

Statistics II

Week 1

Content for Today

1. GitHub overview and setup
2. Introduction to `RMarkdown`
3. Assignment peer-review with `ghclass`
4. Assignment workflow
5. Brief `dplyr` review

Labs

The optional drop-in lab sessions exist primarily to help with the coding application of the materials presented in the lecture.

Each session will begin with a brief overview of the week's content and a few minutes for questions, followed by coding exercises that will help you to prepare for the assignments and develop your coding skills.

Please attend the lab you're assigned to.

The scripts for each lab will be on GitHub:

<https://github.com/seramirezruiz/stats-ii-lab>

GitHub

GitHub is a version control platform that is industry-standard in any field where people code (though it can also technically be used with any kind of document).

It allows you to collaborate with others on the same piece of code, and keep records of the changes you've made.

Can be used through the terminal, directly in R Studio, or with the **desktop client** (which is the method we'll use in class).

GitHub: Basic Concepts

- **Repository/repo:** Basically a project folder where your files are stored. Git will track changes to anything in this folder.
- **Clone:** make a copy of a repo so you can access it on your local machine.
- **Commit:** When you make changes, you commit them to “save” them.
- **Push:** After you make a commit, you have to push them to the repo. Otherwise they are just saved on your local machine.
- **Pull:** You can pull from the repo to access any changes made by other collaborators. This will give you the most recent version.
- **README:** File with information about the project.
- **.gitignore:** Files that you do not want Git to track (like data you don't want to upload)
- **Branch:** A copy of the repo you can make changes on without affecting others.
- **Merge:** If there are conflicts between your local copy and the copy in the repo, sometimes you may need to open the file and choose which version of the code to keep in order to merge the two versions.

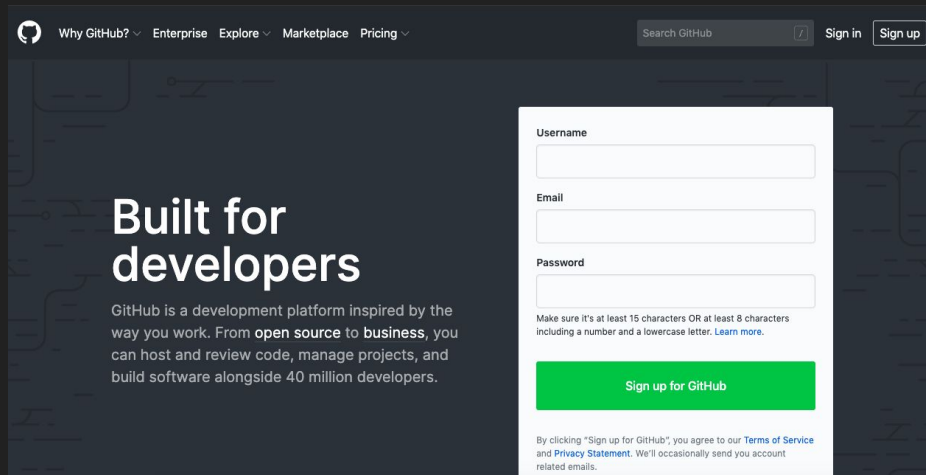
Setting Up GitHub

1. Create an account on github.com
2. Download the desktop client: <https://desktop.github.com/>
3. Create your first repository
4. Make a change to something in the repo and practice committing, pushing, and pulling.

1. Create your GitHub account

<https://github.com/>

1. Select the free version
2. Skip the extra questions
3. Verify your email address



The screenshot shows the GitHub website's sign-up interface. At the top, there is a navigation bar with links for 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. A search bar and 'Sign in'/'Sign up' buttons are also present. The main content area features the heading 'Built for developers' and a paragraph describing GitHub as a development platform. On the right, a white sign-up form is overlaid, containing fields for 'Username', 'Email', and 'Password'. Below the password field, there is a note about password requirements and a 'Sign up for GitHub' button. At the bottom of the form, a small disclaimer states that by signing up, the user agrees to the Terms of Service and Privacy Statement.

Why GitHub? Enterprise Explore Marketplace Pricing

Search GitHub

Sign in Sign up

Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside 40 million developers.

