Project Management and Reproducibility in RStudio: Part 1

Doug Joubert

2022-10-06

Table of contents

Configuration

This class features exercises that will help you learn by doing. To prepare, please install the following on your machine:

- If you haven't already, please sign-up for a free GitHub account.
- Install the latest version of R, the current version is 4.2.0
- Install latest version of RStudio, v2022.07.0-548 or later

Resources

Introduction to Reproducible Publications with RStudio: https://carpentries-incubator.github.io/Reproducible-Publications-with-RStudio/

R for Reproducible Scientific Analysis: https://swcarpentry.github.io/r-novice-gapminder/

Reproducible Research Data and Project Management in R: https://annakrystalli.me/rrresearchACCE20/

Using Projects: https://support.rstudio.com/hc/en-us/articles/200526207-Using-RStudio-Projects

Introduction

This one-hour class focuses on data and project management using R and RStudio. RStudio makes it possible to work on a complete research project in a more efficient, integrated, and organized manner. RStudio also connects with Git and GitHub and learners will have a chance to experiment with this integration and understand its advantages for collaboration and version control. This intermediate-level course is designed to be relevant to students from different disciplines. Some familiarity or experience in R and RStudio is recommended but not required. Students are encouraged to install R and RStudio before the webinar so that they can follow along with the instructor. Attendees will need to download the class data before the webinar.

Upon completion of this class students should be able to:

- Define scientific reproducibility
- Discuss best practices for organizing data in an RStudio project
- Discuss the importance of using a data dictionary and read me files
- Ensure that their data is machine readable
- Create a new R project using a GitHub repository
- Distinguish between pulling and pushing data from a repository

Scientific Reproducibility

According to the U.S. National Science Foundation (NSF) subcommittee on replicability in science (Bollen et al., 2015):

Science should routinely evaluate the reproducibility of findings that enjoy a prominent role in the published literature. To make reproduction possible, efficient, and informative, researchers should sufficiently document the details of the procedures used to collect data, to convert observations into analyzable data, and to perform data analysis.

Reproducibility refers to the ability of a researcher to duplicate the results of a prior study using the same materials as were used by the original investigator. Reproducibility is a minimum necessary condition for a finding to be considered rigorous, believable, and informative.

Nature Survey on Reproducibility

A 2016 survey in Nature revealed that not being able to reproduce experiments is a problem across all domains of science [Figure 1]:

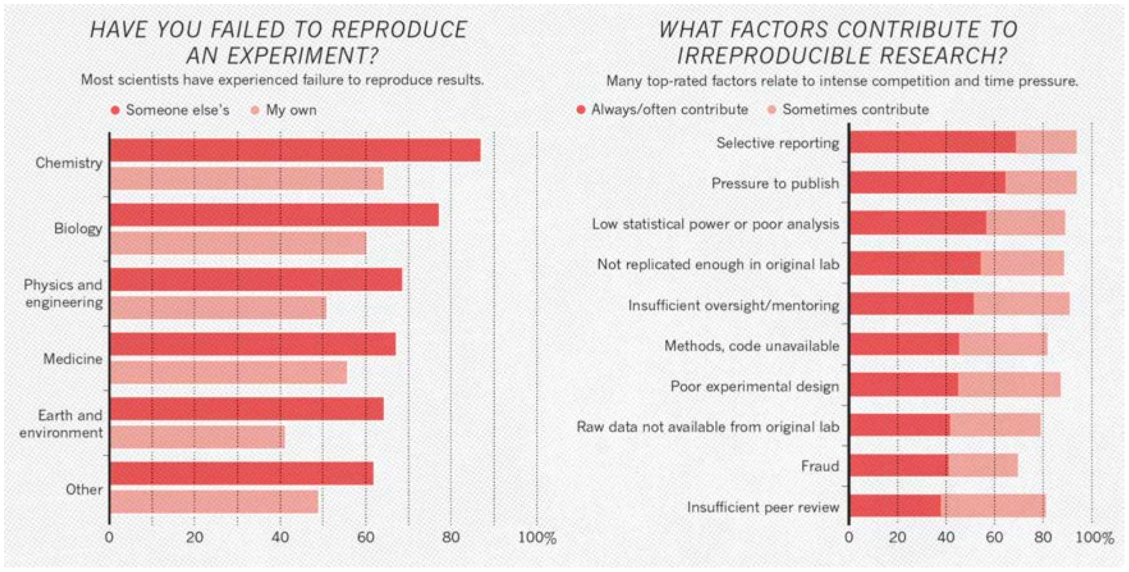


Figure 1: Reproducibility by discipline and problem identified (Baker, 2016).

