Reproducible Research in R

2020-11-16

Contents

Reproducible Research in R

- Level: beginner-intermediate
- Duration: 6 hours
- Student numbers: 25-30

Welcome to the Reproducible Research in R (RRR)[1] workshop. The main aim of this workshop is to set you on the right path of making your research more reproducible and shariable. Reproducible research means that future you and anyone else will be able to pick up your analysis and reproduce the same results, including figures and tables. Reproducible research also implies well-documented research, your code should be well commented and the reasons behind functions and methods should be explained thoroughly throughout the analysis. The communication aspect should not be an afterthought, it should be recorded with your analysis as you are going through it. Rmarkdown is a way of literate programing[2] that keeps code, words and sentences together. The ability to easily collaborate and share your analysis goes hand-in-hand with good record-keeping and reproducibility. We are going to repurpose the git version control tool and leverage the GitHub remote hosting provider for managing and sharing our work. Git + GitHub will provide a very powerful resource for global collaboration and exposure of your work. In this workshop, we are going to version control our work and push it to github, which can then be accessed by your collaborators and supervisors. Git + GitHub should become an integral part of your workflow.

The RRR course given by the Monash Bioinformatics Platform[3] for the Monash Data Fluency[4] initiative. Our teaching style is based on the style of The Carpentries[5].

Learning outcomes

Attendees will learn how to:

- write vanilla markdown, Rmarkdown and bookdown documents
- use `knitr`, `rmarkdown` and `bookdown` R packages to build various document types including PDF, HTML and DOCX
- create reproducible Rmarkdown documents leveraging `.Rproj` and `.RData`
- include in-line citation and full references list in to Rmarkdown document using `.bib` files
- create presentations from Rmarkdown documents that include R code
- work with `git` version control tool
- create reproducible and "backed up" analysis via remote repositories (e.g github)

Workshop description

This workshop is an introduction to writing and communicating research using Rmarkdown. Rmarkdown is an easy way to create documents that include your R code and its output, such figure and tables. Rmarkdown

[1]https://github.com/MonashDataFluency/r-rep-res
[2]http://www.literateprogramming.com/knuthweb.pdf
[3]https://www.monash.edu/researchinfrastructure/bioinformatics
[4]https://monashdatafluency.github.io/
[5]https://carpentries.org/

is a single document that can be "knitted" and shared as various document types such as PDF and HTML. Rmarkdown supports scientific writing such as use of citations and figure cross-referencing. Rmarkdown can also be used to create presentations that include your R code and its output. We will also cover bookdown, which is an extension to Rmarkdown that allows the creation of larger documents, such as books with multiple chapters.

In this workshop, we will also cover git version control tool[6] which can help with organising and "checkpointing" Rmarkdown documents, associated R code and data. Git is not a backup system, but it does allow one to retrieve older versions of your work. Git, together with remote repositories like GitHub[7] can provide a centralised location for your research. Together Rmarkdown, git and github can enable reproducible research that is visible and accessible to the greater public, including supervisors and management.

Prerequisite

This is an introductory level workshop, however some prior exposure to R and familiarity with RStudio is assumed. It is highly recommended that you read this article in full[8], if you have to prioritise, read at least these section (1,2,6,10). Additionally, it is highly recommended that you create an account at GitHub[9] and remember your password.

Keywords

- R
- Rmarkdown
- communication
- reproducibility
- git and github

Schedule

10:00-10:30am (30 minutes) Welcome and warm up

```
\(   )/
```

10:30-12:00pm (1.5 hours) Rmarkdown

- Introduction (30 minutes)
- Vanilla markdown (30 minutes)
- Rmarkdown (30 minutes)

12:00-1:00pm (1 hour) lunch

1:00-3:00pm (2 hours) More Rmarkdown

- Git and GitHub markdown (40 minutes)
- More Rmarkdown (40 minutes)
- YAML header (40 minutes)

3:00-3:15pm (15 minutes) Tea break

3:15-4:45pm (1.5 hours) Even more Rmarkdown

- Bibliographies (30 minutes)
- Bookdown (30 minutes)

[6]https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control
[7]https://github.com
[8]https://peerj.com/preprints/3159/
[9]htts://github.com

- Miscellaneous (30 minutes)

4:45-5:00pm (15 minutes) Warm down

Authors and copyright

This course is developed for the Monash Bioinformatics Platform by:

- Paul Harrison[10]
- Adele Barugahare[11]
- Kirill Tsyganov[12]

[10]mailto:paul.harrison@monash.edu
[11]mailto:Adele.Barugahare@monash.edu
[12]mailto:kirill.tsyganov@monash.edu
[13]http://creativecommons.org/licenses/by/4.0/

Part I: Rmarkdown from scratch

In the first part of the book we will begging with blank text file and finish with a sophisticated HTML webpage ready to be shared with the world. We will learn three main components of the Rmarkdown document, markdown syntax, YAML header and the code chunks, together with the rmarkdown R package[14] we are table to convert Rmarkdown files into HTML, PDF and Word documents effortlessly.

[14] https://github.com/rstudio/rmarkdown

Chapter 1

Markdown syntax

We will begging exploring concise markdown syntax in the first section of the book. You will be amazed how with just few additional text markups and the smart functions from rmarkdown R package[1] we can create a world of different document types.

1.1 Introduction

Let's begin with install `rmarkdown` package using the following command

```
install.packages("rmarkdown")
```

Once we have the package installed, let's open a plain text file by either executing the following command

```
file.edit("learning.Rmd")
```

or via help menu at the top left of RStudio

```
File
   New File
         Text File
```

and then saving that file as `learning_rmd.Rmd` by pressing `ctrl+s`. Note that `.Rmd` extension is important it triggers RStudio to understand that we are working with Rmarkdown file, which results in new button `Knit` appearing at the top of the text editor window

And let's write our first bit of text and press `Knit` button at the top of the console pane or use keyboard shortbut `ctrl+shift+k`

```
I would like to learn Rmarkdown.
```

— Excellent work there !

1.2 RStudio tweaks

Let's do a couple quick RStduio tweaks to help us throughout the day. Use menu bar at the top to navigate to `Global Options` which should bring a new dialog up.

```
Tools
   Global Options
```

[1]https://github.com/rstudio/rmarkdown

```
R General -> (untick) Restore .RData into workspace at startup
R General -> Save workspace to .RData on exit: Never
R Markdown -> Show output preview in: Viewer Pane
```

1.3 Headers

Let's start out first journey into marking text up (or down?)

Add a `#` hash symbol in front of the short sentence that we wrote and press `Knit` again

```
# I would like to learn Rmarkdown
```

Now play with different header levels and don't forget to press `Knit`

```
#### I

### would like

## to learn

# Rmarkdown
```

— Excellent, you are getting a hang of it!

1.4 Bullet points

Let's now add a bit more content to our Rmarkdown document. First of all remove all of the previous text and type out this one and press `Knit` or `ctrl+shift+k`

```
# Introduction

> outcomes:

- learn how to create and share Rmarkdown document
- learn various document type, e.g html, pdf and docx
- ~~learn to fly the rocket!~~


The bottom line is - I would like to learn Rmarkdown!
<br>
Like **seriously** become **good** at it !
```

— This is looking good!

1.5 Task list

You should understand by now that by using different number of `#` symbols we are creating different section in our document, so let's create another section and `Knit`

```
## Task list for today

- [X] Get aspired to learn Rmarkdown
- [ ] Get to know three core components of the Rmarkdown document (header, body, R chunks)
- [ ] Practice, practice !
```

1.6 Code and Math blocks

Both mathematical equations and code chunks can be embedded inline or standalone. Let's first add new, code section to our document and include a short code chunk

1.6.1 Code chunk

~~~
## Code

You can use single backtick on each side of the code for inline or triple backticks for standalone code

```
#!/bin/bash

echo "Why am I doing BASH in this course?"

Rscript --vanilla run_away.R
```
~~~

One useful addition to the code block syntax (triple backticks) is ability to specify language for highlighting purposes, which realy helps with readability. The code about is `BASH` let's add that to our code block ` ```BASH ` and `Knit` again.

It doesn't matter if you use lower case `bash` or upper case `BASH` word, but for other languages it may matter, you should double check.

We will get to this a bit later in the book, but a little spoiler, if you don't mark your code chunk with any language name, `knitr` won't be able to recognise that chunk as code and your Rmarkdown document will miss out on some useful features e.g `code_folding`, but let's talk more about shortly

1.6.2 Math chunk

Let's add another section and include a one math equation into our document.

```
## Math
This is a covariance equation, random variable $X$ co-variace with random variable $Y$, where $\\bar{X}

$$
cov(X,Y) = \frac{\\sum(X_{i}-\\bar{X})(Y_{i}-\\bar{Y})}{N-1}
$$
```

More on LaTex here[2]

1.7 Images

We add images the same way we add any other hyperlink using the following syntax, except we also add `!` symbol at the front to emphasize that this is an image and not a text hyperlink

`[text](link)`

where "link" is a path to the file, can be on the internet i.e URL or local path on your system

Let's add Rmarkdown Hex sticker image to our document under another section "Images" and press `Knit`

[2]https://en.wikibooks.org/wiki/LaTeX/Mathematics

```
## Images

![hex_image](https://bookdown.org/yihui/rmarkdown/images/hex-rmarkdown.png)
```

Everything had worked well and we can see the image, except the image is pretty much the only thing we can see now. Let's adjust image size with the following addition. We are resizing the image to 50 % of the original size.

```
## Images

![](https://bookdown.org/yihui/rmarkdown/images/hex-rmarkdown.png){width=50%}
```

1.8 Final look

Just in case you've got lost, this is how our first section should look like

~~~
# Introduction

> outcome:

- learn how to create and share Rmarkdown document
- learn various document type, e.g html, pdf and docx
- ~~learn to fly the rocket!~~


The bottom line is - I would like to learn Rmarkdown!
<br>
Like **seriously** become **good** at it !

## Task list for today

- [X] Get aspired to learn Rmarkdown
- [ ] Get to know three core components of the Rmarkdown document (header, body, R chunks)
- [ ] Practice, practice !

## Code

You can use single backtick on each side of the code for inline code or triple backticks for standalone

```bash
#!/bin/bash

echo "Why am I doing BASH in this course?"

Rscript --vanilla run_away.R
```

## Math

This is a covariance equation, random variable $X$ co-variace with random variable $Y$, where $\bar{X} 

$$cov(X,Y) = \frac{\sum(X_{i}-\bar{X})(Y_{i}-\bar{Y})}{N-1}$$

[for more LaxTex syntax here](https://en.wikibooks.org/wiki/LaTeX/Mathematics)
~~~

```
## Images

![](https://bookdown.org/yihui/rmarkdown/images/hex-rmarkdown.png){width=50%}
```

Chapter 2

YAML header

YAML header is a short blob of text, specially formatted with `key: value` pairs tags, that seats at the top of our Rmarkdown document. The header not only dictates the final file format, but a style and feel for our final document. Later in the book we will learn that YAML header can seat in a stand alone file, such as `_site.yml` or `__output.yml`.