Username

Email

Password

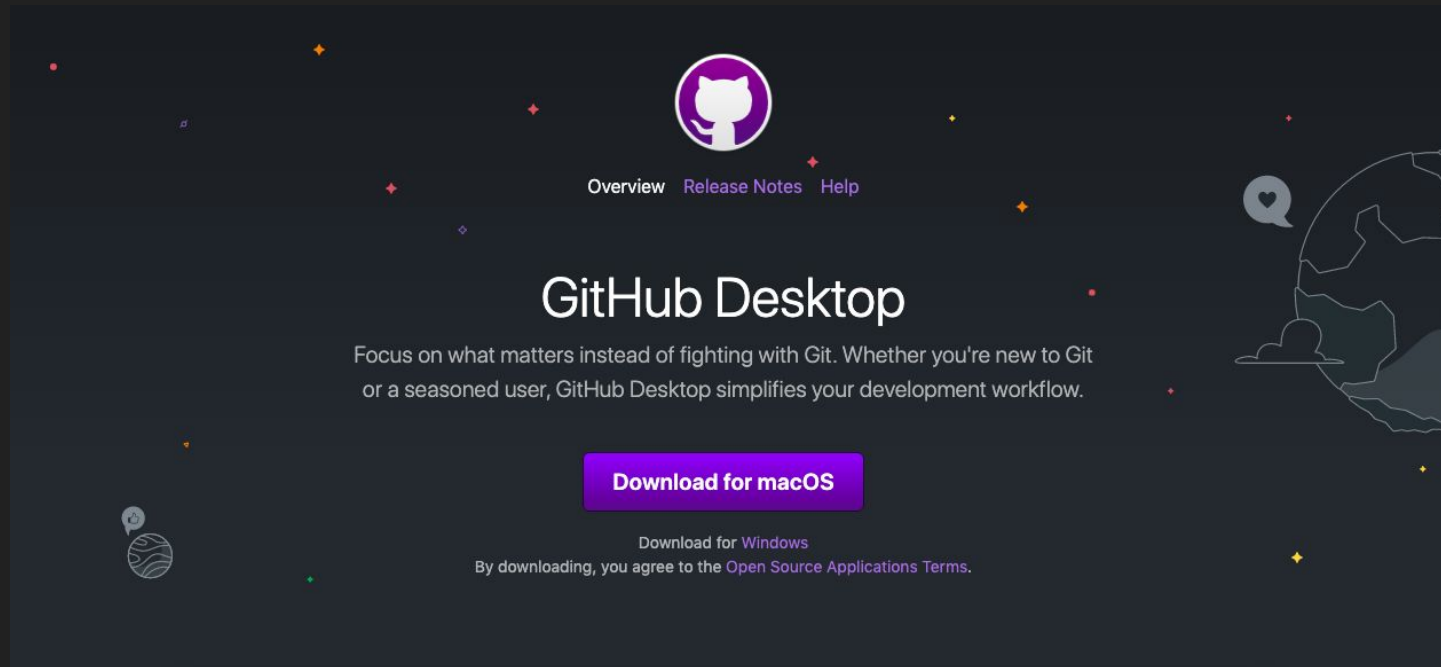
Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Sign up for GitHub

By clicking "Sign up for GitHub", you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails.

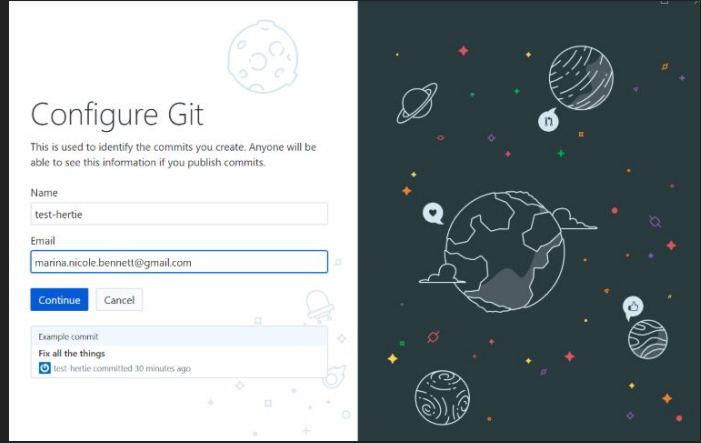
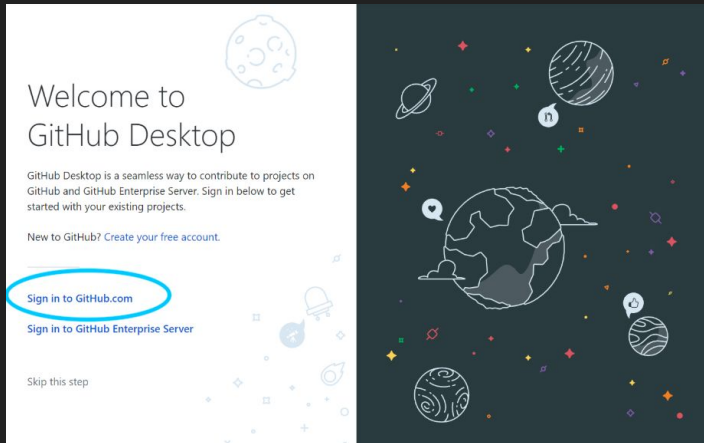
2. Download GitHub Desktop

