STAT40830
Adv Data Programming with R (online)

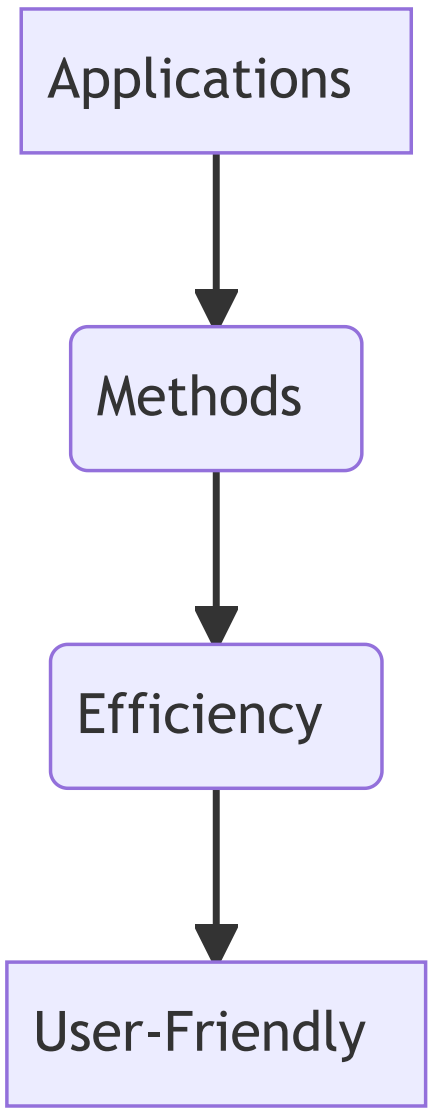
Topic 1: Good practices when working with R

Isabella Gollini

Overview

1. Programming and collaborating
2. Programming style
3. Dissemination
    - R Package
    - Reproducibility & Dynamic report
    - Create user interface
4. Version control
5. Control dependencies with `renv`

1. Programming and collaborating



Different perspectives

- Developer
- Teacher
- Learner
- Collaborator

2. Programming style

2. Programming style

- Your programming style should be accessible to allow users to understand it and potential contribute by extending improving your code.
- See: style.tidyverse.org

2.1 Commenting

- Your code is under continual development and can be a good starting point for the implementation of more advanced methodologies.
- You may need to share your code. Keep it full of comments.
- If you are working on an R script, write at the top of each functions comments that can be converted with `roxygen2` into function documentation!

2.2 roxygen2

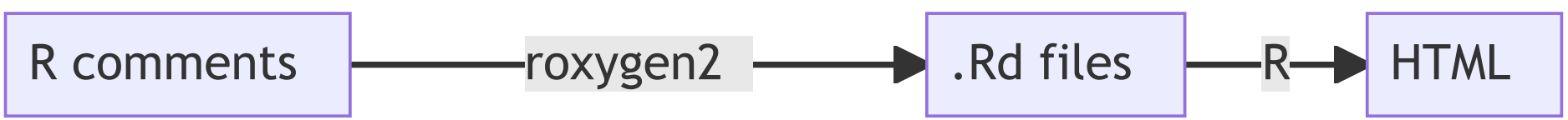
In roxygen2 you write specially formatted comments in your R script:

```
#' Add a Column to a Data Frame
#'
#' Allows you to specify the position. Will replace existing variable
#' with the same name if present.
#'
#' @param x A data frame
#' @param name Name of variable to create. If a variable of that name
#' already exists it will be replaced
#' @param value Values to insert.
#' @param where Position to insert. Use 1 to insert on LHS, or -1 to insert
#' RHS.
#' @examples
#' df <- data.frame(x = 1:5)
#' add_col(df, "y", runif(5))
#' add_col(df, "y", runif(5), where = 1)
#'
#' add_col(df, "x", 5:1)

```

2.2 roxygen2

- `#'` create **roxygen comments**
- `@param`, `@examples` are **roxygen tags**.
- You can learn more from the **Roxygen Quick Reference**. You find it by clicking on *Help* > *Roxygen Quick Reference*
- RStudio helps you set everything up when you have a function ready in your R script: leave the cursor in your function and go to *Code* > *Insert Roxygen Skeleton*
- `roxygen2` works for documenting R packages.



2.2 roxygen2

roxygen2 translates to .Rd

```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/add_col.R
\name{add_col}
\alias{add_col}
\title{Add a Column to a Data Frame}
\usage{
add_col(x, name, value, where = -1)
}
\arguments{
\item{x}{A data frame}
\item{name}{Name of variable to create. If a variable of that name already exists it will be replaced}
\item{value}{Values to insert.}
\item{where}{Position to insert. Use 1 to insert on LHS, or -1 to insert on RHS.}
}
\description{
Allows you to specify the position. Will replace existing variable with the same name if present.
}
```

- Don't modify the .Rd file if you used roxygen2!

2.2 roxygen2

R translates to HTML for viewing:

add_col R Documentation

Add a Column to a Data Frame

Description

Allows you to specify the position. Will replace existing variable with the same name if present.

