Getting Started

Jiaqi Li

1/24/2022

Contents

Follow steps 1-12 in `README.html` before beginning this document. All of your assignments will be written in R Markdown. This is an R Markdown document. There are three components of this document that we will pay attention to in this file:

1. the YAML header at the top of the document
2. the Markdown syntax used throughout the document
3. the executable code blocks interspersed throughout this document

Git and GitHub

Git, the software which you just set up in RStudio, is software which helps with version control and collaboration. GitHub is a public website that makes working with Git and creating joint projects (repositories or "repos") simpler. Our course has two repos for you to use–the portfolio and the final project. Your portfolio repo will not be shared with other students and contain only your work. Your final project repo will be shared among your group so all of you will be able to work on the same documents.

Within a repo, you can make your own Branch. A branch is a version of your repo which you can change without changing the original files (called the Main branch). Branches are local (on your own device) and

remote (on a server). In most cases, you'll only need to make branches when you're working in a group and don't want to disrupt other work; if no one else is editing the repo, working in Main is fine. In your Portfolio repo, you can work in Main. In your final project repo, *only* if you know you will be the only person editing a file (for example, working in class together and everyone watching your screen), you can most likely work in Main with no issues.

Editing a local branch won't change the remote branch until you explicitly tell the remote branch to update itself. By making a "commit" to your branch, you save its current local state and the changes you made. It's best practice to make many commits as you edit your documents, usually after a discrete section has been completed. To make the remote branch match your branch, you "Push" your commits. When you're in a shared repo, others will then be able to open your branch and see its current state. In other words, you synced your document with a version of the branch that everyone can see.

If they already have a version of the branch local to them, however, they need to make sure they are also synced to the remote version. To do so, they "Pull" the remote branch onto their local branch. This syncs their local branch with the current version of the remote branch. When Pulling, you must make sure to commit your changes beforehand. As a rule of thumb, PULL when you start working on a branch that someone else has worked on (often Main), then PULL before you PUSH to a shared branch to avoid as many merging errors as possible. For now, if you've chosen to make a branch, no one else should edit it.

Once you've finished what you wanted to do in your branch, you can make a Pull Request on GitHub. With a Pull Request, you are asking to pull your branch into the Main branch. A Pull Request, if approved, will merge your branch and Main branch, and you will no longer be able to make changes on your old branch. You can still make new branches or work in Main if you need to. Pull Requests are a great way to update others on what you've done, as they can look through your code and provide feedback before it gets merged into Main. That way, you can edit it before the branch gets closed. Once the Pull Request has gone through, you need to pull Main on your local device to see the updates.

> You do not need to make Pull Requests when working in Main, but you do need to Commit, Push, and Pull.

Overall, Git branching workflow can be summarized by: Branch -> Pull -> Edit -> Commit -> Pull -> Push -> Pull Request

Git workflow for working or viewing Main can be summarized as: Pull -> Edit -> Commit -> Pull -> Push

As you go through this document, you will be walked through making Commits, Pushes, and Pull Requests on the branch you created in `README` step 11.

RStudio setup

You can change your own RStudio set-up to your comfort. In particular, the colors of the text, background, highlight, can be customized by changing the theme of your RStudio. It's like shifting certain websites or applications from light mode to dark mode.

> Checkpoint 0: Open RStudio Preferences -> Appearance, then apply an Editor Theme or other customizations you like the look of!

YAML Header

R Markdown documents begin with a YAML header. This is the text that you see between the '—' (lines 1-6 of this document). This header includes a number of key-value pairs that provide metadata about the document (such as its title, who created it, when it was last updated, and the format in which it should be published). There are certain keys that appear by default in R Markdown files (e.g. title, author, date, output). Depending on the output format you specify, there may be additional options (such as whether your document should have a table of contents and how figures should be sized by default).

> Checkpoint 1: Add your name as the author of this document by editing the YAML header where it says to ADD YOUR NAME. Be sure to keep the quotation marks around your name.
>
> After successfully completing checkpoint 1, commit your code (see instructions in Readme step 13).

For this course, we will sometimes publish Markdown files as HTML documents (web documents) and sometimes as PDF documents. To compile this .Rmd file as one of these documents, we need to 'knit' the file. You can knit this document, but clicking the 'Knit' button in the taskbar above. This document is currently set to output as an HTML file. We know this because in the YAML header, the 'output' key is followed by the value 'html_document.'

> Checkpoint 2: Click 'Knit' in the taskbar above. A new window will open with this document formatted as an HTML document. Change the output from html_document to pdf_document. Click 'Knit' again. Notice how the output changes.
>
> After successfully completing checkpoint 2, commit your code again.

