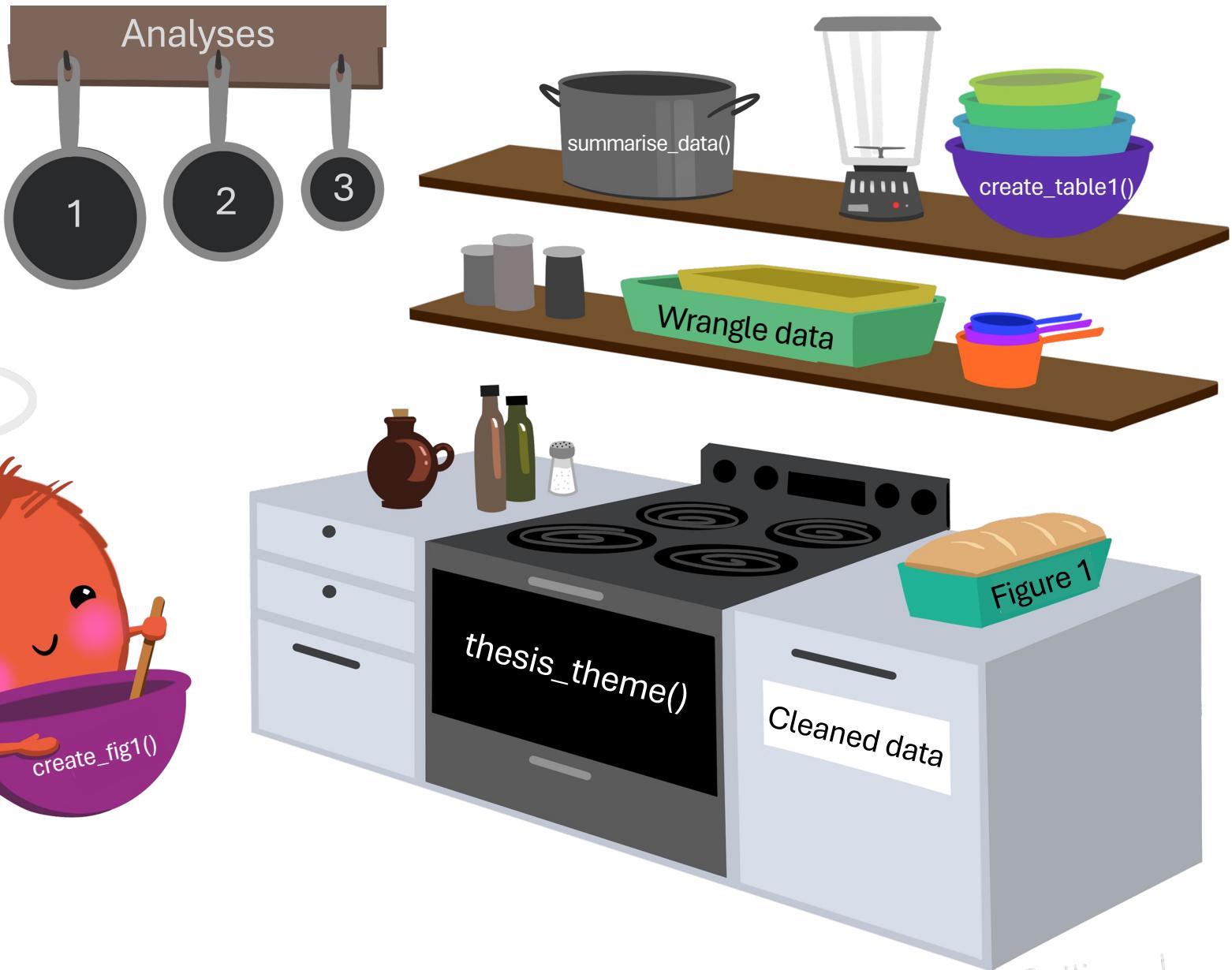


Break up complexity



@allison_horst

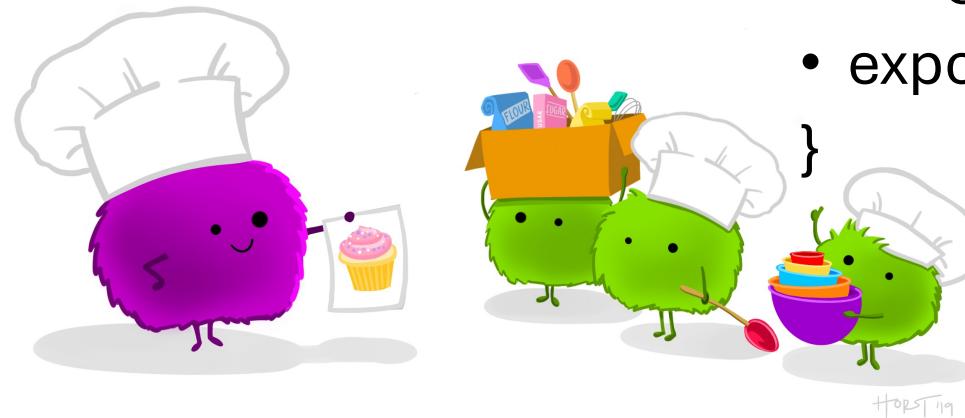
Modularisation



@allison_horst

Isolate the main ‘steps’

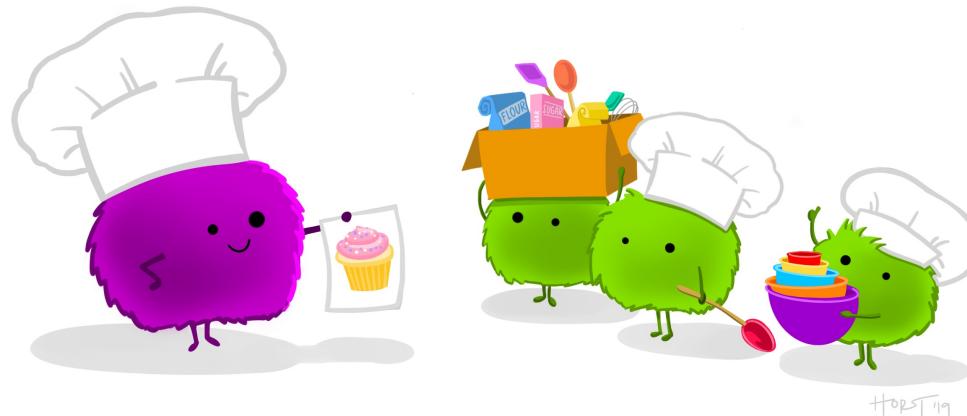
- Make cupcake
 - Gather ingredients
 - Gather equipment
 - Measure our ingredients
 - Mix dry ingredients
 - Mix wet ingredients
 - Put into cake pan
 - Bake
 - Decorate



- create_fig1 {
 - clean_data()
 - fig1_wrangle_data()
 - my_phd_theme()
 - fig_base_plot()
 - Legends
 - Axis title
 - Colour of grouping variable
 - export_figure()

Try it yourself!

- Pull out the main tasks in your script
- Can you put these into separate functions
- Can you put these sub functions into a wrapper?