<https://desktop.github.com/>



2.5 Login and set up

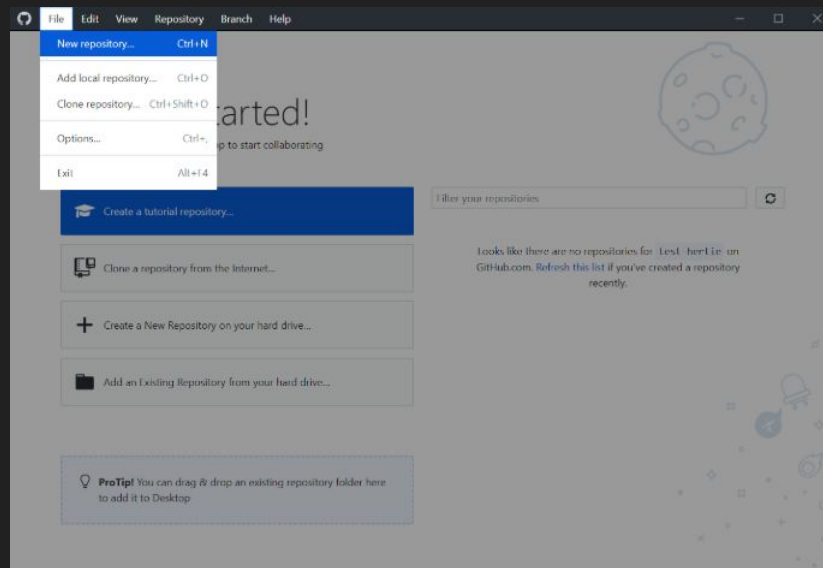
Log in with your GitHub name and email



3. Create your first repository

First, create a folder on your desktop where you want your repo to live.

Maybe where you will keep materials for this class.



3. Create your first repository

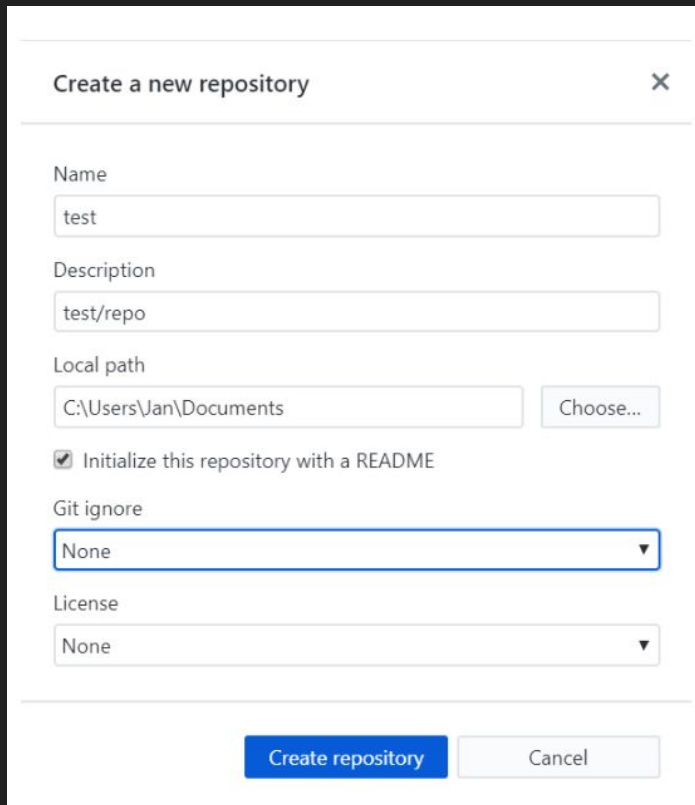
Name: name of your repo

Description: description for your repo

Local path: where you created the folder on your desktop

Click “initialize with a README”

Ignore everything else and create repo



The screenshot shows the 'Create a new repository' dialog box. It has a title bar with a close button (X). The form contains the following fields and options:

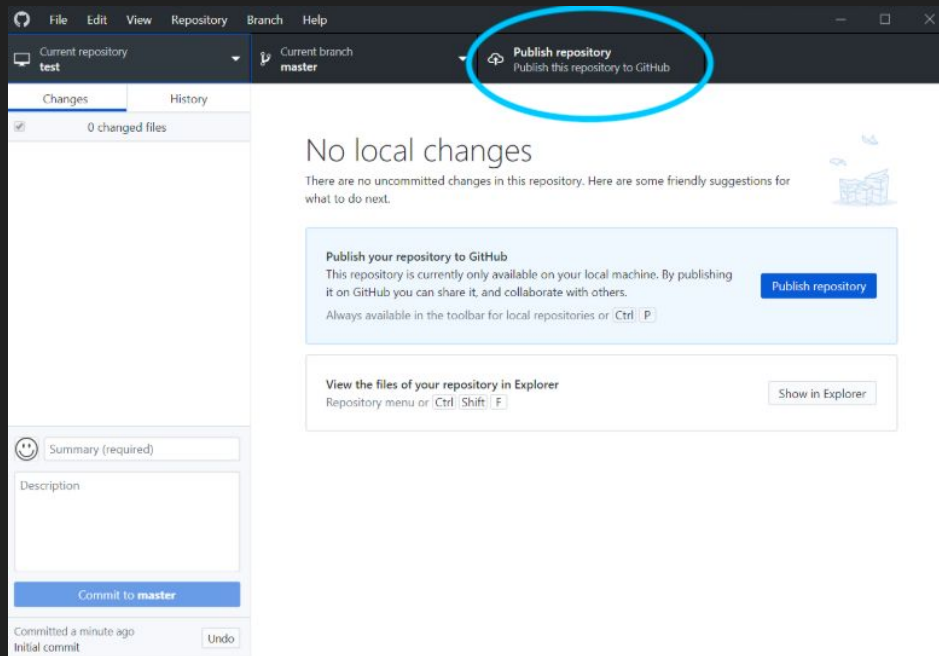
- Name:** A text input field containing 'test'.
- Description:** A text input field containing 'test/repo'.
- Local path:** A text input field containing 'C:\Users\Jan\Documents', followed by a 'Choose...' button.
- Initialize this repository with a README:** A checked checkbox.
- Git ignore:** A dropdown menu with 'None' selected.
- License:** A dropdown menu with 'None' selected.