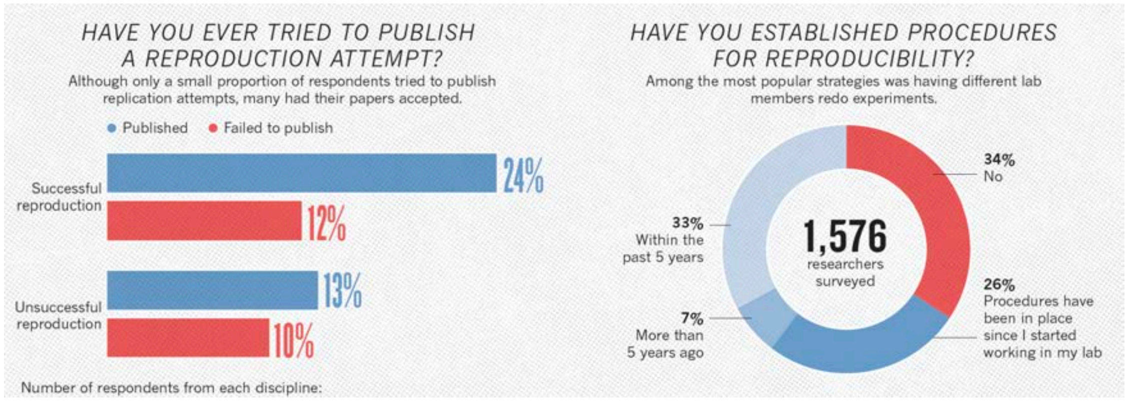
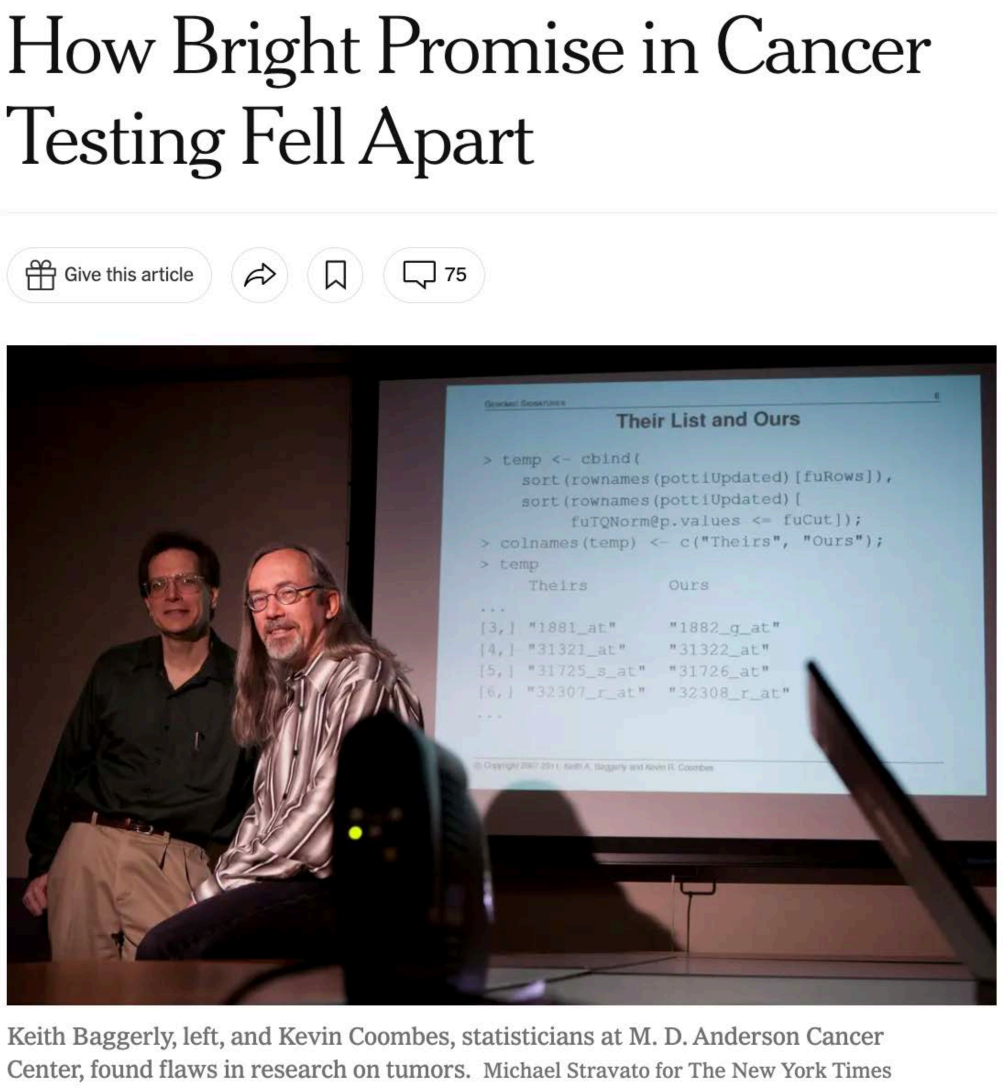