2.1 Introduction

Let's begging by adding some personality to our document via YAML configuration. For now we are going to be embedding YAML header into our document right at the top. The header needs to be enclosed in triple dashes on either side as following

```
---
# header here
---
```

Inside YAML header we can use `#` symbol to indicate a comment, those won't have any effect on the document

Typical structure of the YAML header is key, value pairs separated by a colon e.g

```
---
key: value
---
```

YAML is hierarchical and can have nested structure, key value pairs e.g

```
---
key1:
  value1:
    key1.1: value1.1
    key1.2: value1.2
---
```

Spaces (indentation) is very important to YAML header, so be sure to indent correct amount. Below are two examples of the valid YAML headers and one example of the invalid YAML header

2.1.1 Example 1 (Correct YAML)

This is a valid yaml header. In this example we have a key `title` with scalar value `Learning Rmarkdown`

```
---
title: "Learning Rmarkdown"
---
```

2.1.2 Example 2 (Correct YAML)

This is also valid YAML header. Here we have `rmd_files` key with list of values. We can specify a list value in two way

```
---
rmd_files: ["00-index.Rmd", "01-introduction.Rmd"]
---
```

Alternatively

```
---
rmd_files:
  - 00-index.Rmd
  - 01-introduction.Rmd
---
```

2.1.3 Example 3 (Incorrect YAML)

This on the other hand is incorrect YAML header because we are not specify a list values correctly

```
---
rmd_files:
  00-index.Rmd
  01-introduction.Rmd
---
```

— Alright, Let's try it out!

2.2 Title, date and author keys

Let's add our first YAML header to the top of our document. Let's include the title of the document, author name i.e yourself and date. **Note** a nice trick of including inline `R` code inside the YAML header.

Press `Knit` button to render the document

```
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-16"
---
```

2.3 Output key

Output is one of the central keys in Rmarkdown ecosystem, arguably. The final results directly depends on the value you are going to use for that key.

Rmarkdown package can output several document types, each has various ke, value pairs a.k.a options of flags, refere to the documentation page for more information prefixing function name a `?`. Here are most common document types:

- `rmarkdown::html_document`
- `rmarkdown::ioslides_presentation`
- `rmarkdown::word_document`

Let's try out `html_document` output type by including it into our YAML header and press `Knit`

```
output: html_document
```

If you don't see much difference after including `html_document` this is because it is the default output. Now, however, we have means to further configurate our *html* document.

2.4 Configuring output document

We will learn a bit later how to get information about all possible key, value pairs for any given document type, for now let's just explores these few. Let's add `toc` and `theme` keys to our YAML header and press `Knit`

```
output:
  html_document:
    toc: true
    theme: readable
```

— This looks great!

2.5 Final look

Just in case you've got lost, this is how our first section should look like

```
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-16"
output:
  html_document:
    toc: true
    theme: readable
---
```

Chapter 3

The code chunks

We have already seen that we can highlight any code with triple backticks and the language keyword in the first section of the book. The Knitr R package[1] provides a mechanism for including **live** code into the document instead. This means that apart from code, but can see and interact with the results of such code. This is extremely useful for research communication and reproducibility.

For this part of the book, we will need to install a few additional packages. Copy and paste install packages code into your RStudio console

```
install.packages(c("dplyr", "DT", "ggplot2"))
```

3.1 Introduction

Each code block is referred to as "code chunk". You can have as many code chunks as you like. The code chunk is essentially what we have already been using, i.e triple backticks on either side of the code blob, but now add curly brackets at the end of the first triplet e.g

~~~
```{r}

```
~~~

Look for green "insert" button at the top of the text editor pane, in order to insert R chunk. Alternatively use keyboard shortcut `alt+shift+i`

Each chunk is highly configurable via chunk options, using key value pairs once again. The very first chunk option is `engine` i.e which language is included in the code chunk. Yes Rmarkdown is capable of doing many different languages. Use drop down arrow on the green "insert" button to explore some of the most common languages.

N.B: Name of the engine i.e the language name, always position first in the R chunk, usually key "engine" is omitted.

Let's include our first bit of R code. We are going to load a library `magrittr`, to get a pipe operator (it isn't overly important to understand the pipe operator for this course), then we are going to call base function `avaliable.packages()` and store results in the variable `avail_packages` and `Knit` our document once again. Note that I'm also including new section header.

~~~
# R code

```{r}
~~~

[1]https://cran.r-project.org/web/packages/knitr/

```
library(magrittr)

avail_packages <- available.packages(contriburl = contrib.url("https://mirror.aarnet.edu.au/pub/CRAN/"))

#head(avail_packages)

avail_packages %>% head
```
```

— It's computing !

3.2 Chunk name

It is very useful to name your code chunks, they can server multiple purposes such as cross-referencing and debugging. Names are optional, but if you are including them, they must be in a second position.

General layout of any chunk is

```
```{r chunk_name, options}
```
```

Let's add a `get_data` name to our code chunk. We also going to wrangle our data a little bit to get most relevant information for us and press `Knit`

```
```{r get_data}

library(magrittr)
library(dplyr)

avail_packages <- available.packages(contriburl = contrib.url("https://mirror.aarnet.edu.au/pub/CRAN/"))
                    as.data.frame() %>%
                    as_tibble() %>%
                    rowwise() %>%
                    mutate(Depend = length(unlist(strsplit(Imports, split = ",")))) %>%
                    ungroup() %>%
                    select(Package, Version, Depend, Imports, License)
```
```

— How is it looking?

3.3 Chunk options

For comprehensive overview of code chunks that `knitr` refer to definitive guide[2] from the author of `Knitr` and `Rmarkdown` R packages.

Here is a short list of the most common that tend to be used by me.

[2]https://yihui.name/knitr/options/
```

| name | value | type | description |
|---|---|---|---|
| child | NULL | code_evaluation | A character vector of filenames. Knitr will knit the files and place them into th |
| engine | 'R' | code_evaluation | Knitr will evaluate the chunk in the named language, e.g. engine = 'python'. R |
| eval | TRUE | code_evaluation | If FALSE, knitr will not run the code in the code chunk. |
| include | TRUE | code_evaluation | If FALSE, knitr will run the chunk but not include the chunk in the final docun |
| fig.align | 'default' | plots | How to align graphics in the final document. One of 'left', 'right', or 'center'. |
| fig.cap | NULL | plots | A character string to be used as a figure caption in LaTex. |
| fig.height | 7 | plots | The height to use in R for plots created by the chunk (in inches). |
| fig.width | 7 | plots | The width to use in R for plots created by the chunk (in inches). |
| echo | TRUE | results | If FALSE, knitr will not display the code in the code chunk above it's results in |
| results | 'markup' | results | If 'hide', knitr will not display the code's results in the final document. If 'hold' |
| message | TRUE | results | If FALSE, knitr will not display any messages generated by the code. |
| warning | TRUE | results | If FALSE, knitr will not display any warning messages generated by the code. |

— There are way too many of them, good luck !

## 3.4 Message and warning options

It is great that we have live code in our document, but results seems to be popluted with additional text. Those are "messages" from the package loading and some could be additional warning about casting values to different type. It depends, but generally suppressing message really helps with document apperance. I also tend to suppresss warnings, but this can cause issue in downstread analysis, be sure you understand your warnings before suppressing them.

Let's add a couple of options to our code chunk and press `Knit`

~~~
```{r get_data, message = FALSE, warning = FALSE}
~~~

— Much better, yeah?

## 3.5 More code chunks and data wragnling

Let's add a couple more chunks and wrangle our data a little more.

*It is okay if you don't understand some of R code in this example. The main point of these R chunks is to illustrate what can be done*

~~~
## Getting and wrangling the data

```{r get_data}
make_url <- function(package) {
  paste0('<a href="https://cran.r-project.org/web/packages/', package, '">', package, '</a>')
}

library(magrittr)
library(dplyr)

avail_packages <- available.packages(contriburl = contrib.url("https://mirror.aarnet.edu.au/pub/CRAN/")
                    as.data.frame() %>%
                    as_tibble() %>%
                    rowwise() %>%
                    mutate(Depend = length(unlist(strsplit(Imports, split = ",")))) %>%
                    ungroup() %>%
                    select(Package, Version, Depend, Imports, License) %>%
                    mutate(Package = make_url(Package),
~~~

```
                           Depend = ifelse(is.na(Imports), -1, Depend))
```

## Displaying table

```{r make_table, message=FALSE, warning=FALSE}
library(DT)
avail_packages %>% datatable(escape = FALSE)
```

## Plot the data

```{r plot}
library(ggplot2)
avail_packages %>% ggplot(aes(Depend)) + geom_bar()
```
```

— Amazing !

3.6 Session info

Let's include session info

```
sessionInfo()
```

3.7 Setup chunk

While it is great to be able to configure each chunk individually, but sometimes setting up a "global" chunk can help with making document chunks more manageable and easier to read.

Let's include a "setup" chunk (name of the chunk can be anything) and set some of the options globally for the entire document. Note that we can also move all of our libraries into that chunk. This may or may not be a good idea, once again all depends what you need to show the person you are communicate this too. Don't forget to press `Knit`

Note that `include` option here set to `FALSE` meaning that chunk won't be included at all into the final document, as oppose to `eval = FALSE` where chunk is included but not evaluated.

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE,
                      message = FALSE,
                      warning = FALSE)

library(magrittr)
library(dplyr)
library(DT)
library(ggplot2)
```
````

— How about that?

3.8 Final look

Just in case you've got lost, this is how our second section should look like
````

~~~
# R code

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE,
                      message = FALSE,
                      warning = FALSE)

library(magrittr)
library(dplyr)
library(DT)
library(ggplot2)
```

## Getting and wrangling the data

```{r get_data}

make_url <- function(package) {
  paste0('<a href="https://cran.r-project.org/web/packages/', package, '">', package, '</a>')
}

library(magrittr)
library(dplyr)

avail_packages <- available.packages(contriburl = contrib.url("https://mirror.aarnet.edu.au/pub/CRAN/")
                    as.data.frame() %>%
                    as_tibble() %>%
                    rowwise() %>%
                    mutate(Depend = length(unlist(strsplit(Imports, split = ",")))) %>%
                    ungroup() %>%
                    select(Package, Version, Depend, Imports, License) %>%
                    mutate(Package = make_url(Package),
                           Depend = ifelse(is.na(Imports), -1, Depend))
```

## Displaying table

```{r make_table, message=FALSE, warning=FALSE}
library(DT)
avail_packages %>% datatable(escape = FALSE)
```

## Plot the data

```{r plot}
library(ggplot2)
avail_packages %>% ggplot(aes(Depend)) + geom_bar()
```
~~~

# Chapter 4

# Extras

So far you have learned the core set of features and knowledge that sets you up for recording your work in reproducible manner. We will show next a couple of useful tweaks that you can do to your document that might help in organising and presenting your work. In addition we will practice generating other document types.

## 4.1 Code folding

We haven't talked a lot about `eval` and `echo` chunk options. You can both of them in combination to present relevant bit of information to the audience. For example some might be interested in the code, others in the results or maybe both. We will let you play with various combination of those options at your own time. Instead we are going to add another tags (key value pair) to our YAML header, `code_folding`. This belong to `html_document` and should be indented appropriately.

Let's set it to `hide` and added to our header and press `Knit`

```
code_folding: "hide"
```

overall it should have the following structure

```
output:
  html_document:
    code_folding: "hide"
```

— This looks neat !

## 4.2 Presentation slides (ioslides)

Now that we have done all of the hard work of getting, wrangling and plotting the data, let's present it to others.

1. Make a copy of your original document

*This isn't strictly needed, but most likely you will want to adjust things a little for your presentation, so let's make a copy by running the following command*

```
file.copy("learning.Rmd", "presenting.Rmd")
```

Alternatively use "Files" tab in one of your panes, under "More" dropdown menu select "copy" and type in `presenting.Rmd` name

2. Change output type in YAML header

Let's change our output type to `ioslides_presentation` and add a couple of useful tags

```
output:
  ioslides_presentation:
    widescreen: true
    smaller: true
```

3. Press `Knit` to render

— Ripper !