Usage

```
add_col(x, name, value, where = -1)
```

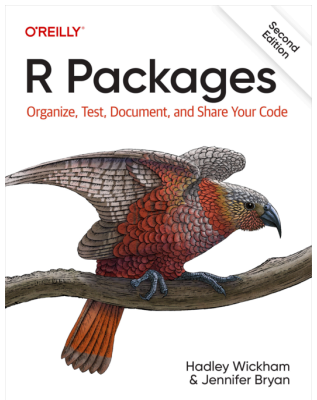
Arguments

`x`	A data frame
`name`	Name of variable to create. If a variable of that name already exists it will be replaced
`value`	Values to insert.
`where`	Position to insert. Use 1 to insert on LHS, or -1 to insert on RHS.

3. Dissemination

3.1 R package

Write your R package!



r-pkgs.org

- Start thinking about writing an R package as soon as you write your first function for your project, or even before.
- `usethis::create_package("mypackage")` will setup your package
- Check if the package name is available: `available::available("mypackage")`
- Easy to share, great documentation, an excuse to keep your folders tidy!
- You will learn more later on in this module!

3.2 Reproducibility & Dynamic report

File format easy to use for all the collaborators, such as

where you intersperse code chunks (R, Python, Julia, C++, SQL) with markdown text

- Options can be controlled on a document or chunk level whether to show code and/or output.

3.3 Not everyone loves coding

3.3 Not everyone loves coding

Choose alternative to show you code:

- Just show the output if working on a document
- Create a nice package with user friendly functions that minimise the amount of code needed
- Create nice interfaces with

- You will learn more later on in this module!

4. Version control

4. Version control

Git & GitHub/GitLab are useful:

- for Version Control
- To share your work
- To collaborate
- You will learn more about using Git/GitHub with R later on in this module! https://happygitwithr.com/

5. Control dependencies with `renv`

5.1 Package dependencies

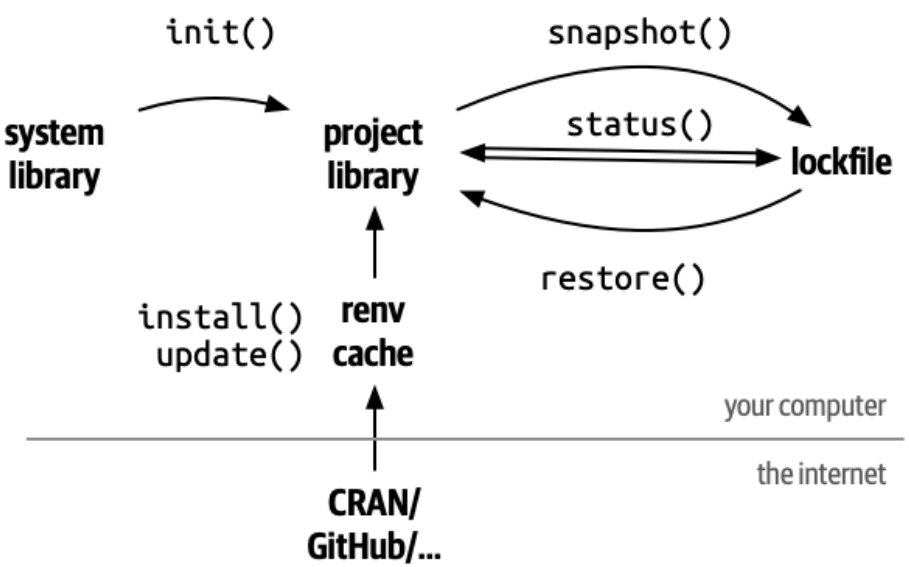
- Be careful with package dependencies! They may stop to maintain it, or make major changes!
- Always make a note of which functions you are calling from that package (if not too many!)
- Call external functions with `package::function()`
- Check if you can remove them! For example move from the `magrittr` pipe operator `%>%` to the native pipe `|>`.

Chaining %>% vs |>

- |> is the native pipe operator (available in R 4.1.0 or later)
- Some of you might be familiar with the %>% pipe provided by the `magrittr`/`tidyverse` package.
- For most cases |> and %>% behave identically.
- It's preferable to use |> since it's part of "base R", and it's simpler.
- For more info look at this blog post: https://www.tidyverse.org/blog/2023/04/base-vs-magrittr-pipe/

5.2 Control dependencies with `renv`

The `renv` package helps you create reproducible environments for your R projects.



rstudio.github.io/renv

5.2 Control dependencies with `renv`

- `renv` creates the following:
  - a folder `renv` containing the files `settings.json`, `activate.R` to setup renv, and a folder `library` that contains a project-specific library
  - a lockfile `renv.lock` containing a list of dependencies and their version, making it easy to share the project.
  - A project R profile `.Rprofile`.
- You only need to share 4 files: `renv.lock`, `.Rprofile`, `renv/settings.json` and `renv/activate.R` to recreate your project environment.

5.2 Control dependencies with `renv`

You can use `renv` in a project by setting it from your project (open the `.Rproj` file, go to *Environment*, and select *Use renv for this project*)

You can also run:

```
renv::init()
```

You update the lockfile by taking a `snapshot`:

```
renv::snapshot()
```

- Learn more about `renv` from rstudio.github.io/renv/

Engage with the other users!

Take home messages

- Write neat code full of comments
- Use version control
- Write reproducible and dynamic reports
- Your code must be accessible and easy to use (R package and/or Shiny app)
- Use `renv` to control dependencies
- Share your work (*not your assignments!*) and listen to all feedback