When we're making certain types of changes (e.g. table of contents, section numbering, themes, etc) we need to change the YAML formatting from:

```
output: pdf_document
```

To:

```
output:
pdf_document:
(more options here)
(sometimes more here too)
```

Making sure to include the indents.

> Checkpoint 3: Add a table of contents (toc) to your document through YAML setting `toc: true`. Knit your document to PDF, then to HTML. Notice how the same setting can be used for both document types. Optionally, add `toc_float: true` to your HTML.
>
> After successfully completing checkpoint 3, commit your code again.

If you would like to know more about customizing HTMLs and PDFs with the YAML headers, chapters 3.1 and 3.3 of Xie, Allaire, and Grolemund's free online book "R Markdown: The Definitive Guide" go into great detail with examples.

Bibliographies

To add citations to R Markdown files, we can either manually type them into the document, or we can add a BibTeX file (.bib) to our R project and reference the bibliographic metadata in that file to automatically insert citations into the document. BibTex files with full bilbiographic metadata can be exported from citation managers like Zotero and then added to the same directory as your R Markdown files. We highly encourage you to develop a practice of storing your citations in citation managers like Zotero. Here is a great tutorial video on getting started with Zotero, and here is another tutorial on exporting BibTex files.

Alternatively, you can use websites like Google Scholar to retrieve a citation in BibTeX formatting and then paste it into your BibTex file. If you search for an article or book in Google Scholar, underneath the link to the resource, you will find a link with the text "Cite." When we click on that Cite link, the resulting box

will include a link at the bottom with the text "Bibtex." If you click on that link, you will find a Bibtex entry for that resource that you can copy and paste into a BibTex file in your R project.

Open the `setting-up.bib` file by clicking it in the Files menu on the bottom right of your RStudio interface. There should be one reference in this file already. You'll notice that this entry has a bunch of metadata listed - the resource's title, author, and publisher, for example. The very first string of characters following the opening bracket at the top of a reference in the .bib file is the unique key for that resource. For example, Xie, Dervieux, and Riederer (2020) is the key for the R Markdown Cookbook. We can use that key to reference the resource in our R Markdown file.

There are three things that we need to do to add citations from our .bib file in our R Markdown file. First, we need to add `bibliography: FILE_NAME.bib` to our YAML header in the R Markdown file. When we knit the document, this communicates that R should look for that file in our project to find the bibliographic metadata needed to produce in-text citations and bibliographies. Second, we need to add a .csl file to the YAML header. This tells R which citation format (APA, MLA, Chicago, etc.) to use when configuring bibliographies. Finally, where we want to include an in-text citation in the body of the document, we need list the unique key for the citation, following the '@' symbol. For example to cite the R Markdown cookbook, we would type `@xie2020r`. With these three things in place, when you knit the document with the YAML header, there will be an in-text citation in the key's place in the format specified by the .csl. The full citation will also appear at the bottom of the document.

> Checkpoing 4: Update the YAML header to include a bibliography with `setting-up.bib`. Go to google scholar, search for "R Markdown: The Definitive Guide," copy the BibTeX citation and paste it into `setting-up.bib` as the second entry. Then, add an in-text citation to it below. Knit your document to make sure it shows up.
>
> After successfully completing checkpoint 4, commit your code again. Additionally, commit your setting-up.bib file.

Xie, Allaire, and Grolemund (2018)

Note that empty .bib's have already been provided in all of your assignment folders in this portfolio, so you need only copy and paste BibTex entries into those files. We've also already added references to those files and the appropriate .csl files to the assignment YAML headers.

If you would like to read more about citing in R Markdown, see this chapter.

Markdown

Markdown is a mark-up language - or a computer language that we can use to communicate to a digital system how we want text and images displayed when it is published. When you type text into software like Microsoft Word, you can change the size and font, add bullets, and insert images by clicking on buttons in the text editor, and the changes are immediately visible. Markdown is different. To format and style text in Markdown, we use certain designated symbols and syntax.

For instance, let's say we want to create a heading in our document. To do this we would add a '#' and a space in front of the text of our header like this: (note that Heading One will be the largest text, and subsequent headings will be increasingly smaller)

Heading One

Heading Two

Heading Three

Heading Four

Heading Five Heading Six

If we want to add bullets to an R Markdown file, we can add asterisks (*) and a space in front of the text we want bulletted:

- Bullet one
- Bullet two
- Bullet three

If we want to add a hyper-link to an R Markdown file, we can place the text we want hyper-linked in brackets '[]' and the link in parentheses '()' immediately following the bracketed text:

This is a link to our GitHub organization.

We can add images in a similar way, but include an ! beforehand.