Documenting for future us



You will thank yourself later

roxygen2 templates



```
#' Get the latest galah download
#'
#' @param path_to_galah_data path where parquets of ALA downloads are stored
#'
#' @return path to most recent galah download
#'
#' @examples
#' arrow::open_dataset(get_latest_download())
#
get_latest_download <- function(path_to_galah_data = 'data/galah/'){
  # Grab all files in folder
  all_downloads <- list.files(path_to_galah_data)

  # Grab the dates only
  dates <- stringr::str_extract(all_downloads,
                                pattern = regex("[0-9]{4}--[0-9]{2}--[0-9]{2}"))

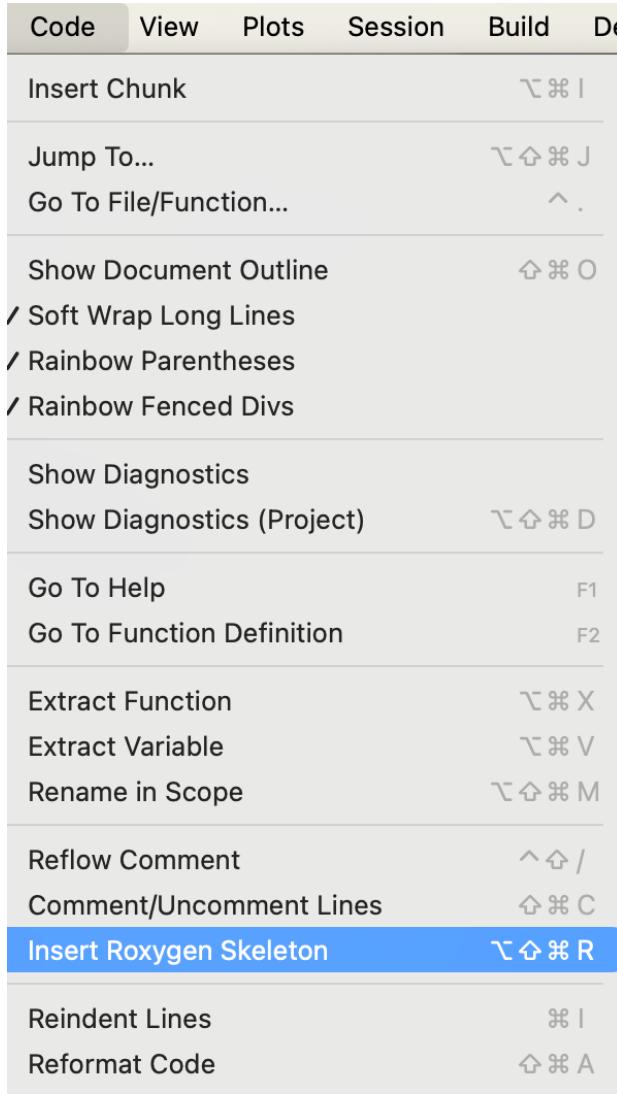
  # Determine the most recent date
  most_recent <- max(lubridate::ymd(dates)) %> as.character()

  # Identify the file name that is the most recent
  most_recent_file <- stringr::str_subset(all_downloads, most_recent)

  # Get path to most recent file
  paste0(path_to_galah_data, most_recent_file)
}
```

- Meta-data for your functions
- Breaks up long .R files
 - More keywords to Ctrl/CMD-F
- Quick transition to an R package
- Guides user to what they need to know about your function

Try it yourself!



- Document 1 of your own functions using roxygen

```
#' Get the latest galah download
#'
#' @param path_to_galah_data path where parquets of ALA downloads are stored
#'
#' @return path to most recent galah download
#'
#' @examples
#' arrow::open_dataset(get_latest_download())

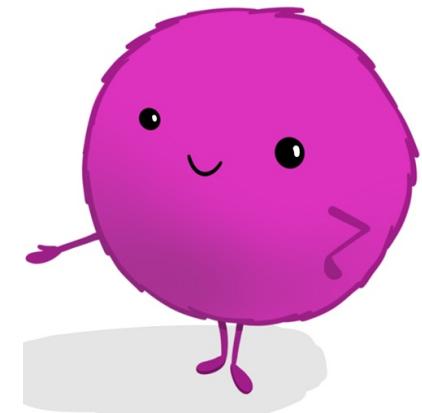
get_latest_download <- function(path_to_galah_data = 'data/galah/'){
  # Grab all files in folder
  all_downloads <- list.files(path_to_galah_data)

  # Grab the dates only
  dates <- stringr::str_extract(all_downloads,
                                pattern = regex("[0-9]{4}--[0-9]{2}--[0-9]{2}"))

  # Determine the most recent date
  most_recent <- max(lubridate::ymd(dates)) %> as.character()

  # Identify the file name that is the most recent
  most_recent_file <- stringr::str_subset(all_downloads, most_recent)

  # Get path to most recent file
  paste0(path_to_galah_data, most_recent_file)
}
```



Rendered magic!