Figure 2: Attempts to publish reproducible research (Baker, 2016).

Five Compelling Reasons to Practice These Principles

If contributing to science and other researchers seems not to be compelling enough, here are 5 selfish reasons to follow reproducibility best practices (Markowetz, 2015).

Helps to avoid data loss and disaster



How Bright Promise in Cancer Testing Fell Apart

Their List and Ours

```
> temp <- cbind(
    sort(rownames(pottiUpdated)[fuRows]),
    sort(rownames(pottiUpdated)[
         fuTQNorm@p.values <= fuCut]);
> colnames(temp) <- c("Theirs", "Ours");
> temp
     Theirs         Ours
...
[3,] "1881_at"      "1882_g_at"
[4,] "31321_at"     "31322_at"
[5,] "31725_s_at"   "31726_at"
[6,] "32307_r_at"   "32308_r_at"
...
```

Keith Baggerly, left, and Kevin Coombes, statisticians at M. D. Anderson Cancer Center, found flaws in research on tumors. Michael Stravato for The New York Times

Figure 3: Two statisticians from MD Anderson found with research from Duke University.

The issues discovered by Baggerly and Coombes could have easily been spotted by any co-author before submitting the paper:

- Data sets are not huge and can easily be spot-checked on a standard laptop
- You do not have to be a statistics wizard to realize that patient numbers differ, labels got swapped or samples appear multiple times with conflicting annotations in the same data set.

Why did no one notice these issues before it was too late? Because the data and analysis were not transparent and required forensic bioinformatics to untangle.

Makes it easier to write papers

Transparency in your analysis makes writing papers much easier. For example, in a dynamic document (R-markdown) so that the results automatically update when the data are changed.

Helps reviewers see it your way

What do you think is a major complaint of authors when going through peer review and addressing comments? "Reviewers didn't even read the paper and had no idea what we were really doing. So, practice making the data and well-documented code easily accessible to the reviewers.

Enables continuity of your work

Can you think of any examples where "not" employing this principle can lead to problems?

- "I am so busy, I can't remember all the details of all my projects"
- "I did this analysis 6 months ago. Of course I can't remember all the details after such a long time"
- "My principal investigator (PI) said I should continue the project of a previous postdoc, but that postdoc is long gone and hasn't saved any scripts or data".