At the bottom right, there are two buttons: 'Create repository' (highlighted in blue) and 'Cancel'.

3. Create your first repository

Currently, this repo is only available locally. To use it, you need to “**push**” it to GitHub.

Click **publish repository** to do this.

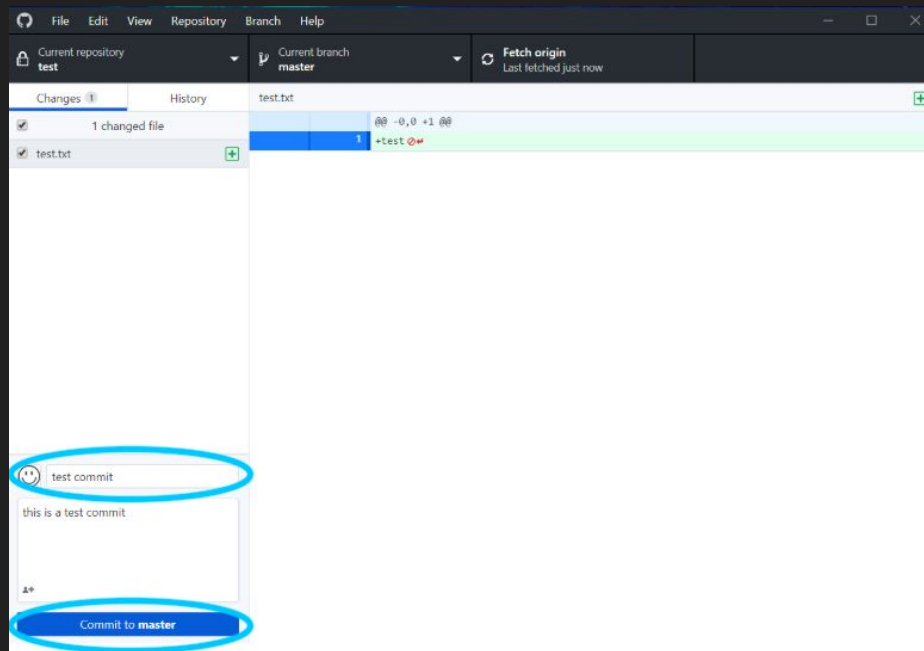


4. First commit

Add a simple text file with “test” etc. to your repo on your desktop.

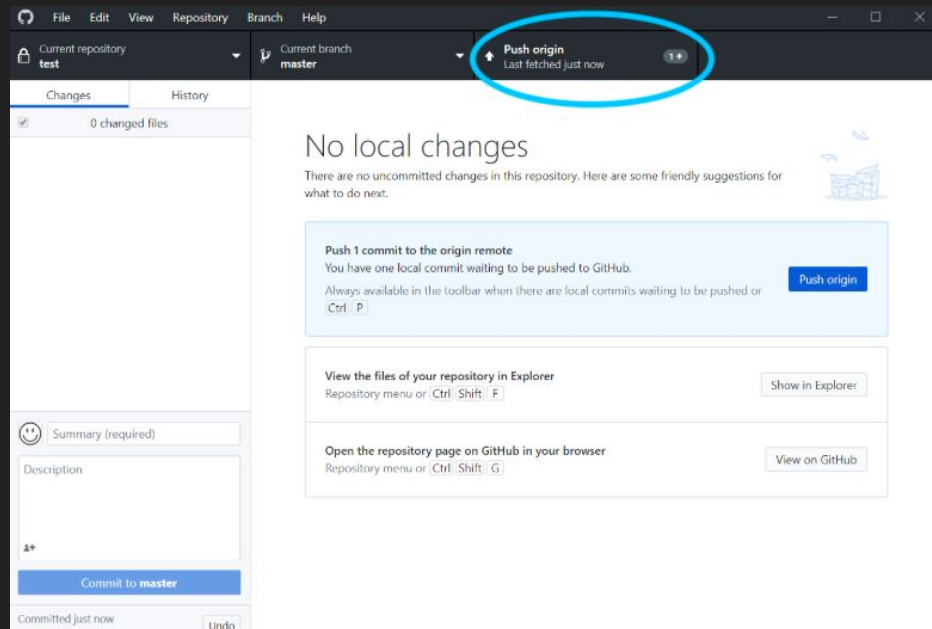
It will then be visible in GitHub Desktop.

Add a “**commit message**,” and click “**commit to master**.”



