Analysis of Next Generation Sequencing Data
Introduction to R Markdown and git

Luce Skrabanek

14 January, 2020

1 Introduction to R Markdown

1.1 What is R?

R is a free software environment for statistical computing and graphics (`www.r-project.org`). It can effectively analyze large-scale datasets, such as those resulting from high-throughput sequencing experiments. It promotes automated and reproducible analyses of scientific data, creates a wide spectrum of publication quality figures, and has an extensive library of add-on packages to facilitate many complex statistical analyses. Because it is free and ubiquitously available (it runs on Windows, Mac, and Linux computers), your investment in learning R will pay dividends for years to come.

1.2 What is RStudio?

While R is very powerful, it is essentially a command line program and is thus not the friendliest thing to use. Especially when learning R, a friendlier environment is helpful, and RStudio provides this, giving you things you expect in a modern interface like integrated file editing, syntax highlighting, code completion, smart indentation, tools to manage plots, browse files and directories, visualize object structures, etc.

From your computer, choose the RStudio application. This will start R under the hood for you.

2 Staying organized

A good practice is to create a new RStudio project for every project. This is RStudio's way of supporting and streamlining the common practice of keeping all the input data, R scripts, and other files associated with that analysis together. If you tell R about this directory, it will, by default, load and save files from it. We call this the working directory. You can browse files and directories from the Files tab of the lower right panel. This will create a .Rproj file in your project directory. Now, whenever you want to work on that analysis again, opening that .Rproj file will bring you back to exactly where you left off (the same working directory, the same files open, the same command history).

2.1 R Markdown

This is a way of combining not just R code, but also notes and comments about the code and project. It can be thought of as akin to a lab notebook, which you can use to capture scientific ideas and the associated analysis results and communicate these with colleagues.

To start a new R Markdown document, choose *File* ⇒ *New File* ⇒ *R Markdown*, or select R Markdown from the green cross icon. You will be asked to fill in some basic metadata (name of document and author) and what you want the default output format to be.

There are three parts to an R Markdown document:

1. a YAML header, surrounded by triple dashes, which is automatically generated for you when you open a new document

   ```
   ---
   title: "Some title"
   author: "Luce"
   output: html_document
   ---
   ```

2. the chunks of R code that will be run, surrounded by triple backticks (```)

   ```
   ``` {r optional_name, optional_options}
   R code here
   ```
   ```

3. text, with Markdown formatting.

To produce a complete report including code and text, click the Knit button. If you haven't already saved the file, it will ask you to do so now. Try this now, with the default content given to you on creation.

By default, an HTML file will be produced, with a preview either in the Preview panel, or externally (which preference can be set from the Options menu (the cog icon)). The HTML file can be sent to colleagues or your PI, and viewed in a browser. Other output options include PDF and Word, although you will need to install separate packages for that functionality. We will talk more about external packages later.

The sample R Markdown document contains chunks of R code. We will understand this later, but for now, realize that each R code chunk can be run separately. The output from each chunk can

include both the code and the results (use `echo=TRUE` as a chunk option) or just the results (use `echo=FALSE`). The output, including figures, may be shown inline (in the .Rmd file), but many people find this distracting, and prefer to see the output in the Console and Plot panels, which can be set from the Options menu (select Chunk Output in Console).

It is also possible to include mini-code chunks inline, surrounded by single backticks, and prefixed by the letter `r`. These will get evaluated, and get filled in, whenever the document is knitted.

What we care about for today is the text. The formatting of the text is accomplished with standard Markdown formatting. Examples include:

- Different level headings are prefixed by varying numbers of `#`.
- Italic or bold text is surrounded by one and two `*`, respectively,
- Bulleted list items are prefixed by `*`,
- Numbered list items are prefixed by numbers.

There are a number of guides to help you construct your R Markdown documents directly from within RStudio:

- an external R Markdown cheatsheet PDF, which lists all the most commonly used commands and syntax, accessed from *Help* ⇒ *Cheatsheets* ⇒ *R Markdown Cheat Sheet*,
- an external markdown reference guide PDF, accessed from *Help* ⇒ *Cheatsheets* ⇒ *R Markdown Reference Guide*, also found inline at *Help* ⇒ *Markdown Quick Reference*.

3 Introduction to Git

Git is an open-source version control system (VCS), that was started by Linus Torvalds. Version control systems not only store the current version of a file (or set of files), but previously saved versions as well (or rather, the modifications to that file). This saves you from having to keep multiple copies of the same file, and also allows you to see, or go back to, any older version of the file (especially handy if you are trying something that breaks your code.) With a VCS, you can revert a single file back to a previous state, or even the entire project. You can compare changes over time, and see when a particular line or code chunk was introduced.

Using a VCS also generally means that if you screw things up or lose files, you can easily recover, but just as importantly, from a rigor and reproducibility viewpoint, this also means that you can regenerate results from any previous iteration of your analysis pipeline. As your analysis methods evolve, you'll always be able to reproduce results in a systematic way.

While there are a few different VCSs out there, Git is the one preferred by most software developers as, among other reasons, it stores file changes efficiently and ensures file integrity better.

Git can operate locally, but its great strength is in allowing multiple users to collaborate on a project. The most widely used central Git repository is called GitHub.

While the main ideas behind Git are relatively simple, the tech-specific jargon can make it seem intimidating. A few definitions will help us get started:

1. A "repository" (usually abbreviated to "repo") is a location where all the files for a particular project are stored.
2. To "diff" a file is to look at the differences between the current version and the version stored in the repo.
3. To "commit" a file is to save the current version in the repo. When you commit a file (or set of files), you should include a "commit message" which is a sentence summarizing the changes. You can always look back at the git log, to see the history of all the changes that have been made.

Two useful resources, which will help to demystify a lot of the jargon, are:

1. `https://git-scm.com/book/en/v2`
   Documentation from the developers of git.
2. `https://marklodato.github.io/visual-git-guide/index-en.html`
   An explanation of git commands and how git works, using diagrams!

3.1 Some tips

1. Your repos should be scalable! What this generally means is that you should not put very large (e.g., NGS or imaging) datasets under git's control, and you probably shouldn't be versioning intermediate results.
2. Be careful about how you deal with passwords in config files, and any other stuff you don't want to leak.
3. It's a good idea to have many small repositories (at least one per project). Commit often!

Once you have Git installed on your local machine, then every time you open a new R project in RStudio, you have the option of associating it with a git repository. If this option is selected, you will have a new tab in your environment/history panel, which will give you a GUI to Git, allowing you to see how your files differ from the repository, select which files to save (commit), and even push to a remote repo (like GitHub).

3.2 Installing git

To get you started, the git tutorial at `https://happygitwithr.com/` is very thorough and provides an excellent walkthrough for installing git on your own machine (see Chapter 6).