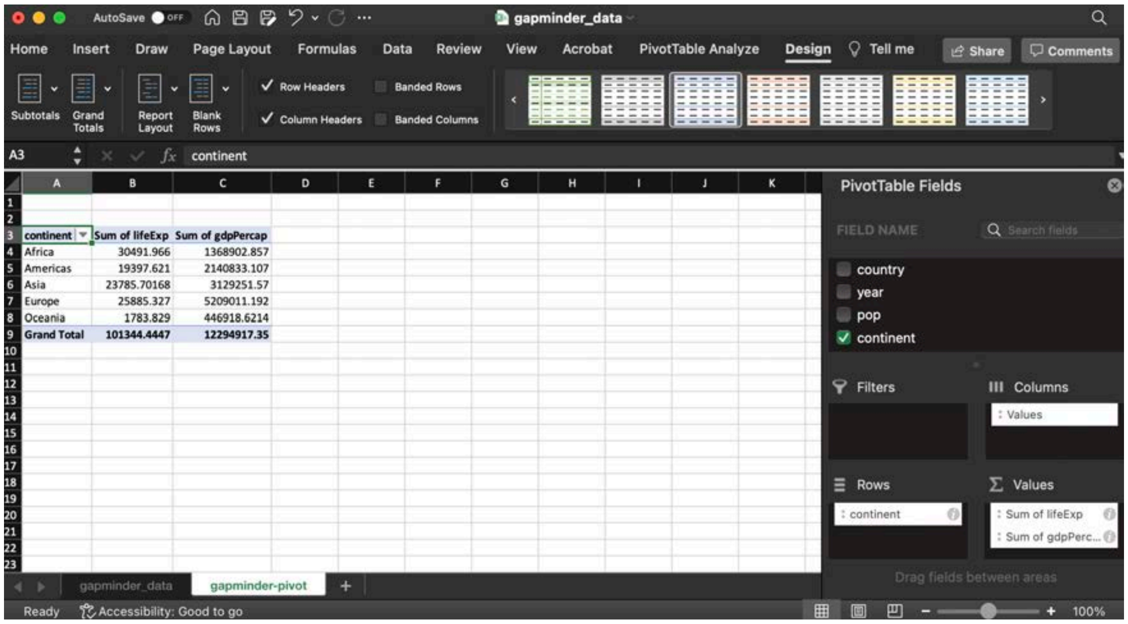
Helps to build your reputation

For several papers, we have made our data, code, and analyses available as an Experiment Package on Bioconductor. When I came up for tenure, I cited all these packages as research output of my lab. Generally, making your analyses available in this way will help you to build a reputation for being an honest and careful researcher.

Why Should I learn R

More Powerful

Why should I learn R when Pivot Tables do the same thing? Pivot Tables are great for summarizing the data. For example, Figure 4 is showing a Pivot Table for gapminder data.



continent	Sum of lifeExp	Sum of gdpPercap
Africa	30491.966	1368902.857
Americas	19397.621	2140833.107
Asia	23785.70168	3129251.57
Europe	25885.327	5209011.192
Oceania	1783.829	446918.6214
Grand Total	**101344.4447**	**12294917.35**

Figure 4: Pivot Table in Excel.

The only thing you can really do with a Pivot Table is get averages, maximum and minimum values. With R you can also group the data by and run one regression model that would go over all 145 countries. You can also display statistics like the coefficients or the r2 value, with 2 lines of code. I agree with Chemistry Cat, Excel is so basic.

Figure 5: Chemistry cat thinks Excel is basic.

Free and Open Source

This gets at a more philosophical reason for using R. Although the use of Excel is pretty pervasive, not everyone has access to or can afford Office 365 (Heiss, 2022).

Learning how to perform advanced analytics in Tableau or Stata is a great skillset. However, these products are very expensive.

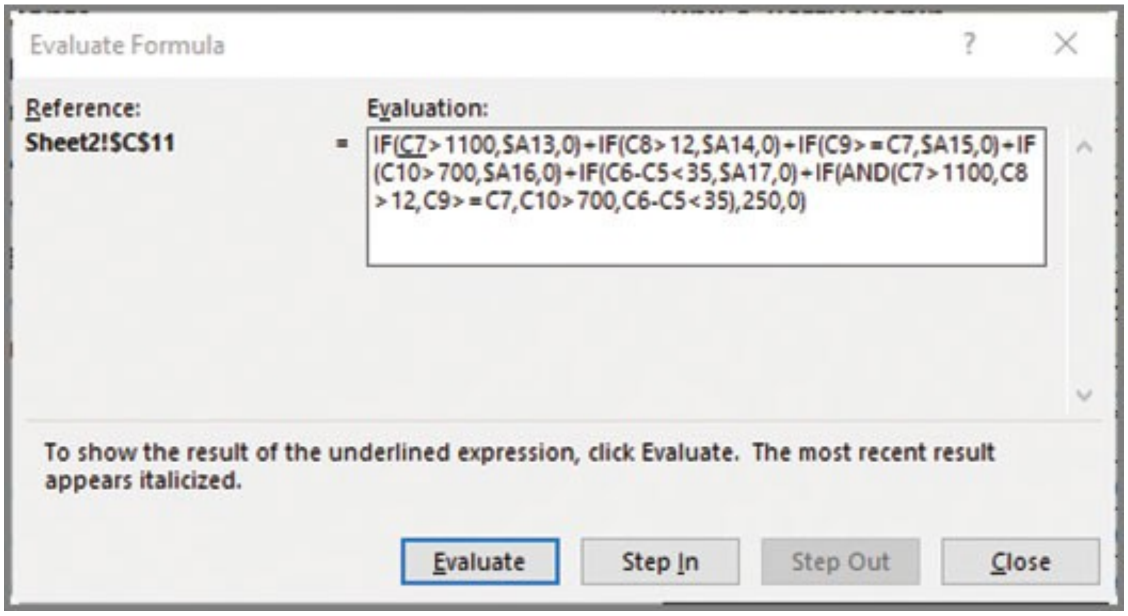
Also, if you are sharing the results of your analysis in Excel (don't do that), not everyone will be able to open that file.