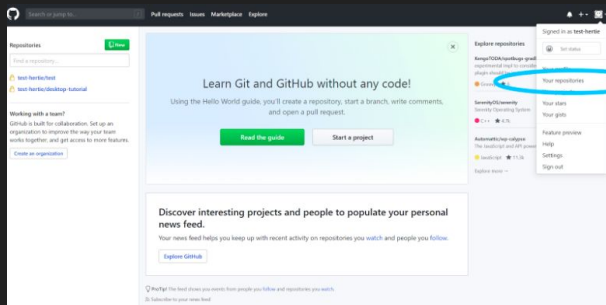
4. First commit

Now click “**push to origin**,” and your changes will be visible on GitHub.

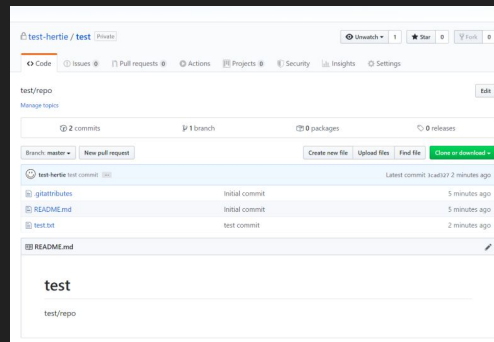


You can now see your commit on GitHub

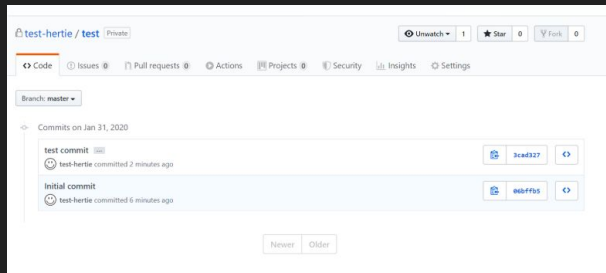
1



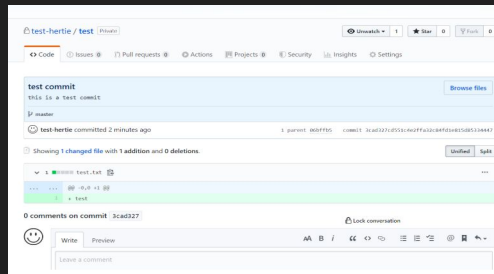
2



3



4

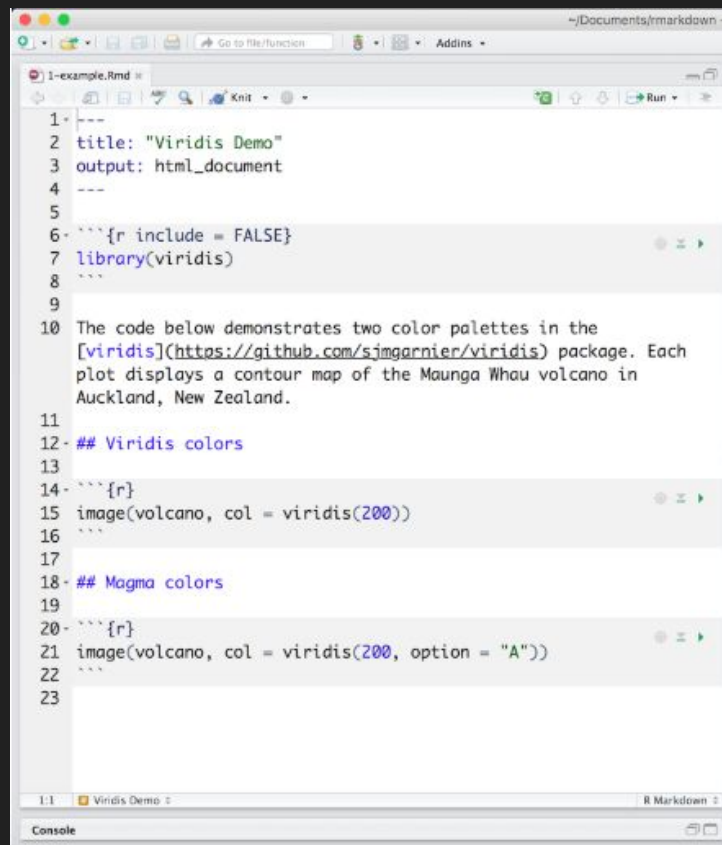


Introduction to Rmarkdown

RMarkdown is an authoring framework for data science. A single RMarkdown file can be used to:

- Save and execute code
- Generate high quality reports that can be shared with an audience

We will use RMarkdown to submit our weekly assignments and review our peers' work.