## 4.3 ioslides features

Keyboard shortcuts:

- `f`: enable fullscreen mode
- `w`: toggle widescreen mode
- `o`: enable overview mode
- `p`: show presenter notes
- `Esc` key: to exits

Syntax to add presentor notes to the slide

```
<div class="notes">
This is my *note*.

- It can contain markdown
- like this list
</div>
```

— :)

## 4.4 Tabs

Alternative to section can be tabs, in fact two can happily co-exist. To create tabs we need to use `{.tabset}` addition to the header e.g

```
# Header 1 {.tabset}
```

This will make all subsection of the next level to become tabs. That is level 2 headers will become tabs, but level three will remain inside the corresponding tab.

Let's switch to our original document, `learning.Rmd` and try this out by adding `{.tabset}` to two of our first level header, `Introduction` and `R code` and press `Knit`

```
# Introduction {.tabset}

...

# R code {.tabset}

...
```

This look ok, but not the effect that I desired, instead what I wanted is to have two tabs, one for each of the main section. In order to get that results we will need to create new first level header and adjust levels of all other headers to the next deepest

In additional I would set `toc` to false, `toc: false`, since table of content doesn't have a purpose anymore

### 4.4.1 Final look

~~~
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-16"
output:
  html_document:
    toc: false
    theme: readable
    code_folding: "hide"
---

# TL;DR {.tabset}

[place holder]()

## Introduction

> outcome:

- learn how to create and share Rmarkdown document
- learn various document type, e.g html, pdf and docx
- ~~learn to fly the rocket!~~


The bottom line is - I would like to learn Rmarkdown!
<br>
Like **seriously** become **good** at it !

### Task list for today

- [X] Get aspired to learn Rmarkdown
- [ ] Get to know three core components of the Rmarkdown document (header, body, R chunks)
- [ ] Practice, practice !

### Code

You can use single backtick on each side of the code for inline code or triple backticks for standalone

```bash
#!/bin/bash

echo "Why am I doing BASH in this course?"

Rscript --vanilla run_away.R
```

### Math

This is a covariance equation, random variable $X$ co-variace with random variable $Y$, where $\bar{X} =

$$cov(X,Y) = \frac{\sum(X_{i}-\bar{X})(Y_{i}-\bar{Y})}{N-1}$$
[for more LaxTex syntax here](https://en.wikibooks.org/wiki/LaTeX/Mathematics)
~~~

~~~
### Images

![](https://bookdown.org/yihui/rmarkdown/images/hex-rmarkdown.png){width=50%}

## R code

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE,
                      message = FALSE,
                      warning = FALSE)

library(magrittr)
library(dplyr)
library(DT)
library(ggplot2)
```

### Getting and wrangling the data

```{r get_data}

make_url <- function(package) {
  paste0('<a href="https://cran.r-project.org/web/packages/', package, '">', package, '</a>')
}

library(magrittr)
library(dplyr)

avail_packages <- available.packages(contriburl = contrib.url("https://mirror.aarnet.edu.au/pub/CRAN/")
                    as.data.frame() %>%
                    as.data.frame() %>%
                    as_tibble() %>%
                    rowwise() %>%
                    mutate(Depend = length(unlist(strsplit(Imports, split = ",")))) %>%
                    ungroup() %>%
                    select(Package, Version, Depend, Imports, License) %>%
                    mutate(Package = make_url(Package),
                           Depend = ifelse(is.na(Imports), -1, Depend))
```

### Displaying table

```{r make_table, message=FALSE, warning=FALSE}
library(DT)
avail_packages %>% datatable(escape = FALSE)
```

### Plot the data

```{r plot}
library(ggplot2)
avail_packages %>% ggplot(aes(Depend)) + geom_bar()
```
~~~

# Part II: Git, GitHub and RStudio Rproj file

Git, GitHub and RStudio projects are three independent things; a tool, a place and an anchor, respectively. It is an onerous task to gain appreciation for all three. One great resources for that is Happy Git and GitHub for the useR[1]. For now we would like you to know that all three together can help you with keeping on top of your project, minimise content and data loss and will make your work naturally more collaborative. We will just tip our toes into that world.

[1]https://happygitwithr.com/

# Chapter 5

# Git (part one)

In the part one section of the book we will show you how to work with git on your local compute, without any regard to the external (remote) place such as GitHub. In part two we will show you how to work with external (remote) repositories hosted at GitHub

## 5.1 Introduction

Git[1] is a version control tool, one of many tools, but it is very popular. Git was designed for **tracking versions** of software development, but it has been re-purpose for doing general tracking of text and data i.e scientific research.

Below is an illustration of the differences between do-it-yourself (DIY) version control system, and git version control. DIY version control systems are great with two caveats:

- no one else will understand it
- the future you will forget the awesome schema that you have invented

## 5.2 Git init

Let's initiate new git repository, which is a lingo for marking our current folder **special**. From now on everything in that folder will be tracked by git. **BUT** we still have to actively save (commit) changes.

```
Tools
    Version Control
               Project setup
```

```
Version control system: git
```

Alternatively execute the following code in R console and re-launch RStudio project

```
system("git init")
```

Once that's done, you should see new `Git` tab appearing in "Environment, History …" pane

— That was unusual …

## 5.3 Git commit

Let's click on that new "Git" tab and select a couple of file and then click "commit" button.

[1] https://git-scm.com/doc

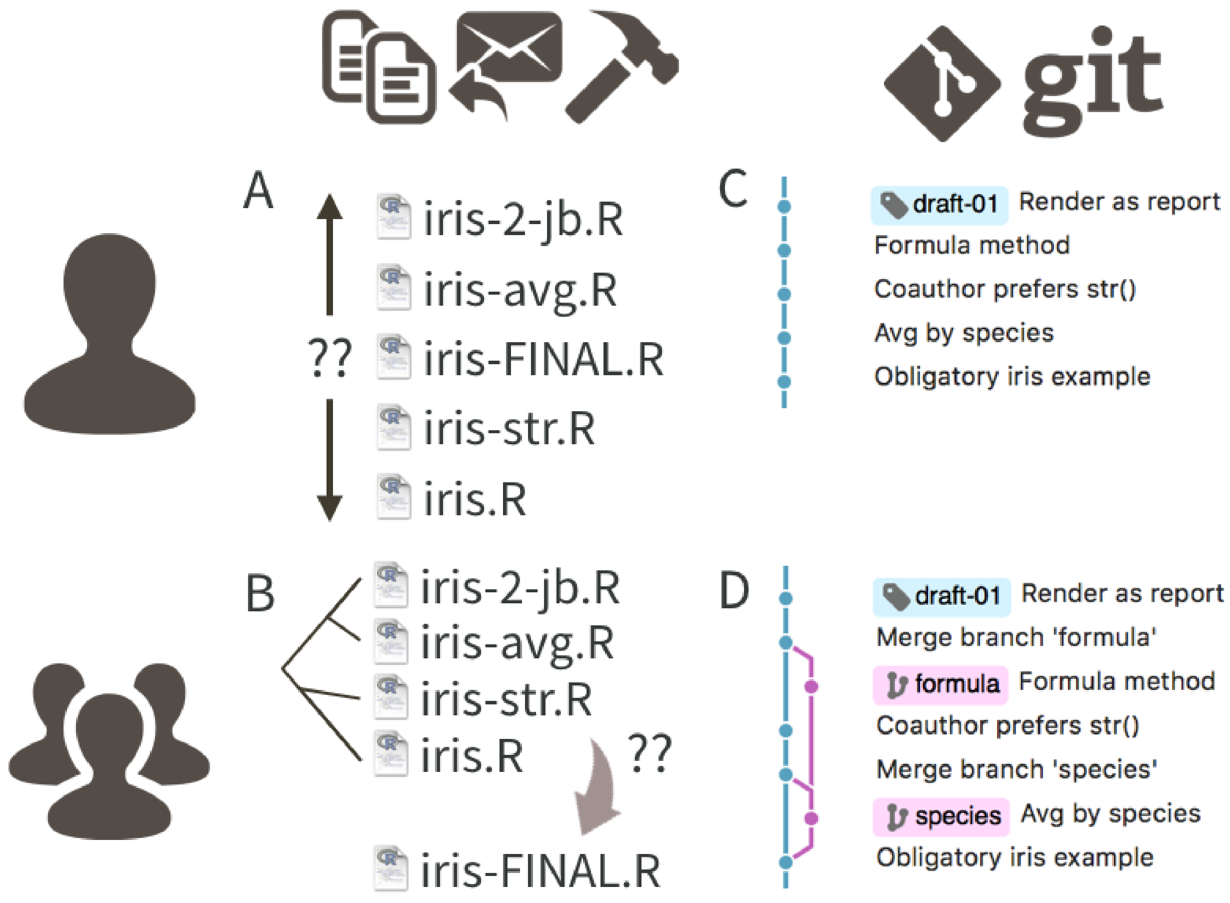
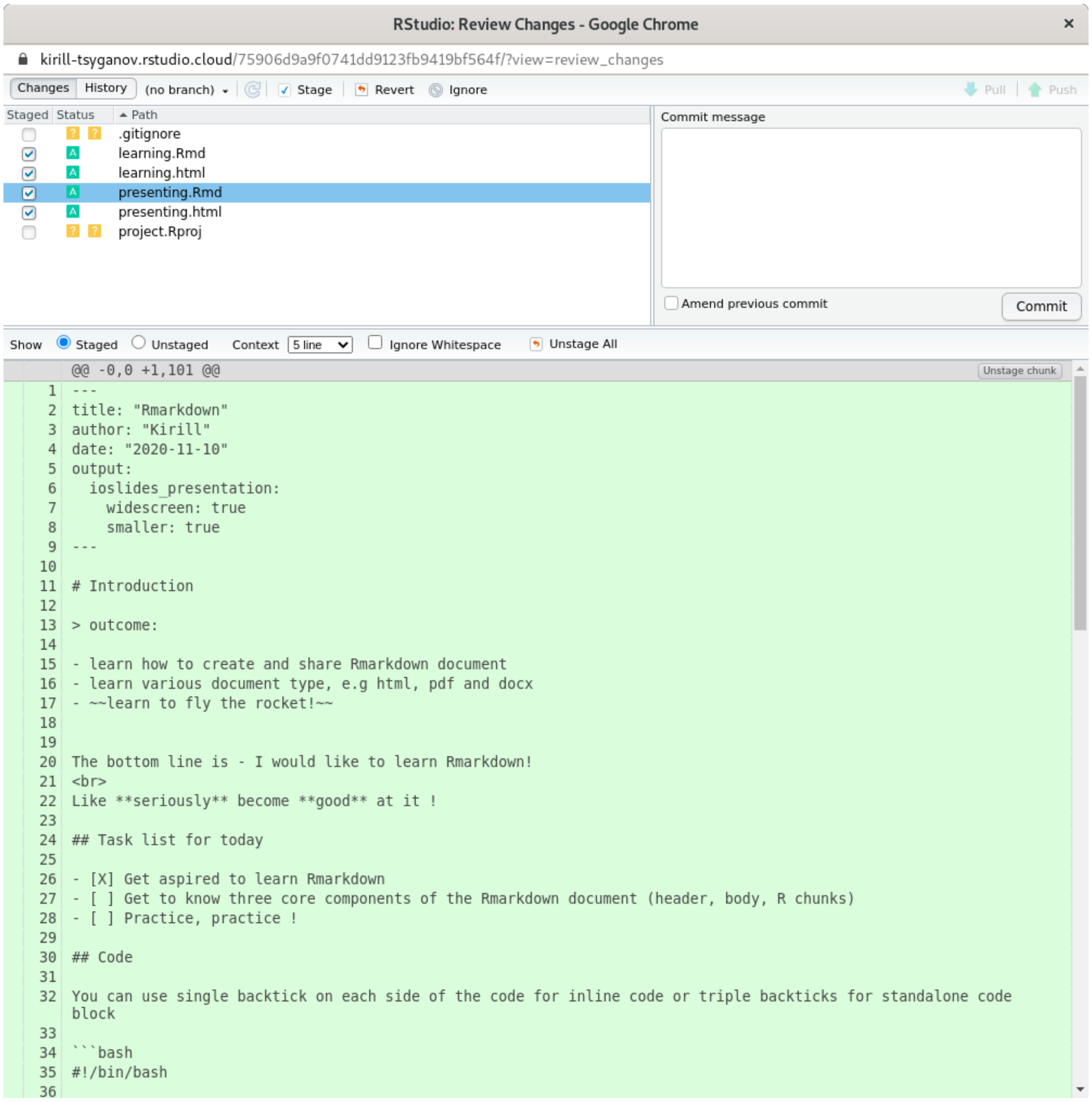


Figure 5.1: This is an example of git version control vs DIY versioning via filesystem

1. Select these files:

- `learning.Rmd`
- `learning.html`
- `presenting.Rmd`
- `presenting.html`

A new dialog should pop up, you should see this window 5.2



| Staged | Status | Path |
|---|---|---|
| ☐ | ? ? | .gitignore |
| ✓ | A | learning.Rmd |
| ✓ | A | learning.html |
| ✓ | A | presenting.Rmd |
| ✓ | A | presenting.html |
| ☐ | ? ? | project.Rproj |