Reproducibility

Figure 6 is displaying a complicated worksheet formula. The only way this would be reproducible is if you write down all of the steps for:

- Every menu you clicked
- Every cell you clicked on to add a formula or changed the formatting

Have you ever seen an Excel spreadsheet with an accompanying set of instructions?


Evaluate Formula

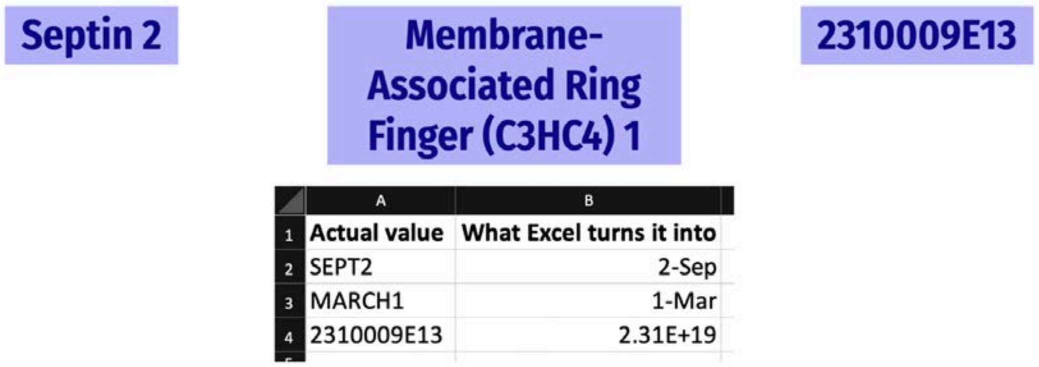
Reference: Sheet2!C11

Evaluation: = IF(C7>1100,$A13,0)+IF(C8>12,$A14,0)+IF(C9>=C7,$A15,0)+IF(C10>700,$A16,0)+IF(C6-C5<35,$A17,0)+IF(AND(C7>1100,C8>12,C9>=C7,C10>700,C6-C5<35),250,0)

To show the result of the underlined expression, click Evaluate. The most recent result appears italicized.

Figure 6: Complicated excel formula.

Figure 7 is displaying a common Excel formatting error that occurs when you are working with biological data.


Septin 2
Membrane-Associated Ring Finger (C3HC4) 1
2310009E13

	A	B
1	Actual value	What Excel turns it into
2	SEPT2	2-Sep
3	MARCH1	1-Mar
4	2310009E13	2.31E+19

Figure 7: Excel conversion error (Heiss, 2022).

A 2016 study by Ziemann, Eren, and EL-Osta found gene name errors in 20% of the papers that they reviewed (Ziemann et al., 2016). A quick search in Web of Science using the terms “Reproducible statistical analyses” OR “Reproducibility” resulted in over 7,000 published papers, in the last 10 years..

It is a great practice to not open any data files in Excel to prevent accidental conversion errors.

General Recommendations

From: (Heiss, 2022)

Don’t Touch the Raw Data

Don’t ever do analysis on the original data, or, if you do, then explain what you did to the data.

Self-documenting and Reproducible Code

More about this in the second part of this class. However, consider writing your reports or papers in markdown. Markdown combines text with code.

Use Open Formats

Use open formats as much as possible. That means sharing your data in csv or tab-delimited format.

A lot of large companies and government organizations use markdown for their publishing and research.

Using Version Control

Using version control has the power to revolutionize how you work in R. Being able to have a record of all the changes you’ve ever made to your code both locally and on a remote website is powerful. But getting everything set up can be a challenge. in this section I will walk through everything you need to do to use Git and GitHub alongside RStudio.

Git Versus GitHub

1. [Git] (https://en.wikipedia.org/wiki/Git “Opens in a new window”) is open source software for version control. Using Git, you can do things like see all previous versions of code you’ve ever created in a project.