Section 3 of this Cheatsheet provides a more comprehensive list of syntax rules for R Markdown.

> Checkpoint 5: Format the capitalized text immediately below this set of instructions as **bold** text. Then knit the document to check how the text gets displayed in your published document.
>
> After successfully completing checkpoint 5, commit your code again.

Make THIS TEXT bold.

Markdown and Line Spacing

Markdown has a couple quirks surrounding linespacing once the document is knit. If we include an empty line between new paragraphs, they will be separated in our knitted documents. If we include no new lines and no spaces at the end of the previous line they will merge together.
If we include no new lines **and** 2 spaces following the end of previous line, the two paragraphs will be on separate lines but have no blank lines between them.

Tabs at the start of the line do nothing.

In short, we need to put line breaks between every paragraph to make sure our document is readable when knit.

Code chunks

R Markdown documents can include chunks of code and their output. R code chunks are designated by starting a line with '`{r}' and ending a line with '`.' See below.

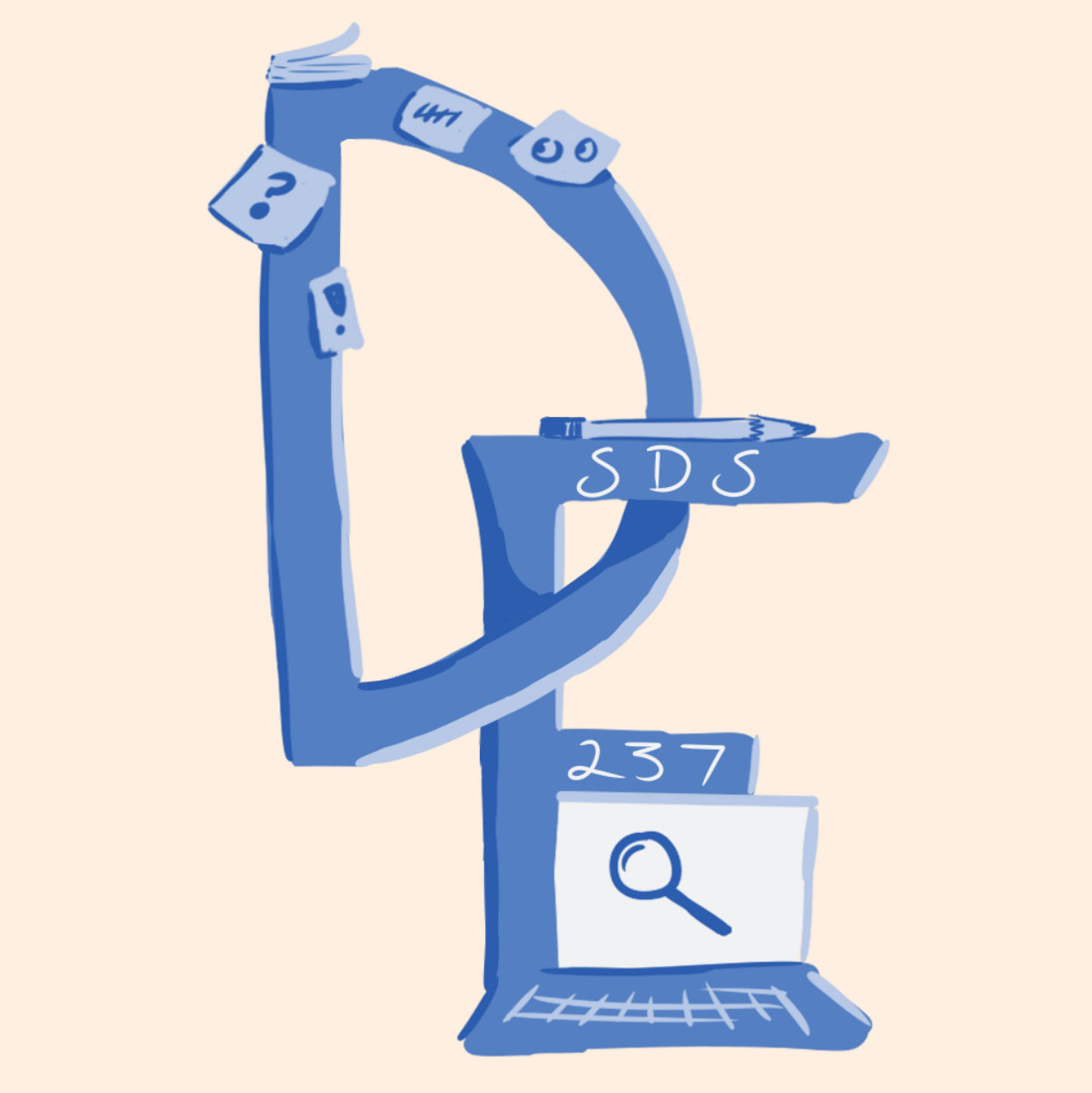


Figure 1: This is our course logo, designed by Zoe Scheffler Fall '21

```
sum <- 2 + 2

sum
```

```
## [1] 4
```