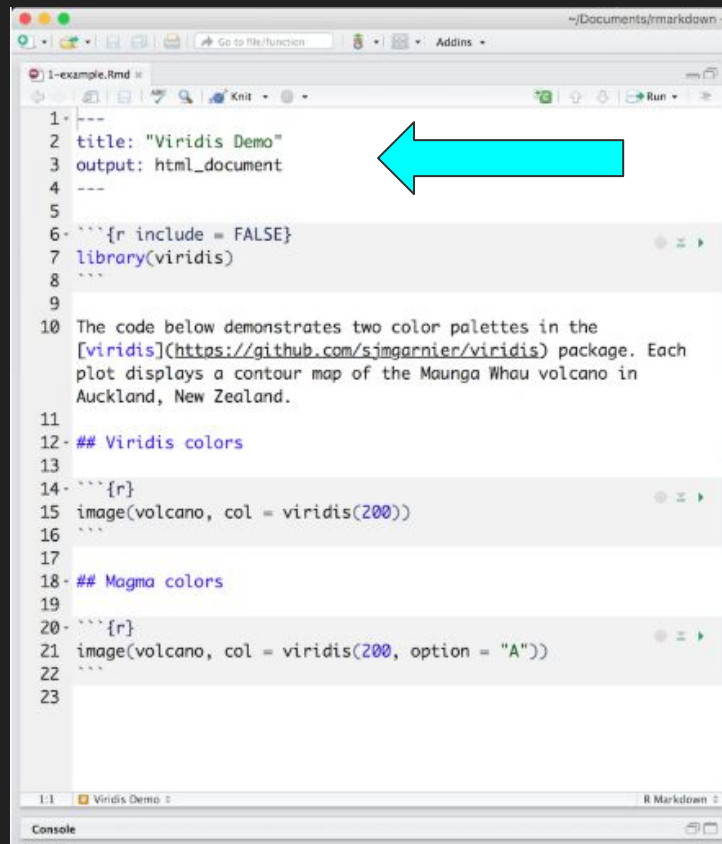


```
1 ---
2 title: "Viridis Demo"
3 output: html_document
4 ---
5
6 ```{r include = FALSE}
7 library(viridis)
8 ```
9
10 The code below demonstrates two color palettes in the
11 [viridis](https://github.com/sjmgarnier/viridis) package. Each
12 plot displays a contour map of the Maunga Whau volcano in
13 Auckland, New Zealand.
14
15 ## Viridis colors
16
17 ```{r}
18 image(volcano, col = viridis(200))
19 ```
20
21 ## Magma colors
22
23 ```{r}
24 image(volcano, col = viridis(200, option = "A"))
25 ```
```


Contents in Rmarkdown

RMarkdown files support three types of content:

- **YAML headers surrounded by ---s**
Meta-data that guides the file build-up process.
(not used in assignments, since submissions are anonymous)



```
1 ---
2 title: "Viridis Demo"
3 output: html_document
4 ---
5
6 ```{r include = FALSE}
7 library(viridis)
8 ```
9
10 The code below demonstrates two color palettes in the
11 [viridis](https://github.com/sjmgarnier/viridis) package. Each
12 plot displays a contour map of the Maunga Whau volcano in
13 Auckland, New Zealand.
14
15 ## Viridis colors
16
17 ```{r}
18 image(volcano, col = viridis(200))
19 ```
20
21 ## Magma colors
22
23 ```{r}
24 image(volcano, col = viridis(200, option = "A"))
25 ```
```

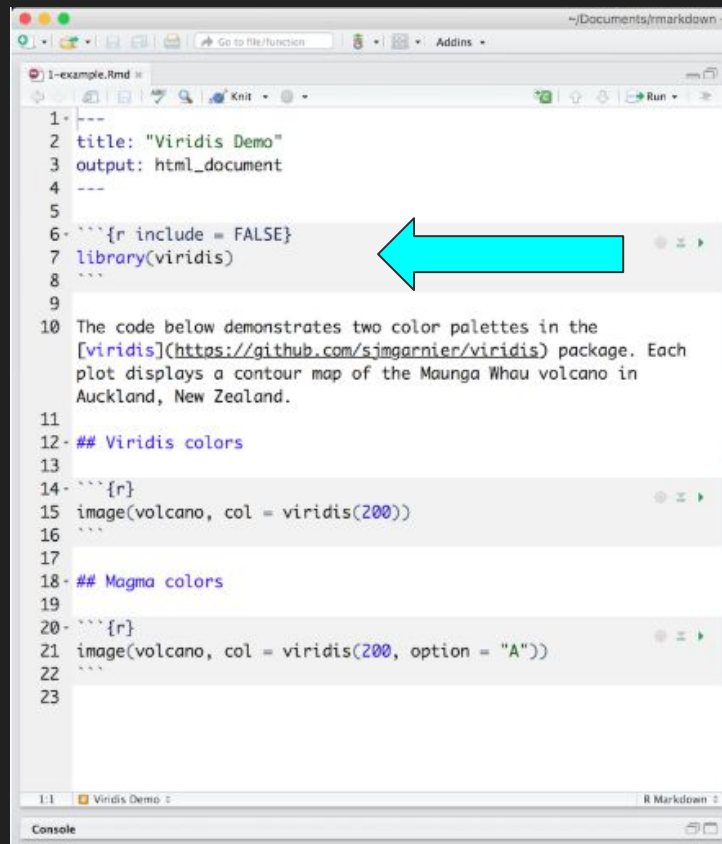
Contents in Rmarkdown

RMarkdown files support three types of content:

- R code chunks surrounded by ````s
Chunks take code as an input. It works in the same way your .R scripts did for Stats I.