2. GitHub is the most popular service for collaborating on code using Git

3. GitLab and BitBucket are two other options

It is possible to use Git without using GitHub, though most people combine the two.

Why Should I Use Version Control for My Project?

There are many reasons you would want to adopt a Git/GitHub workflow:

1. **Using Git and GitHub serves as a backup**. Because GitHub has a copy of all the code you have locally, if anything were to happen to your computer, you’d still have access to your code.

2. **Using Git and GitHub allows you to use version control.** Ever had documents called report-final.pdf, report-final-v2.pdf, and report-final-v3.pdf? Of course, you have. Instead of making copies of files over fear of losing work, version control allows you to see what you did in the past, all while keeping single versions of documents.

3. **Using Git and GitHub makes it possible to work on the same project at the same time as collaborators.** For many using Git/GitHub is better than collaborating using Dropbox, Google Drive, or OneDrive. Git and GitHub have built-in tools that enable simultaneous asynchronous work, a major benefit for those working in teams.

You might want to check out this 2016 talk by Alice Bartlett of the Financial Times, on Git and GitHub. Another great resources (one that I used for this section) can be found on **Happy Git and GitHub for the useR**

About Authentication to GitHub

When we interact with a remote Git server, such as GitHub, we must include credentials in the request. This proves we are a specific GitHub user, who's allowed to do whatever we're asking to do.

You can access your resources in GitHub in a variety of ways: in the browser, via GitHub Desktop or another desktop application, with the API, or via the command line. Each way of accessing GitHub supports different modes of authentication.

- Username and password with two-factor authentication
- Personal access token
- HTTPS
- SSH key

Authenticating in your browser

You will authenticate using your GitHub.com username and password. You may also use two-factor authentication and SAML single sign-on, which can be required by organization and enterprise owners. The help documentation for this topic on GitHub is an excellent resource.

Personal access token for HTTPS

With HTTPS, we will use a **personal access token (PAT)**. Head over to chapter 10 if you really want to set up SSH keys.

Let it be known that the password that you use to login to GitHub's website is NOT an acceptable credential when talking to GitHub as a Git server. This was possible in the past, but those days are over at GitHub. You can learn more in their blog post Token authentication requirements for Git operations.

Here's the error you'll see if you try to do that now:

```
remote: Support for password authentication was removed on August 13, 2021. P
lease use a personal access token instead.
remote: Please see https://github.blog/2020-12-15-token-authentication-requir
ements-for-git-operations/ for more information.
fatal: Authentication failed for 'https://github.com/OWNER/REPO.git/'
```

Creating your PAT and Connecting it to Git

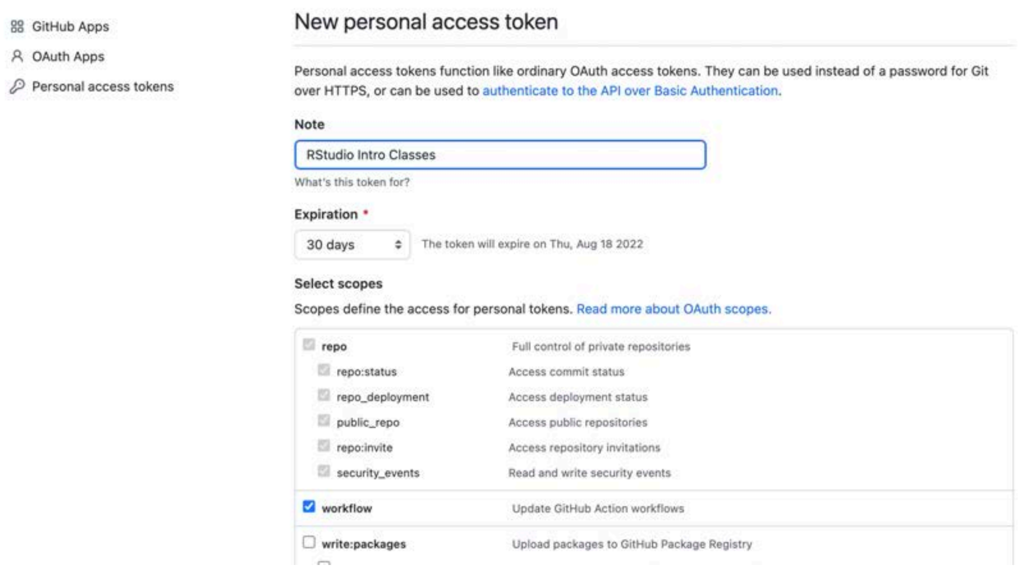
This is a very minimal account of getting and storing a PAT. This might be all you need when you're first getting yourself set up. You can always come back later and read other parts of this chapter.