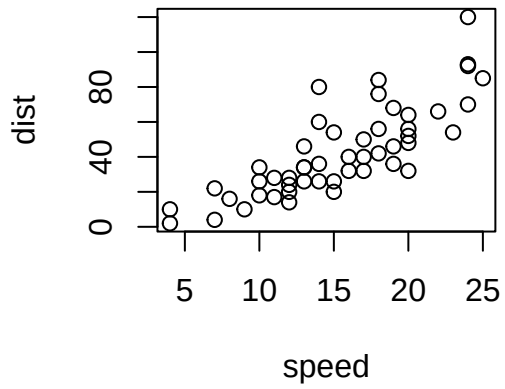
RStudio makes it easy for you to add a code chunk to a document by clicking the green button +C button in the taskbar and clicking R. By default when you publish a document with a code chunk, both the code block and the output of the code will be displayed in the document. If you add 'echo = FALSE' between the leading curly brackets of the code chunk (following 'r'), only the output of the code will be displayed. The code that created that output will be hidden.

```
## [1] 4
```

Section 5 of this Cheatsheet lists other options we can set for code chunks.

> Checkpoint 6: Add a code chunk below. Copy and paste the following code and place it in the code chunk: plot(cars) Referencing the linked cheatsheet, set the figure width to 3 and the figure height to 3. (You will need to separate these options with a comma.) 'Knit' the document to see how your plot renders in the document.
>
> After successfully completing checkpoint 6, commit your code again.

```
plot(cars)
```



Notice how the object `sum` showed up in our Environment tab on the top right. Any new object we make with the assignment symbol `<-` will show there. If you press the broom icon in the Environment tab, your list of objects will be cleared. It can be nice to clear your Environment if you have many objects that you aren't using. Since we have the code to remake `sum`, we don't have to worry if we clear it from the Environment.

RStudio comes with standard data, including the `cars` dataset we used for our plot. It's not in our environment because we haven't saved it with `<-`.

Checkpoint 7: Make a code chunk below, choose a name for the cars dataset, then save `cars` to it by using `<- cars`. Note how after doing so, it will show up in your environment, and if you click on it, you can see the dataset itself.

After successfully completing checkpoint 7, commit your code again.

DELETE THIS TEXT AND ADD CODE CHUNK HERE

Tidyverse

Tidyverse is a series of packages produced by RStudio designed to produce easy ways to handle and wrangle data. To use the commands provided by the Tidyverse, you need to install it, then load the package in.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Anytime we use a package for additional commands, you'll need to load it in with `library(package name)` and sometimes downlaod it first with `install.package("package name")`. The Tidyverse introduces the key character, `%>%`, called a pipe. A pipe strings commands together. It's analogous to an assembly line; person A picks up a box, changes it, hands the adjusted box it to person B. Person B changes it again, hands it off, etc. The people are the commands, the box is the object you're creating, and the act of handing the box between the people is what the pipe does.

Checkpoint 8: Replace "YOUR DATASET NAME" with the name of your cars dataset. Delete the #'s and run the code. What did the mutate function do? Hint: If you're unsure what a function does, typing ?Function into the Console at the bottom of the screen will pull up its Help page.

After successfully completing checkpoint 8, commit your code again.

```
# YOUR DATASET NAME <- YOUR DATASET NAME %>%
  # mutate(dist_meters = dist * 0.3048)

# YOUR DATASET NAME
```

Checkpoint 9: Make a code chunk below. With the Tidyverse command `filter`, find only the cars that took longer than 20 meters to stop. There should be 10. Hint: Use ?filter and scroll to the bottom to see examples if you need. Note that the examples all say the dataset as the first item, but because of piping, you don't need to.

After successfully completing checkpoint 9, commit your code again.

DELETE THIS TEXT AND ADD CODE CHUNK HERE

Coding going forwards

We anticipate that this R Markdown file contains largely review information for many of you. If you're less comfortable with this material or the majority R Markdown was new information, please let us know so we can best support you. The coding in this course should not be a barrier for your success.

> Checkpoint 10: If you would like additional support with R, replace this text.

> Stage, commit, and push your code following instructions on GitHub.

References

(This area is empty because the YAML bibliography header will paste the references here)

Xie, Yihui, Joseph J Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Chapman; Hall/CRC.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Chapman; Hall/CRC.