~~~
@@ -0,0 +1,101 @@
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-10"
output:
  ioslides_presentation:
    widescreen: true
    smaller: true
---

# Introduction

> outcome:

- learn how to create and share Rmarkdown document
- learn various document type, e.g html, pdf and docx
- ~~learn to fly the rocket!~~


The bottom line is - I would like to learn Rmarkdown!
<br>
Like **seriously** become **good** at it !

## Task list for today

- [X] Get aspired to learn Rmarkdown
- [ ] Get to know three core components of the Rmarkdown document (header, body, R chunks)
- [ ] Practice, practice !

## Code

You can use single backtick on each side of the code for inline code or triple backticks for standalone code block

```bash
#!/bin/bash

~~~

Figure 5.2: RStudio pop up dialog for git commit and other git related work

2. Write down a commit message (free form, but keep it informative)

Let's write the following text in the commit message box and press "commit"

```
saving my first Rmarkdown document
```

— happy days!

## 5.4 Git username and useremail

When you are using git for very first time on a new compute you will need to set up your username and email. Typically this is one off operation and almost always forgot about. But git will always remind you about that with the following message 5.4

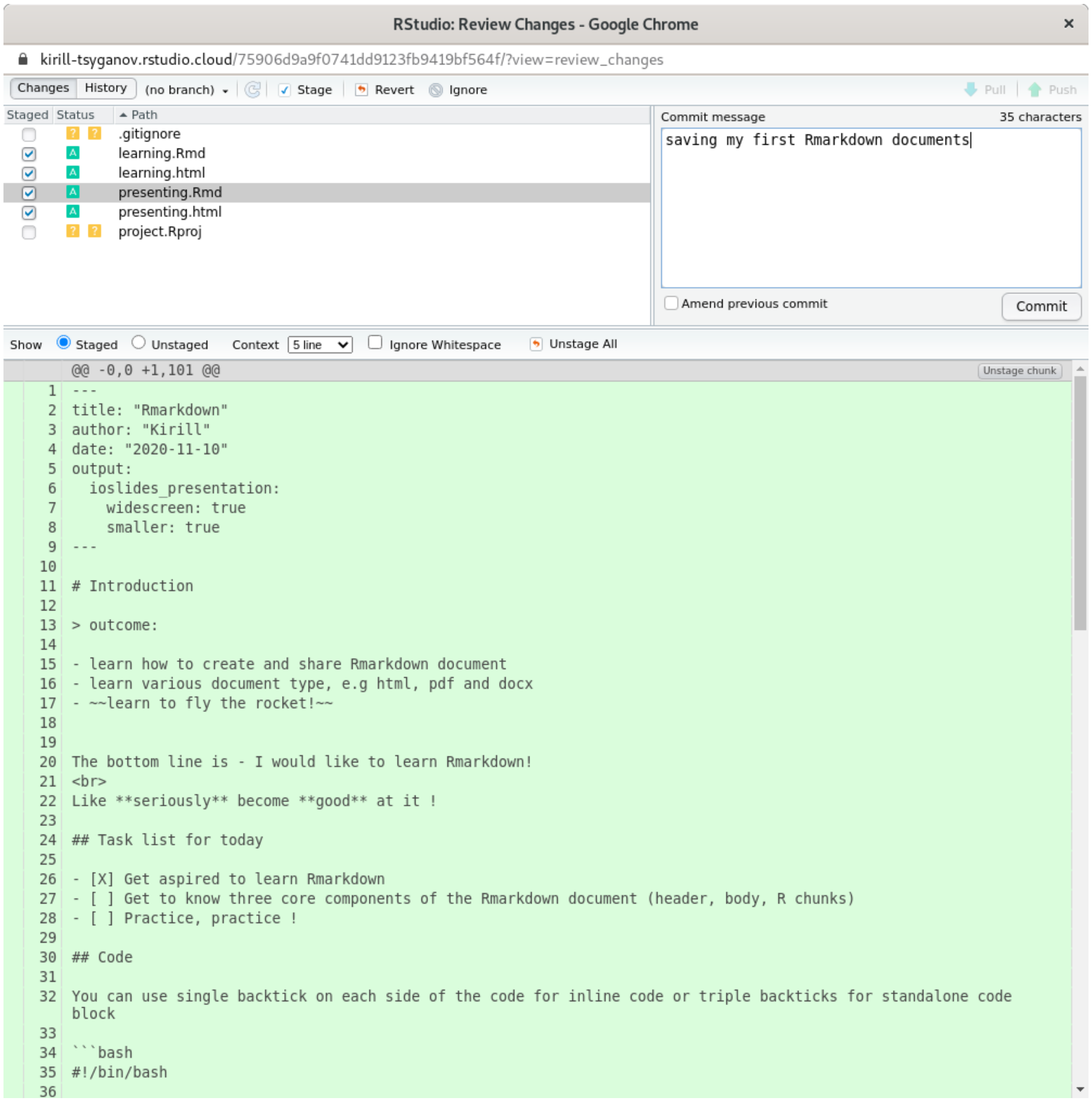
Let's execute the following two commands in R console pane, as suggested by the message. Note if you don't use `--global` flag, then every new git repository will need those two commands (personal preference really).

```
system("git config --global user.name kirill")
```

```
system("git config --global user.email tskiril@gmail.com")
```

Let's switch back to our commit, pop up, window close that messagea and click commit again.

— oops, forgot about that

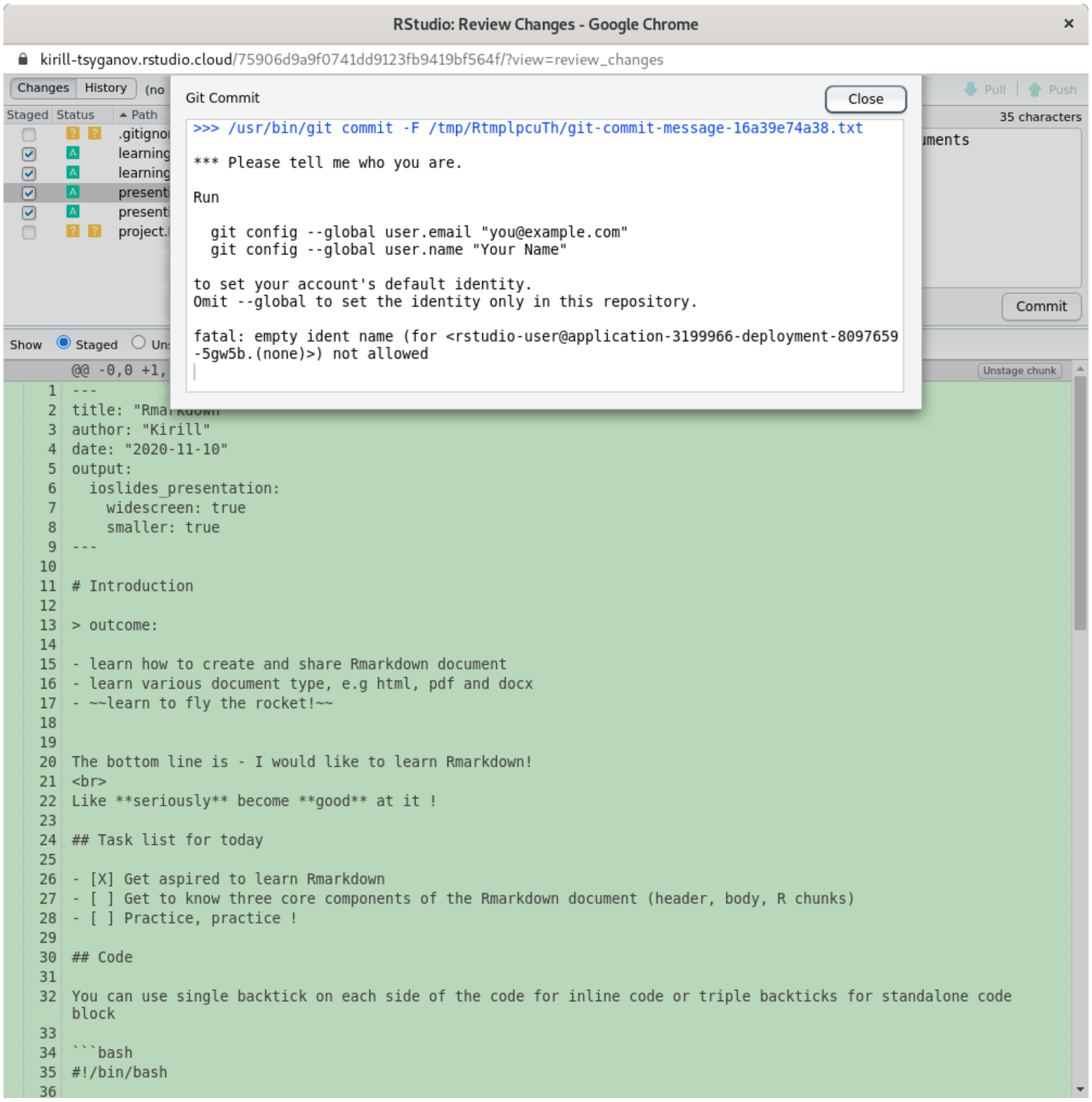


| Staged | Status | Path |
|---|---|---|
| ☐ | ? ? | .gitignore |
| ✓ | A | learning.Rmd |
| ✓ | A | learning.html |
| ✓ | A | presenting.Rmd |
| ✓ | A | presenting.html |
| ☐ | ? ? | project.Rproj |

```
saving my first Rmarkdown documents
```

~~~
@@ -0,0 +1,101 @@
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-10"
output:
  ioslides_presentation:
    widescreen: true
    smaller: true
---

# Introduction

> outcome:

- learn how to create and share Rmarkdown document
- learn various document type, e.g html, pdf and docx
- ~~learn to fly the rocket!~~


The bottom line is - I would like to learn Rmarkdown!
<br>
Like **seriously** become **good** at it !

## Task list for today

- [X] Get aspired to learn Rmarkdown
- [ ] Get to know three core components of the Rmarkdown document (header, body, R chunks)
- [ ] Practice, practice !

## Code

You can use single backtick on each side of the code for inline code or triple backticks for standalone code
block

```bash
#!/bin/bash

~~~

Figure 5.3: RStudio pop up dialog for git commit and other git related work with commit message


RStudio: Review Changes - Google Chrome
kirill-tsyganov.rstudio.cloud/75906d9a9f0741dd9123fb9419bf564f/?view=review_changes
Changes
History
Pull
Push
Staged
Status
Path
35 characters
Commit
Show
Staged
Unstage chunk
Git Commit
Close

```
>>> /usr/bin/git commit -F /tmp/RtmplpcuTh/git-commit-message-16a39e74a38.txt

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: empty ident name (for <rstudio-user@application-3199966-deployment-8097659
-5gw5b.(none)>) not allowed
```

~~~
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-10"
output:
  ioslides_presentation:
    widescreen: true
    smaller: true
---

# Introduction

> outcome:

- learn how to create and share Rmarkdown document
- learn various document type, e.g html, pdf and docx
- ~~learn to fly the rocket!~~


The bottom line is - I would like to learn Rmarkdown!
<br>
Like **seriously** become **good** at it !