1. Go to https://github.com/settings/tokens and click "Generate token".

2. Or you can do this from the R Terminal by using the usethis package. `usethis` is a workflow package: it automates repetitive tasks that arise during project setup and development, both for R packages and non-package projects.

```
# install.packages("usethis")
library(usethis)
create_github_token()

• Call `gitcreds::gitcreds_set()` to register this token in the local Git cre
dential store
  It is also a great idea to store this token in any password-management soft
ware that you use
• Open URL 'https://github.com/settings/tokens/new?scopes=repo,user,gist,work
flow&description=DESCRIBE THE TOKEN\'S USE CASE'
```

3. When you run this section of code your default browser will open and you will be prompted to log into GitHub. After you log in you will automatically go to the create repository page [Figure 4].


GitHub Apps
OAuth Apps
Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

RStudio Intro Classes

What's this token for?

Expiration *

30 days — The token will expire on Thu, Aug 18 2022

Select scopes

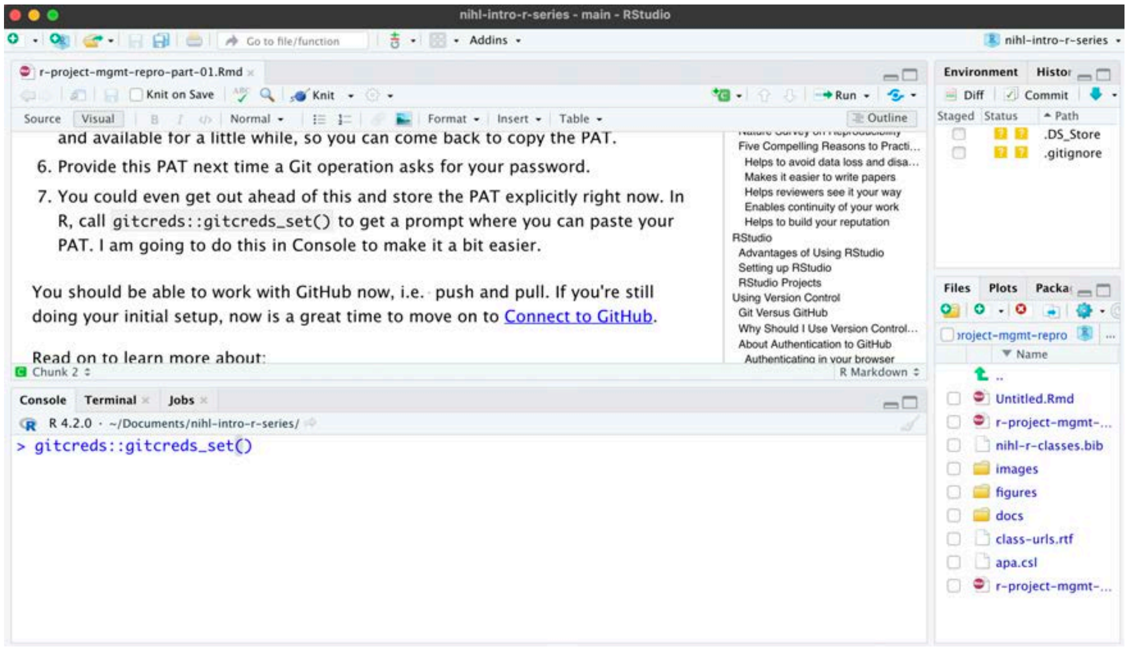
Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events
✓ workflow	Update GitHub Action workflows
☐ write:packages	Upload packages to GitHub Package Registry

Figure 4: Creating a new PAT on GitHub.

Look over the scopes; I highly recommend selecting “repo”, “user”, and “workflow”. Recommended scopes will be pre-selected if you used `create_github_token()`.