start a chunk: `````{r}`

end a chunk: ``````



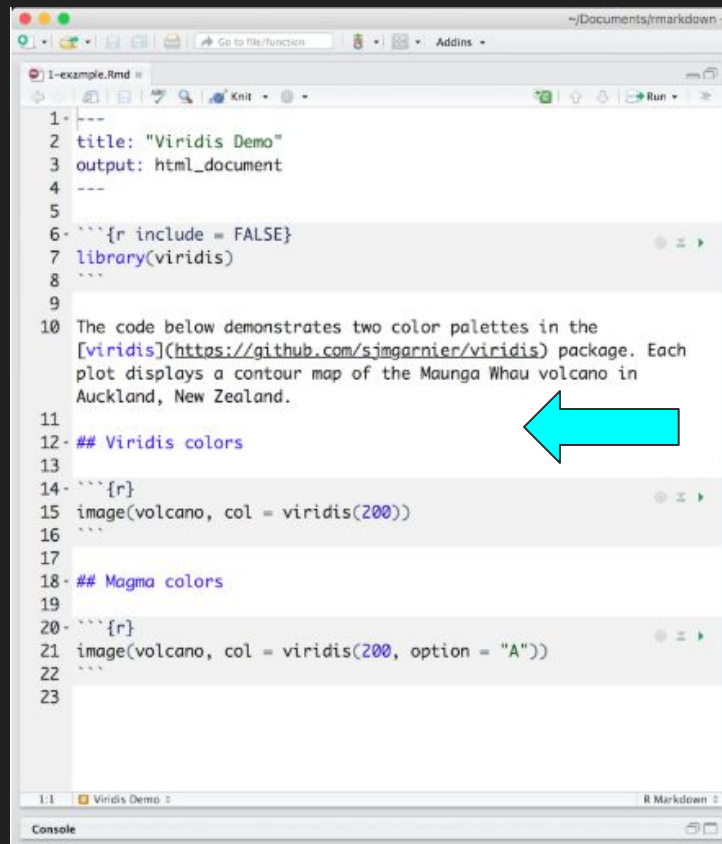
```
1 ---
2 title: "Viridis Demo"
3 output: html_document
4 ---
5
6 ```{r include = FALSE}
7 library(viridis)
8 ```
9
10 The code below demonstrates two color palettes in the
11 [viridis](https://github.com/sjmgarnier/viridis) package. Each
12 plot displays a contour map of the Maunga Whau volcano in
13 Auckland, New Zealand.
14
15 ## Viridis colors
16
17 ```{r}
18 image(volcano, col = viridis(200))
19 ```
20
21 ## Magma colors
22
23 ```{r}
24 image(volcano, col = viridis(200, option = "A"))
25 ```
26
27 
```

Contents in Rmarkdown

RMarkdown files support three types of content:

- text mixed with simple text formatting
Takes text as input.

[RMarkdown Cheatsheet](#)



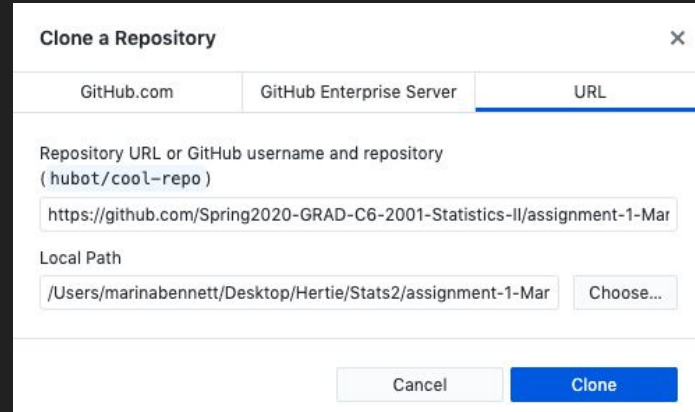
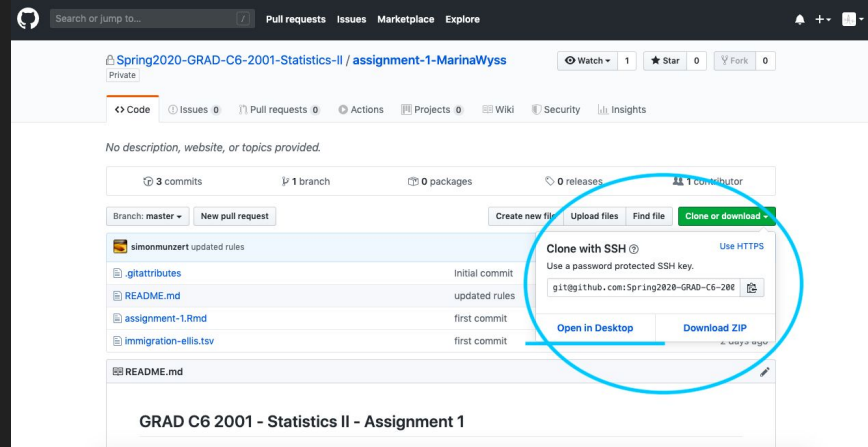
```
1 ---
2 title: "Viridis Demo"
3 output: html_document
4 ---
5
6 ```{r include = FALSE}
7 library(viridis)
8 ```
9
10 The code below demonstrates two color palettes in the
11 [viridis](https://github.com/sjmgarnier/viridis) package. Each
12 plot displays a contour map of the Maunga Whau volcano in
13 Auckland, New Zealand.
14
15 ## Viridis colors
16
17 ```{r}
18 image(volcano, col = viridis(200))
19 ```
20
21 ## Magma colors
22
23 ```{r}
24 image(volcano, col = viridis(200, option = "A"))
25 ```
26
```

ghclass

Each week the professors will invite you to two new private repos each week - one for submissions, and one for reviews.