- Traditional R scripts are hard to read
- Rmarkdown allows you to interweave **text, code, outputs, figures** into a static report
- E.g.
https://fontikar.github.io/DIY_Rpkg/



The mise en place 🍷

- The main set up
- Structure of your R package
- Edit your DESCRIPTION file manually
- Declare your license
- Routine checks
- Your first function
- Document it - Object documentation ✓
- README - Your landing page 😊
- Long form documentation 📖
- Finally, lets talk about tests...📝
- You have yourself an R package! 😊

DIY R package

Fonti Kar

19/07/2021

Below are some of the main functions you will need to create your first R package! 🤘

Please note ► that some of the explanations are heavily borrowed from Hadley Wickham and Jenny Byran's book - "R Packages: Organise, Test Document and Share your Code" which is open access! This is my go-to reference if I have a question about package development, highly recommended! ⭐⭐⭐⭐⭐

The mise en place 🍷

```
#install.packages("devtools")  
library(devtools) #Will load usethis and roxygen2 as well
```

The main set up

This function will create the basic skeleton of your R package 🍷

Be intentional where you create this on your computer. If you create it within another folder, R will kick up a fuss but its okay so long as you are happy with where your R package will live 😊

```
> create_package("ohwhaley")  
New project 'ohwhaley' is nested inside an existing project '.', which is rarely a good idea.  
If this is unexpected, the here package has a function, 'here::dr_here()' that reveals why '.' is regarded as a project.  
Do you want to create anyway?  
1: Nope  
2: Definitely  
3: No
```

```
# Create the basic skeleton of your package  
# create_package("package_name")  
  
usethis::create_package("ohwhaley")
```

The function will open a new Rstudio window and you will need load devtools and testthat again

Guide your users

- Do the heavy lifting for them
- Goal = aid reproducibility
- Explain:
 - What your function does
 - What input function requires
 - **Print out the input**
 - Execute your function
 - The output
 - **Print out the output**
- E.g. <https://labs.ala.org.au/>

The screenshot shows a blog post page with a dark orange header. The title is 'Quantifying species range and overlap with fire-burned areas using concave hulls'. Below the title is a brief description: 'Calculating range overlap is an efficient way to estimate the impact of natural disasters on biodiversity. Here we'll use curated datasets to compute concave hulls to visualise the spatial distribution of *Apidae* (Bees) and *Daviesia* (Bitterpeas) and their overlap with burned areas of the Black Summer fires of 2019-2020.' At the bottom of the header are several small buttons labeled 'EUKARYOTA', 'ANIMALIA', 'PLANTAE', 'SUMMARIES', 'MAPS', and 'R'.

On this page

- Download data
- Overview of data
- Pre-cleaning
- Species range overlap with fire-burned areas
- Make map
- Bonus: Vascular plants
- Final thoughts