## Task list for today

- [X] Get aspired to learn Rmarkdown
- [ ] Get to know three core components of the Rmarkdown document (header, body, R chunks)
- [ ] Practice, practice !

## Code

You can use single backtick on each side of the code for inline code or triple backticks for standalone code
block

```bash
#!/bin/bash
~~~

Figure 5.4: RStudio pop up dialog showing additional step needs doing before using git

# Chapter 6

# GitHub (part one)

In this part of the book we will introduce GitHub[1], which is one of several places you can host - store and display your project files. GitHub can only work with git tool, hence the name. However there are other vendors that are a more tools agnostic

## 6.1 Introduction

GitHub has been around for more than 10 years. The original, and the most dominant still, purpose is store software source code in public. By putting your code on display gave at least two important benefits:

1. "free" and easy code oversight: sanity and bugs checks
2. ease of collaboration: take a copy and play with it, PR (pull request) later

GitHub platform has become indispensable to the software industry.

There is some lingo that is associated with GitHub, the most central one is "repository" or "repo" for short. That simply refer to a folder (a.k.a directory) on your computer that had git initiated as per Git init section. Your GitHub account can have any any number of repos, each one of them is a distinct project.

## 6.2 New repository

> Prerequisite: Have to register for a GitHub account first at github.com[2]

Let's begin by making a new repository. Go to github.com[3] and look for plus sign (New repository). Interface may change time to time, but look at the top right hand corner. Once you've clicked on "New repository" button, you should see the following window 6.1

Let's give our new repository a name and a short description. Type the following text and press green button on the bottom of the screen "Create repository"

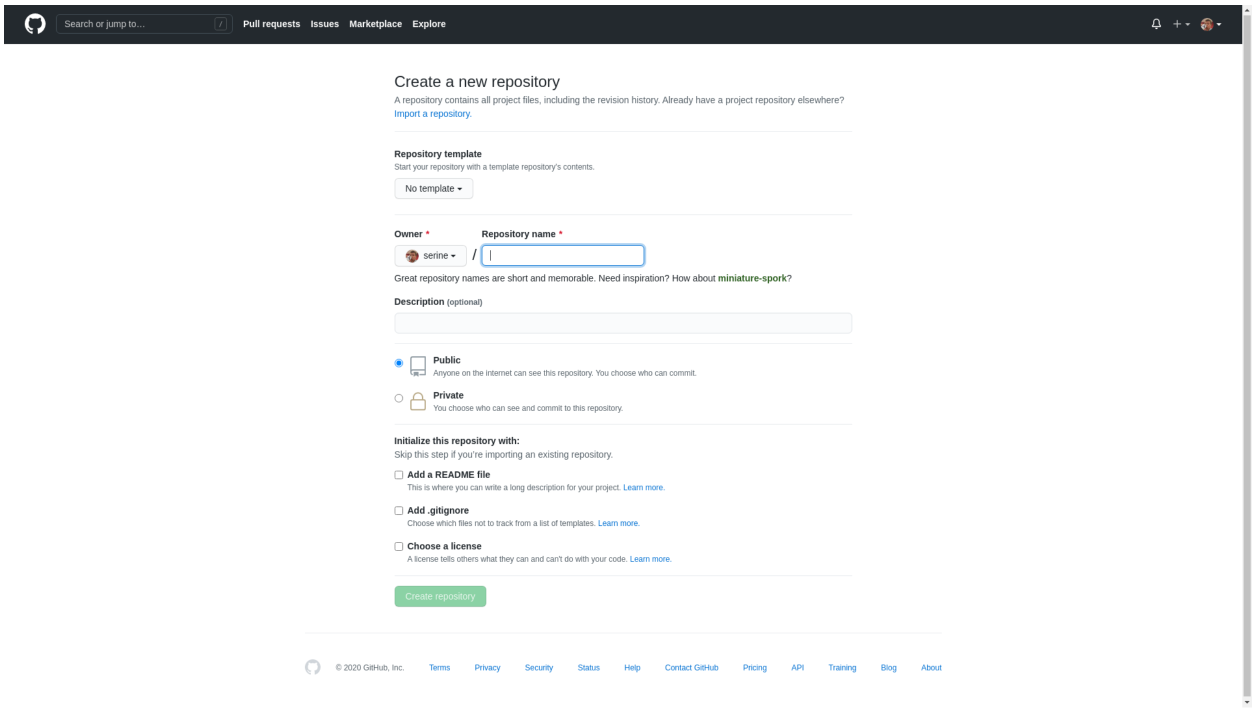
```
learning_rmarkdown
```

```
I'm learning Rmarkdown, Git and GitHub
```

Notes:

- Don't select any other options, keep repository public

[1]https://github.com
[2]https://github.com
[3]https://github.com



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Repository template

Start your repository with a template repository's contents.

No template

Owner * Repository name *

serine /

Great repository names are short and memorable. Need inspiration? How about miniature-spork?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

Choose a license
A license tells others what they can and can't do with your code. Learn more.

Create repository

© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About

Figure 6.1: New repository dialog at github.com

- It is a good practice **NOT** to use spaces in your repository name. Here we are using an underscore instead.

## 6.3 Repository address (git remote‘)

Once you’ve clicked on “New repository” button you should see the following, new window appearing in front of your 6.3 (provided you didn’t select any of the additional files, like README file)

Copy on your clipboard URL that is presented to you at the top of the page. For me the url looks the following

```
https://github.com/serine/learning_markdown.git
```

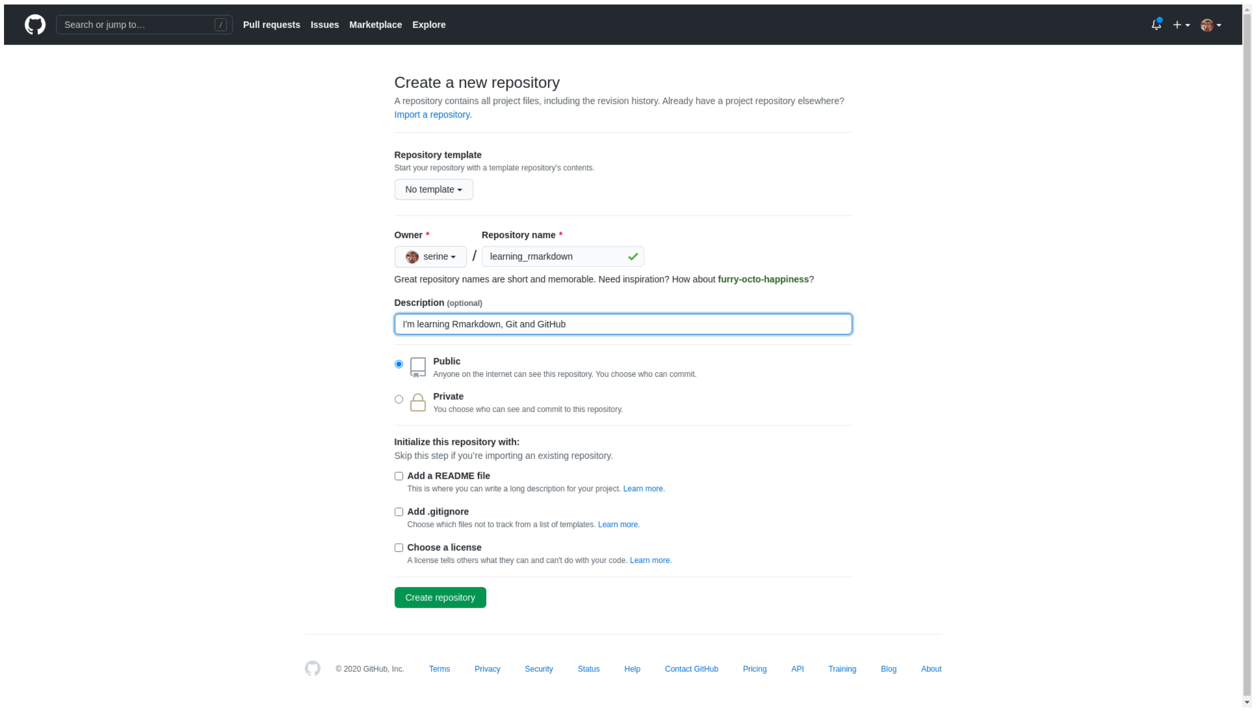
It will look very similar in your case, except the username part. The following if your typical URL structure for github repositories. Note that square brackets is a convention meaning “substituted with”, they don’t need to be included in your actual URL

```
https://github.com/[USERNAME]/[REPOSITORY_NAME]
```

## 6.4 Linking Git with GitHub

Now that you have successfully created new repository at GitHub we need to link (add) address of our github repository to our local git folder (also called repository). This simply means letting git tool know where to deposit (push) our code for storage and display.

Let’s switch back to the RStudio client, where we have been doing work and execute the following command in R console. Note you will need to **substitute** your own URL (github repository address) below.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

**Repository template**
Start your repository with a template repository's contents.

No template

**Owner *** **Repository name ***

serine / learning_rmarkdown ✓

Great repository names are short and memorable. Need inspiration? How about **furry-octo-happiness**?