The first step is to **clone** both repos so you can access them on your desktop and commit changes.

Save them to the folder you created earlier.



ghclass - Your submission

Open GitHub desktop, go to your assignment repo, click **fetch**, then click **pull**. The assignment will be in your assignment desktop folder.

Fill it out, and when you're done, **knit** it to save it as a pretty html file.

Do not change the file names!

Open GitHub desktop again, select the files you want to commit (don't commit useless ones like .Rhistory, .DS_Store, etc.), add a commit message, and click commit, then click **push**. Your assignment has been submitted.

You can repeat the edit - commit - push process to make changes to your submission as many times as you'd like (before the deadline)

ghclass - Reviews

Once the submission deadline has passed, your review repo will contain folders for two of your peers' assignments to review.

Grade their work and fill out your scores in the provided .Rmd document, including in the top header with the q^* scores.

Knit the document (which is a .md file format), and push your changes to GitHub.

ghclass - Rating Reviews

As a final step, you are asked to provide feedback on the quality of the peer-grading of your assignment.

Open GitHub desktop and navigate to the repo where **you created your assignment**, and click **fetch** and **pull**. Two new folders will be in there with your peers' reviews.

Complete the **author rating form**, knit the final document, and commit and push your changes to the .Rmd and .md files from within GitHub desktop.

Assignment Workflow: Rmarkdown, Github, and ghclass

- I. Weekly assignment submission*
- II. Peer-review process
- III. Rating reviewer's feedback

*Students are expected to submit both the .Rmd and a knitted .html versions of their weekly assignments

1. **Clone** the two repos for the week
 - assignment-x-username
 - assignment-x-review-username
2. Read instructions in README.md
3. Complete assignment-x.Rmd
4. Knit to html in RStudio (it will automatically update the .Rmd)
5. **Commit** changes in GitHub Desktop
6. **Push** your submission

Assignment Workflow: Rmarkdown, Github, and ghclass

- I. Weekly assignment submission
- II. Peer-review process*
- III. Rating reviewer's feedback

*Past the submission deadline, `ghclass` will upload the files to your `assignment-x-review-username` folder

**You should have one folder per peer with their `.Rmd` and `.html` submissions, plus the `reviewer-feedback.Rmd` file

1. **Pull** the changes to the `assignment-x-review-username` folder from GitHub Desktop**
2. Explore your peers' submissions
3. Provide and save your feedback in the `reviewer-feedback.Rmd` file of each peer. Knit the `.md` file
4. **Commit** changes in GitHub Desktop
5. **Push** your submission

Assignment Workflow: Rmarkdown, Github, and ghclass

- I. Weekly assignment submission
- II. Peer-review process
- III. Rating reviewer's feedback

*You should have one folder per peer review

1. **Pull** the changes to your assignment-x-username folder from GitHub Desktop*
2. Explore the feedback offered by your peers
3. Rate the feedback and save the author-rating-form.Rmd file of each peer. Knit the .md file
4. **Commit** changes in GitHub Desktop
5. **Push** your submission

Working with R: `dplyr`

Throughout this lab, we will be using the `dplyr` package for most data-wrangling (rather than base R functionality):

We'll often use the pipe operator (`%>%`) to string together commands, and rely on the `dplyr` “verbs”. For example:

`select`: subset columns

`filter`: subset rows

`arrange`: reorder rows

`mutate`: add columns to existing data

`summarize`: summarize values in the dataset

`group_by`: defines groups within dataset

*Let's practice
using the script for
the lab on Github!*

What to do when you get stuck on coding problems

First, don't panic. Then:

1. Check your code (missing parentheses, packages, stray commas, etc.)
2. Google the error message
3. Search on <https://stackoverflow.com/> or look on YouTube
4. Ask for help (from stackoverflow, friends, or your TA)

Further resources

For learning Git and `ghclass`:

- GitHub desktop: <https://tinyurl.com/wu9rjru>
- `ghclass` documentation: <https://tinyurl.com/wfrjol4>

R and RMarkdown:

- Reminder of the basics: <https://tinyurl.com/vkebh2f>
- RMarkdown: The definitive guide <https://tinyurl.com/y4tyfqmq>
- Data wrangling with `dplyr`: <https://tinyurl.com/vyrv596>
- `dplyr` video tutorial: <https://www.youtube.com/watch?v=jWjqLW-u3hc>