Author

Fonti Kar
Margot Schneider



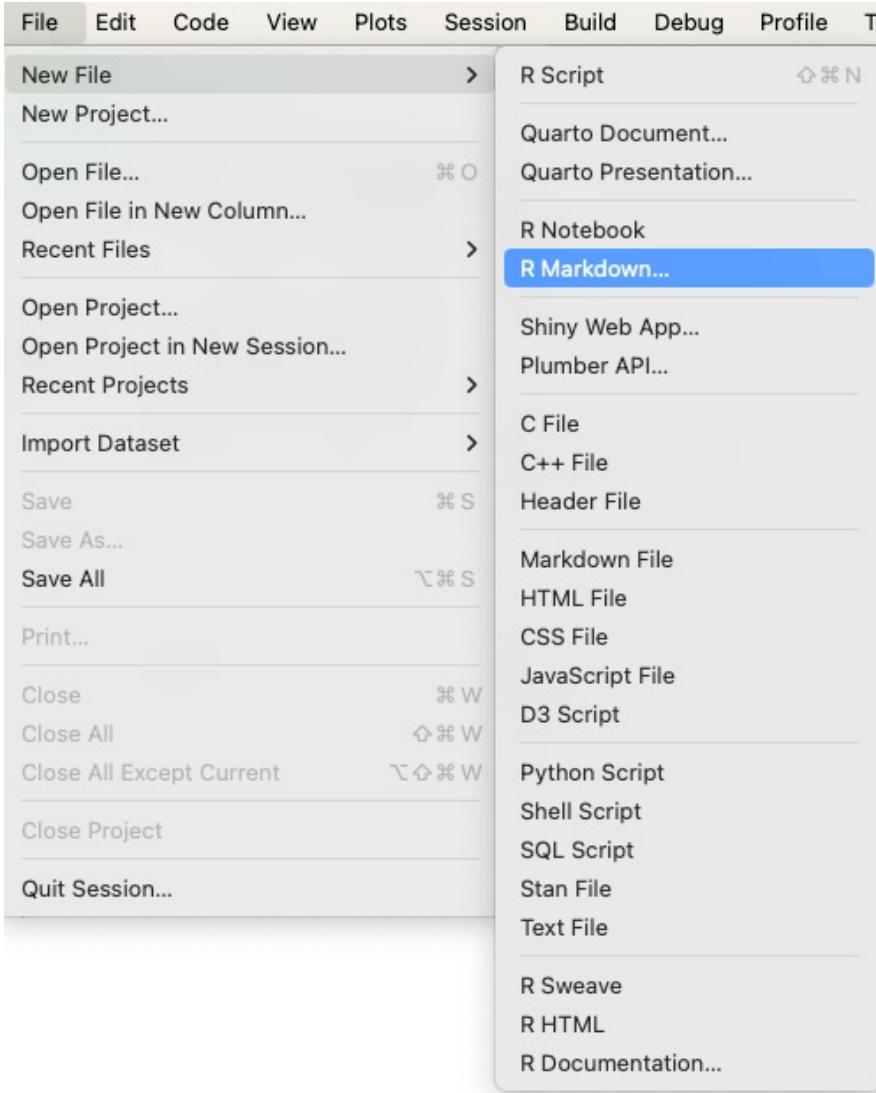
Date

11 April 2023

The 2019/2020 Australian bushfires had a devastating impact on the natural landscape, threatening our native biodiversity. More than ever, decision makers need curated, open access biodiversity data to help respond effectively to future bushfires.

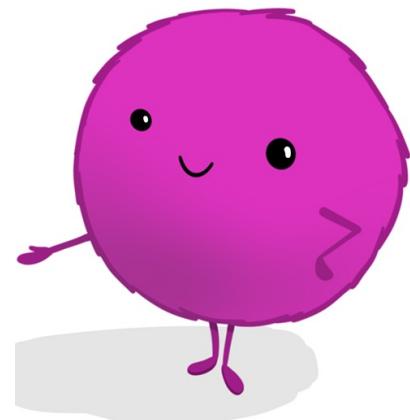
Our team at the Atlas of Living Australia (ALA) has been working with [Invertebrates](#)

Try it yourself!



- Transform your script into a .Rmd
- Break up key parts of your script using headers, text, chunks
- Try knitting it
- Not an R user? Try a Quarto document

Learn more: <https://quarto.org/docs/get-started/hello/rstudio.html>



Learn more

- On writing functions:
 - <https://r4ds.hadley.nz/functions.html>
- Functional programming/modularisation:
 - <https://adv-r.hadley.nz/fp.html>
- On communicating findings using Quarto
 - <https://r4ds.hadley.nz/communicate>
- On documenting code outputs for sustainability
 - <https://youtu.be/OVM5Ok7W1NQ?si=cf6Zzl5jQIKYhCkg>