**Description** (optional)

I'm learning Rmarkdown, Git and GitHub

● **Public**
Anyone on the internet can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

**Initialize this repository with:**
Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. Learn more.

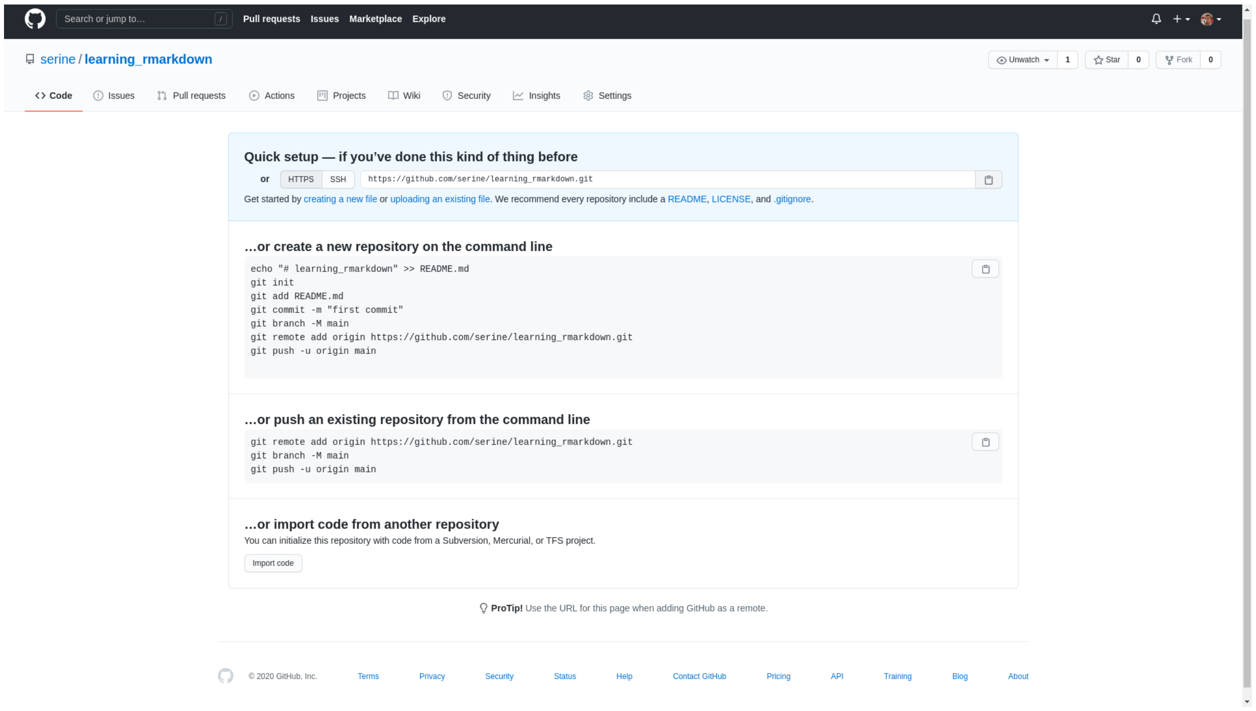
☐ **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.

☐ **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

Create repository

© 2020 GitHub, Inc.

Figure 6.2: New repository dialog at github.com, with text



serine / **learning_rmarkdown**

**Quick setup — if you've done this kind of thing before**

or HTTPS SSH `https://github.com/serine/learning_rmarkdown.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

**...or create a new repository on the command line**

```
echo "# learning_rmarkdown" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/serine/learning_rmarkdown.git
git push -u origin main
```

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/serine/learning_rmarkdown.git
git branch -M main
git push -u origin main
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

**ProTip!** Use the URL for this page when adding GitHub as a remote.

© 2020 GitHub, Inc.

Figure 6.3: Secondary window, once new repository was created at github.com

```
system("git remote add origin https://github.com/serine/learning_rmarkdown")
```

Now you have linked your local git repository with remote one (a.k.a github repository) you can start synchronising your content between two locations. In git lingo **pushing and pulling**. Let's talk more about that in the next section

# Chapter 7

# Git (part two)

In this section we will learn a typical workflow of **pulling** and **pushing** our files and data between local git and remote github repositories. In the next section we will explore some of the GitHub platform specific features that will helps us with sharing our research.

## 7.1 Introduction

As was discussed in previous section we need to link our local git repository with the remote one at the github platform.

For some odd reason you can't simply copy and paste github remote address into relevant section of the setup window using RStudio client. In the figure 7.1 below, input box "Origin" is grayed out. I have tried and I cannot find a way to update that via RStudio client interface.

Steps were described in git init section of the book on how to get to that dialog popup.

## 7.2 Linking Git with GitHub (repeat)

Let's add our github remote address to git via R console command. **Remember to substitute** your own URL (github repository address) into the command below. We also need to relaunch our R project, but closing and reopening RStudio project. This is another, minor, peculiarity of working with git via RStudio client

*If you are using rstudio.cloud*[1]*, relaunch the project via a drop down menu top right hand side corner, three little dots next to the cog symbol, select "Relaunch Project"*

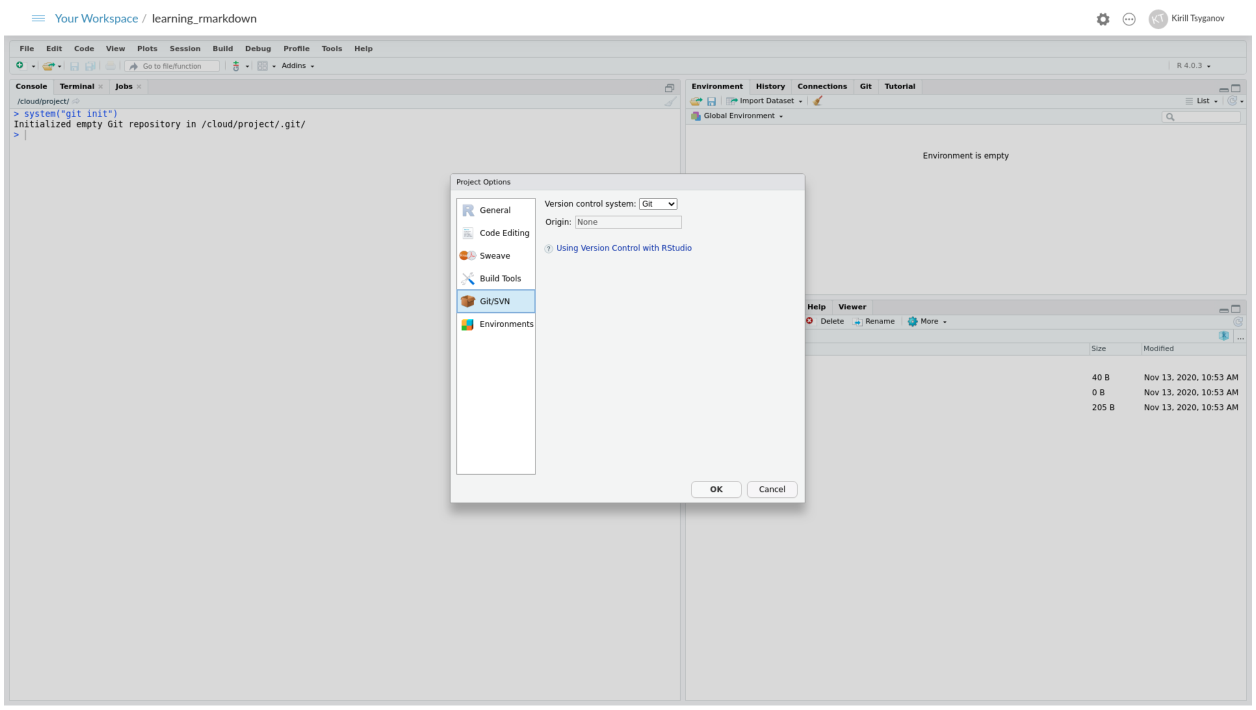
```
system("git remote add origin https://github.com/serine/learning_rmarkdown")
```

Now if you navigate to that the version control dialog you should see "Origin" input box has been populated with your address, as per the figure 7.2

## 7.3 "origin" lingo in git

Origin in the context of git means remote location for your git repository. It is a key (short hand representation) for our URL for when we are pushing and pulling code between local and remote repositories. We don't have to use "origin" key name, we can use any other names, for example `project_x_github_repo`. The only caveat to that RStudio client won't be able to understand that keyword.

[1]https://rstudio.cloud

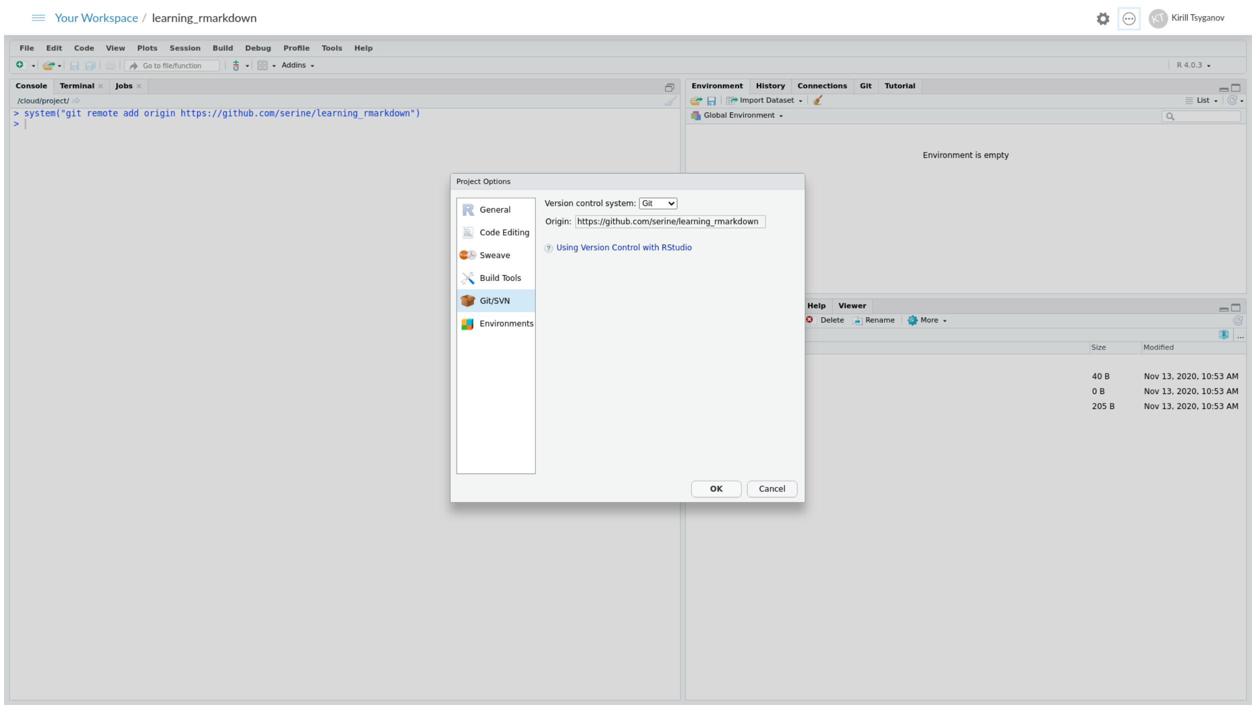


```
> system("git init")
Initialized empty Git repository in /cloud/project/.git/
>
```

Your Workspace / learning_rmarkdown
Kirill Tsyganov
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Jobs
/cloud/project/
Environment History Connections Git Tutorial
Import Dataset
List
Global Environment
Environment is empty
Help Viewer
Delete Rename More
Size Modified
40 B Nov 13, 2020, 10:53 AM
0 B Nov 13, 2020, 10:53 AM
205 B Nov 13, 2020, 10:53 AM
Project Options
General
Code Editing
Sweave
Build Tools
Git/SVN
Environments
OK
Cancel
Using Version Control with RStudio

Version control system: Git
Origin: None

Figure 7.1: Version control dialog



```
> system("git remote add origin https://github.com/serine/learning_rmarkdown")
>
```

Your Workspace / learning_rmarkdown
Kirill Tsyganov
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Jobs
/cloud/project/
Environment History Connections Git Tutorial
Import Dataset
List
Global Environment
Environment is empty
Help Viewer
Delete Rename More
Size Modified
40 B Nov 13, 2020, 10:53 AM
0 B Nov 13, 2020, 10:53 AM
205 B Nov 13, 2020, 10:53 AM
Project Options
General
Code Editing
Sweave
Build Tools
Git/SVN
Environments
OK
Cancel
Using Version Control with RStudio

Version control system: Git
Origin: https://github.com/serine/learning_rmarkdown

Figure 7.2: Version control dialog

If you would like to try this out, run the following commands in the R console

```
system("git remote add project_x_github_repo https://github.com/serine/learning_rmarkdown")
```

```
system("git remote -v")
```

## 7.4 git push

We have already learned how to checkpoint our files with git commit in the previous section. Let's now synchronise our local repository with remote github one by pressing on green arrow up button "push" in the "Git" tab. We will be prompted to enter our github username and the password.

