Reproducible workflows

Gerko Vink

Markup languages and reproducible programming in statistics

The blueprint of **R**

Layers in `R`

There are several 'layers' in `R`. Some layers you are allowed to fiddle around in, some are forbidden. In general there is the following distinction:

- The global environment.
- User environments
- Functions
- Packages
- Namespaces

Environments

The global environment can be seen as a olympic-size swimming pool. Everything you do has its place there.

If you'd like, you may create another, seperate environment to work in.

- A user environment would by default not have access to other environments

Functions

- If you create a function, it is positioned in the global environment.
- Everything that happens in a function, stays in a function. Unless you specifically tell the function to share the information with the global environment.
- See functions as a shampoo bottle in a swimming pool to which you add some water. If you'd like to see the color of the mixture, you'd have to squeeze the bottle for it to come out.

Packages

- Packages have their own space.
  - Everything needed to run the functions in a package is needly contained within its own space
  - See packages as seperate (mini) pools that are connected to the main pool (the global environment)

Loading packages

There are two ways to load a package in `R`

```
library(stats)
```

and

```
require(stats)
```

My advice is to always use `require()`, as it will produce a warning when a package is not found. In other words, it will not stop as function `library()` does.

Namespaces

- Namespaces. These are the deeper layers that feed new water to the surface of the mini pools.
    - Packages can have namespaces.
    - Functions within packages are executed within the package or namespace and have access to the global environment.
    - Objects in the global environment that match objects in the function's namespace are ignored when running functions from packages!

R in depth

Workspaces and why you should sometimes save them

A workspace contains all changes you made to environments, functions and namespaces.

A saved workspace contains everything at the time of the state wherein it was saved.

You do not need to run all the previous code again if you would like to continue working at a later time.

- You can save the workspace and continue exactly where you left.

Workspaces are compressed and require relatively little memory when stored. The compression is very efficient and beats reloading large datasets from raw text.

History and why it is useful

`R` by default saves (part of) the code history and `RStudio` expands this functionality greatly.

Most often it may be useful to look back at the code history for various reasons.

- There are multiple ways to access the code history.

    1. Use arrow up in the console. This allows you to go back in time, one codeline by one. Extremely useful to go back to previous lines for minor alterations to the code.
    2. Use the history tab in the environment pane. The complete project history can be found here and the history can be searched. This is particularly convenient when you know what code you are looking for.

RStudio

Why `RStudio`

- Aggregates all convenient information and procedures into one single place
- Allows you to work in projects
- Manages your code with highlighting
- Completes your code for you
- Gives extra functionality (Shiny, knitr, markdown, LaTeX)
- Allows for integration with version control routines, such as Git (next week).

Working in projects in `RStudio`

- Every project has its own history
- Every research project has its own project
- Every project can have its own folder, which also serves as a research archive
- Every project can have its own version control system
- R-studio projects can relate to Git (or other online) repositories

Archiving

Archiving

There may be tons of reasons for archiving

- The research seminar talks about the why (reproducibility) and about the official and legal requirements (GDPR - May 25, 2018)

In this course we discuss the following - how to obtain a reproducible workflow - how to have your workflow create an archive for you

It is important that your workflow manages the process from start to finish

1. For an analyst, a project starts the moment the data collection ends
2. A project is finished the moment it is published or finalized

In between there are many choices to be made (e.g. data throughput processes) and without documenting every step, the reproducibility of your work may be at stake.

Replicable vs Reproducible

Research results are replicable if there is sufficient information available for independent researchers to make the same findings using the same procedures. ***(Gary King, 1995)***

In computational sciences - such as ours - simply having the data and code means that the results are not only replicable, but fully reproducible.

We would like our results to be as fully reproducible as possible:

A. **Reproducibility is one of the pillars of science**

- It is the standard by which to judge scientific claims
- It helps the cumulative growth of knowledge - no duplication of effort

B. **Reproducibility may greatly benefit you**

- You'll develop better work habits
- Better teamwork - especially with new team members
- Changing or amending your work is much easier
- Higher research impact - more likely to be picked up and cited

For next week

Don't forget to apply for a GitHub educational discount. You'll need it from next week on.