Let's switch to GitHub window in our web browser, update the webpage to see our code appearing on the remote GitHub platform.

— marvelous !

## 7.5 git pull

`git pull` is a complementary command to `git push`, which is actually does two things `git fetch` - bring remote changes to your local repository and `git merge` synchronise files between local and remote copies.

We will stop our git journey here. If you would like to learn more about git and github look into the resource mentioned in the opening of this part.

— sad :'(

# Chapter 8

# GitHub (part two)

In this section of the book we will introduce to you a few GitHub platform specific features that greatly enhance the git tool and provide incredible visibility and transparency to your project.

## 8.1 README.md

`README` is a gold standard file that holds some essential information about the directory (project) that you found it in. The content of the file is free form, but we would say a bear minimum, at least in relation to the project we have in mind (research code and data, type repositories), `README` file should typically contain; short message about motivation for the project, installation instructions and usage as well as some examples.

You might have realised that `.md` stands for markdown, as oppose to `.Rmd` - Rmarkdown. GitHub will seek this file `README.md` exactly, case sensitive, and it will turn it into a front page of your GitHub repository. We will leave this as home work for you todo later.

> **Home work:** Create file called README.md, you can use the following command. Include, level one header with the project name and three level two headers coresponding to the section; motivations, usage and examples. Then save the file, git commit and git push. You should now have a front cover to your GitHub repository

```
file.edit("README.md")
```

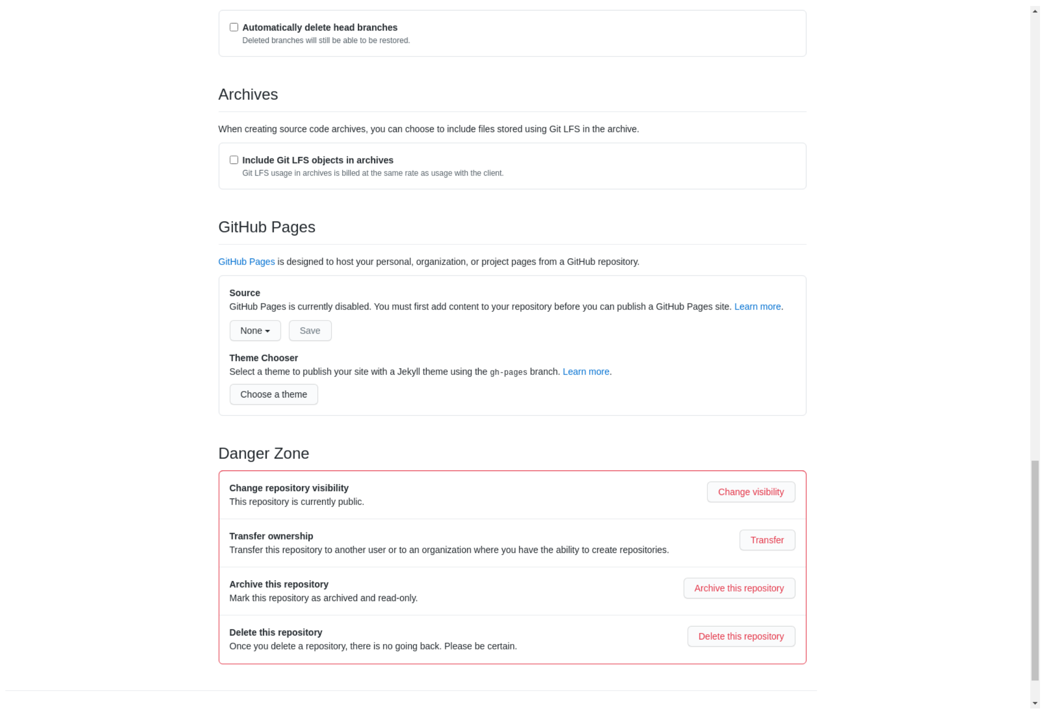
— clever

## 8.2 GitHub pages (gh-pages)

Another useful feature of the GitHub platform is GitHub pages, which allows us to host HTML files. Hosting means making files accessible to the greater internet, sometimes with ability to setup allowed list of users (i.e username and password sign in page). GitHub platform provides **open access hosting** only, to everyone on internet.

Let's enable that feature for our GitHub repository by going to the settings tab, at the top of our repository page and scrolling down to the "GitHub Pages" section, see image 8.1. Select from the drop down menu "master" and click "save" button, you should see the following image 8.2. Copy and Paste URL that appeared at the top of GitHub Pages section, once you've clicked save (you might have to scroll back down to it), you should see the following image 8.3

Once you've set GitHub Pages you can access your HTML file using that URL with additional text pointing to your desired file, for example if I would like to see `learning.html` file I would use the following url https://serine.github.io/learning_rmarkdown/learning.html



☐ **Automatically delete head branches**
Deleted branches will still be able to be restored.

Archives

When creating source code archives, you can choose to include files stored using Git LFS in the archive.

☐ **Include Git LFS objects in archives**
Git LFS usage in archives is billed at the same rate as usage with the client.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

**Source**
GitHub Pages is currently disabled. You must first add content to your repository before you can publish a GitHub Pages site. Learn more.

None ▾ Save

**Theme Chooser**
Select a theme to publish your site with a Jekyll theme using the `gh-pages` branch. Learn more.

Choose a theme

Danger Zone

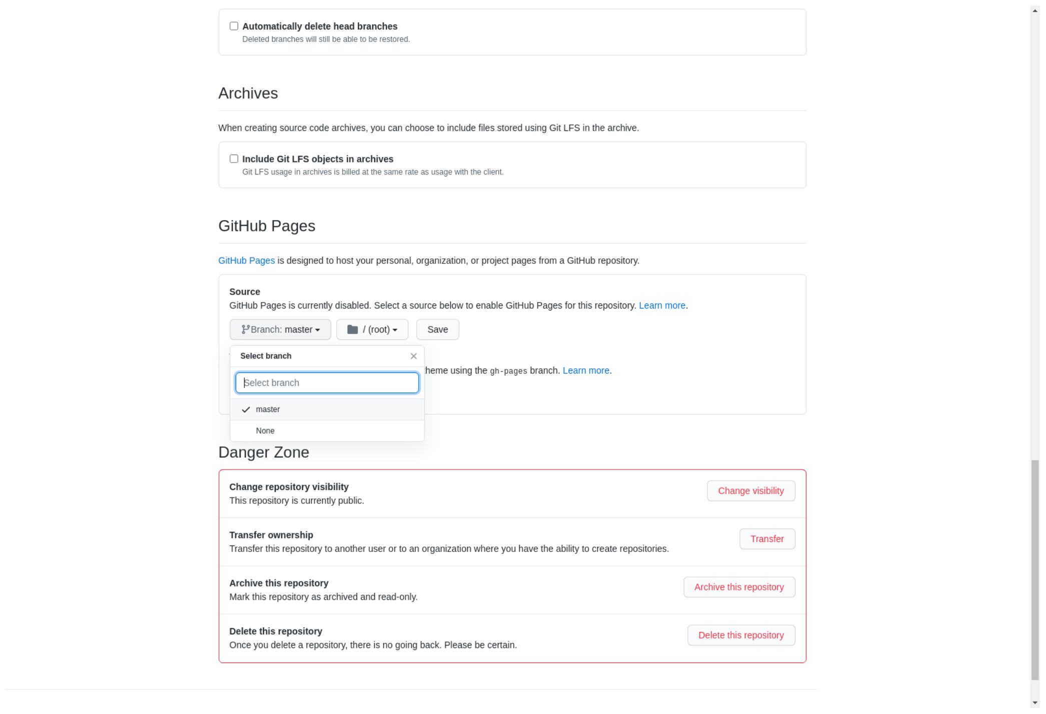
**Change repository visibility**
This repository is currently public. — Change visibility

**Transfer ownership**
Transfer this repository to another user or to an organization where you have the ability to create repositories. — Transfer

**Archive this repository**
Mark this repository as archived and read-only. — Archive this repository

**Delete this repository**
Once you delete a repository, there is no going back. Please be certain. — Delete this repository

Figure 8.1: GitHub Pages section at the GitHub website



☐ **Automatically delete head branches**
Deleted branches will still be able to be restored.

Archives

When creating source code archives, you can choose to include files stored using Git LFS in the archive.

☐ **Include Git LFS objects in archives**
Git LFS usage in archives is billed at the same rate as usage with the client.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

**Source**
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. Learn more.

Branch: master ▾ / (root) ▾ Save

**Select branch** ×

Select branch

✓ master

None

heme using the `gh-pages` branch. Learn more.

Danger Zone

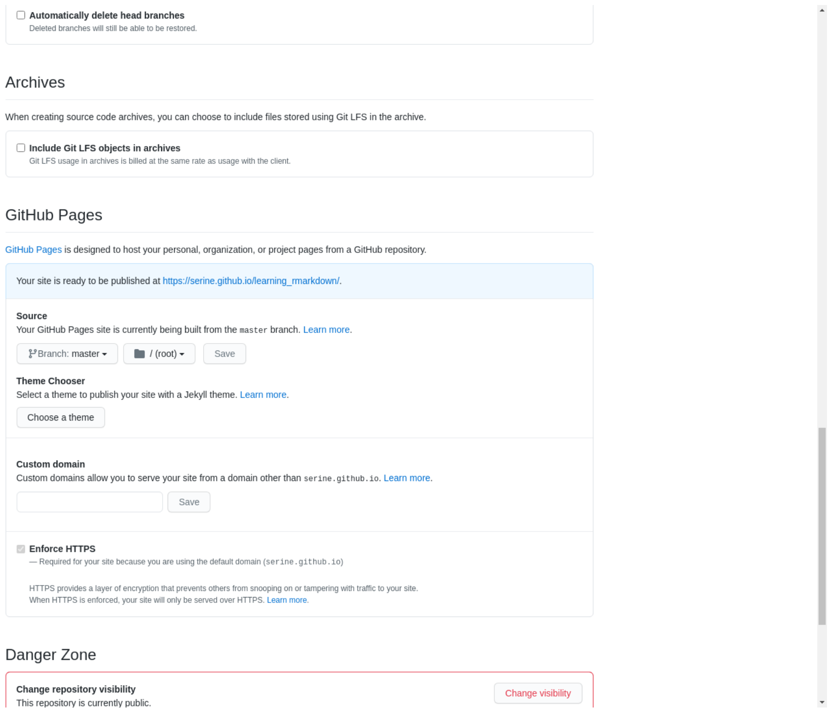
**Change repository visibility**
This repository is currently public. — Change visibility

**Transfer ownership**
Transfer this repository to another user or to an organization where you have the ability to create repositories. — Transfer

**Archive this repository**
Mark this repository as archived and read-only. — Archive this repository

**Delete this repository**
Once you delete a repository, there is no going back. Please be certain. — Delete this repository

Figure 8.2: GitHub Pages section at the GitHub website, with “master” branch selected



Automatically delete head branches
Deleted branches will still be able to be restored.

Archives

When creating source code archives, you can choose to include files stored using Git LFS in the archive.

Include Git LFS objects in archives
Git LFS usage in archives is billed at the same rate as usage with the client.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is ready to be published at https://serine.github.io/learning_rmarkdown/.

Source
Your GitHub Pages site is currently being built from the `master` branch. Learn more.

Theme Chooser
Select a theme to publish your site with a Jekyll theme. Learn more.

Custom domain
Custom domains allow you to serve your site from a domain other than `serine.github.io`. Learn more.

Enforce HTTPS
— Required for your site because you are using the default domain (`serine.github.io`)

HTTPS provides a layer of encryption that prevents others from snooping on or tampering with traffic to your site.
When HTTPS is enforced, your site will only be served over HTTPS. Learn more.

Danger Zone

Change repository visibility
This repository is currently public.

Figure 8.3: GitHub Pages section at the GitHub website, with URL displayed

General schema for GitHub Pages URL as follows

```
https://[USERNAME].github.io/[REPOSITORY_NAME1]/[FILENAME].html
```

— fair dinkum !

## 8.3 Visible contributions

It is hard to easily see a number of contributors to rmarkdown R package project, however number of "forks" is one reasonable indicator for how many people are attempting to collaborate.

Compare:

- community contributions
- stars
- forks
- issue
- pull request

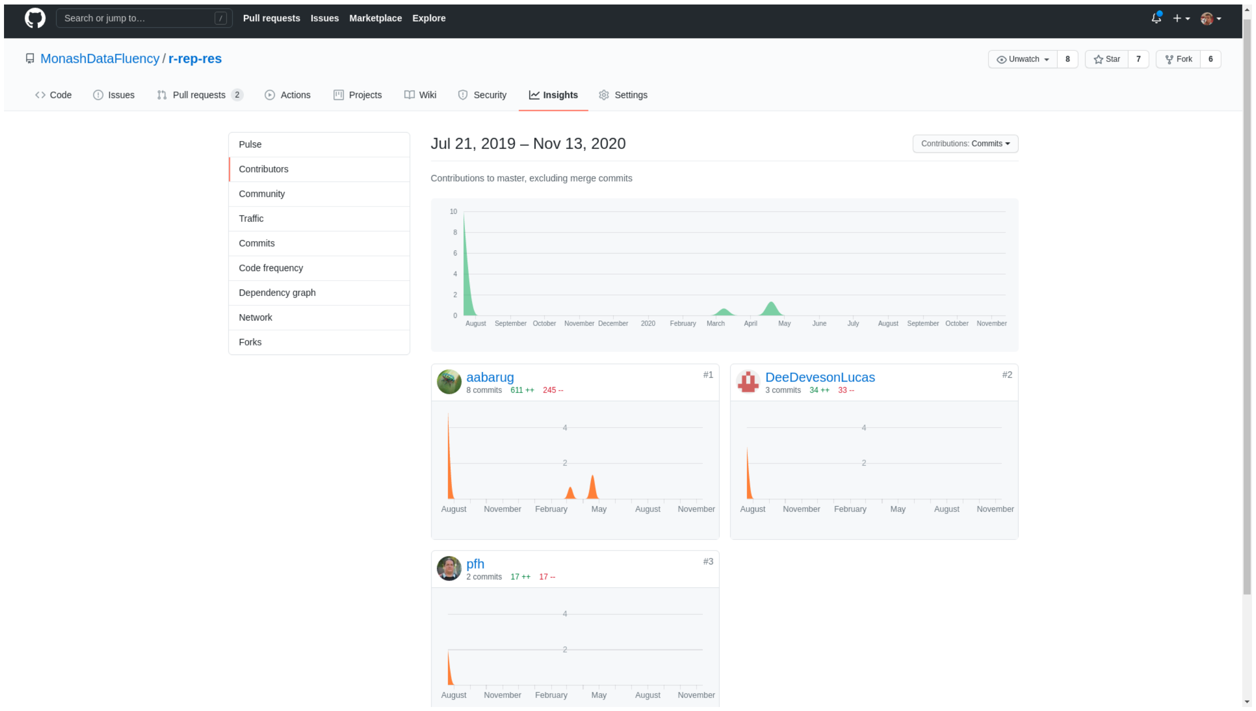


Figure 8.4: Reproducible Research in R (this book) GitHub page, insight tab showing contribution from other users

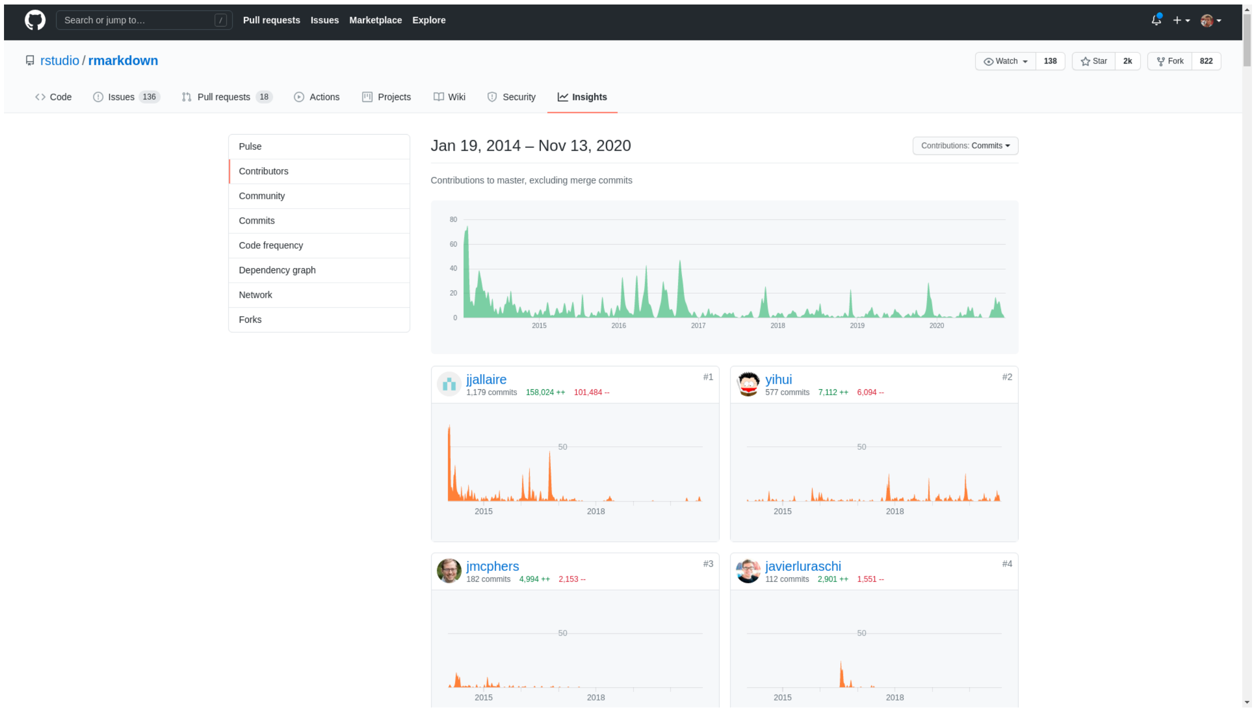


Figure 8.5: Rmarkdown R package GitHub page, insight tab showing contribution from other users

# Part III: Text referencing, Cross referencing and Citation

Linking and referencing different components of the document provides clear pointers to the exact part of the document you are referring to, making it much easier to navigate the document. More importantly referencing external sources demonstrates depth of work and gives rightful acknowledgment to all of the previous authors.

Citations are the scientific currency, please always cite others work

# Chapter 9

# Text referencing

In this section we will explore two different ways for referencing text in the document, classical markdown way, supported by `rmarkdown::html_document` output type and more Rmarkdown specific way supported by `bookdown::html_document2` output type.

## 9.1 Introduction

The most rudimentary referencing we can do in Rmarkdown is linking to the header of the section, which we briefly touched on at the beginning of the book.

### 9.1.1 Starting blob of text (optional)

You may wish to skip this section and instead use any of your previously generated Rmarkdown document. We are going to use slightly trimmed version of what we have been building in the first part of the book.

```
file.edit("learning2.Rmd")
```

~~~
---
title: "Rmarkdown"
author: "Kirill"
date: "2020-11-16"
output:
  html_document:
    toc: false
    theme: readable
    code_folding: "hide"
---

# R code

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE,
                      message = FALSE,
                      warning = FALSE)

library(magrittr)
library(dplyr)
library(DT)
library(ggplot2)
~~~

~~~
```

## Getting and wrangling the data

```{r get_data}

make_url <- function(package) {
file.edit("learning2.Rmd")
  paste0('<a href="https://cran.r-project.org/web/packages/', package, '">', package, '</a>')
}

library(magrittr)
library(dplyr)

avail_packages <- available.packages(contriburl = contrib.url("https://mirror.aarnet.edu.au/pub/CRAN/")
                    as.data.frame() %>%
                    as.data.frame() %>%
                    as_tibble() %>%
                    rowwise() %>%
                    mutate(Depend = length(unlist(strsplit(Imports, split = ",")))) %>%
                    ungroup() %>%
                    select(Package, Version, Depend, Imports, License) %>%
                    mutate(Package = make_url(Package),
                           Depend = ifelse(is.na(Imports), -1, Depend))
```

## Displaying table

```{r make_table}
library(DT)
avail_packages %>% datatable(escape = FALSE)
```

## Plot the data

```{r plot}
library(ggplot2)
avail_packages %>% ggplot(aes(Depend)) + geom_bar()
```
~~~

### 9.1.2 Linking

Three things to know about text linking:

- the syntax `[text](link)`
- the "link" part of the syntax can't have any spaces, replace them with a dash symbol
- the hash symbol (`#`) signals an internal link to the header, as oppose external link to the internet

*allowed alphanumeric characters* `(a-z, A-Z, 0-9)`, `-`, `/`

Let's create a new section at the bottom of our document, using level 1 header with a short sentence beneath.

```
# Text linking

Here is a link to the [Displaying table](#displaying-table) section above
```

## 9.2 bookdown::html_document2

Thus far we have been using `output: html_document` to generate our Rmarkdown documents. That function belong to `rmarkdown` package. Let's us introduce to you another R package, bookdown[1]. This package is an extension of rmarkdown[2] package from the same team. `bookdown::html_document2` function inherent from `rmarkdown::html_document` and provides the following additional features, which we will explore in subsequent section, one at a time:

- text referencing
- cross referencing (figure, tables and sections)
- numbering (figure, tables and sections)

Let's begging by updating our YAML header to this new output type - `bookdown::html_document2` and press `Kint` button. Remember you can use comments symbol (`#`) at the start of the line to mask specific lines in the YAML header.

```
output:
  bookdown::html_document2:
    toc: true
    theme: redable
```

## 9.3 Creating labels

Let's create our first text label with the following syntax `(ref:label)`, where `label` can be any name and add a short sentence that we will want to insert elsewhere in the document later and press `Knit`

```
(ref:barplot) Barplot displaying frequency of dependencies for all of the R packages in the CRAN reposi
```

Note that you should **NOTE** see any text appearing in your final document. This is because we haven't yet referenced the label anywhere in the text

— interesting

## 9.4 Using labels

Let's use our newly created label for figure capture. We will add additional code chunk option into our `plot` chunk using the following code.

~~~
```{r plot, fig.cap='(ref:barplot)'}
~~~

We can reuse that reference label anywhere with the document

— +1

## 9.5 Cross referencing

- `\@ref(intro)`: referencing header section (same results as linking)

can add a section header labels with the following syntax

```
## Rather long header name {#long-header}
```

- `\@ref(fig:label)`: provides pointer to the specific figure

- `\@ref(table:label)`: provies pointer to the specific table

where label is the chunk name

[1]https://github.com/rstudio/bookdown
[2]https://github.com/rstudio/rmarkdown