4. Click “Generate token”.
5. Copy the generated PAT to your clipboard. Or leave that browser window open and available for a little while, so you can come back to copy the PAT.
6. Provide this PAT next time a Git operation asks for your password.
7. You could even get out ahead of this and store the PAT explicitly right now. In R, call `gitcreds::gitcreds_set()` to get a prompt where you can paste your PAT. I am going to do this in Console to make it a bit easier.
8. Figure 5 is displaying the `gitcreds::gitcreds_set()` in the Console. Hit Enter/Return to run this command.


nihl-intro-r-series - main - RStudio
r-project-mgmt-repro-part-01.Rmd
Knit on Save
Knit
Run
Source
Visual
Normal
Format
Insert
Table
Outline
Five Compelling Reasons to Practi…
Helps to avoid data loss and disa…
Makes it easier to write papers
Helps reviewers see it your way
Enables continuity of your work
Helps to build your reputation
RStudio
Advantages of Using RStudio
Setting up RStudio
RStudio Projects
Using Version Control
Git Versus GitHub
Why Should I Use Version Control…
About Authentication to GitHub
Authenticating in your browser
Chunk 2
R Markdown
Environment
History
Diff
Commit
Staged
Status
Path
.DS_Store
.gitignore
Files
Plots
Packages
project-mgmt-repro
Name
Untitled.Rmd
r-project-mgmt-…
nihl-r-classes.bib
images
figures
docs
class-urls.rtf
apa.csl
r-project-mgmt-…
Console
Terminal
Jobs
R 4.2.0 · ~/Documents/nihl-intro-r-series/

and available for a little while, so you can come back to copy the PAT.

6. Provide this PAT next time a Git operation asks for your password.

7. You could even get out ahead of this and store the PAT explicitly right now. In R, call `gitcreds::gitcreds_set()` to get a prompt where you can paste your PAT. I am going to do this in Console to make it a bit easier.

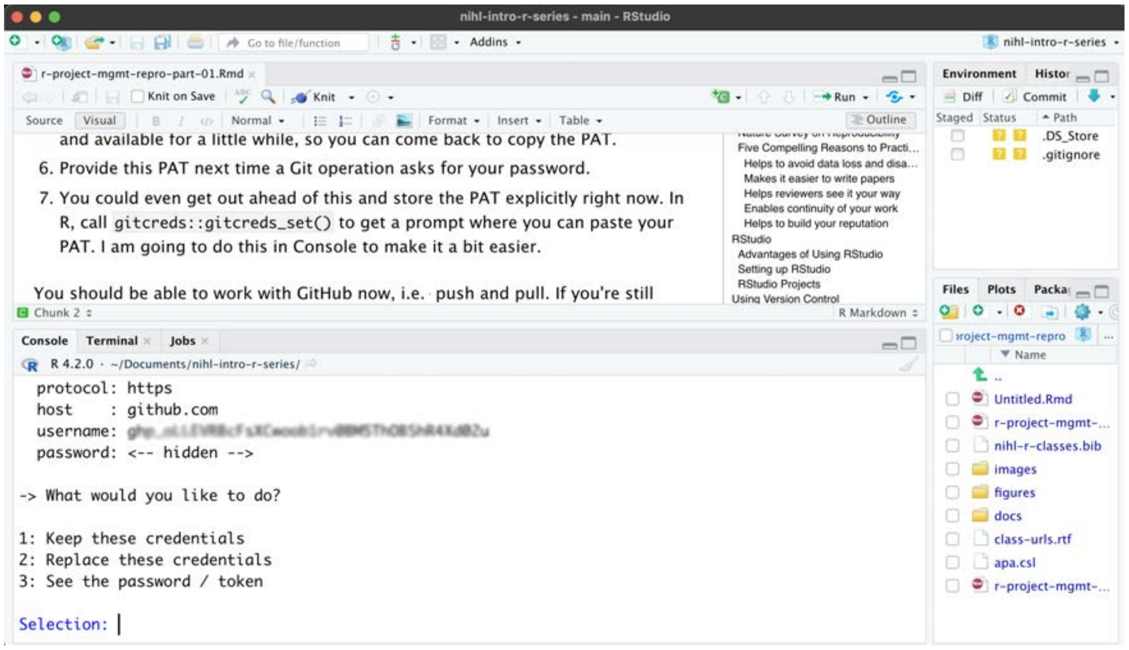
You should be able to work with GitHub now, i.e. push and pull. If you're still doing your initial setup, now is a great time to move on to Connect to GitHub.

Read on to learn more about:

```
> gitcreds::gitcreds_set()
```

Figure 5: Initiating the gitcreds_set() command.

9. Select the most appropriate option. Since we are setting this up for the first time, we would select option 2 [Figure 6].



and available for a little while, so you can come back to copy the PAT.

6. Provide this PAT next time a Git operation asks for your password.

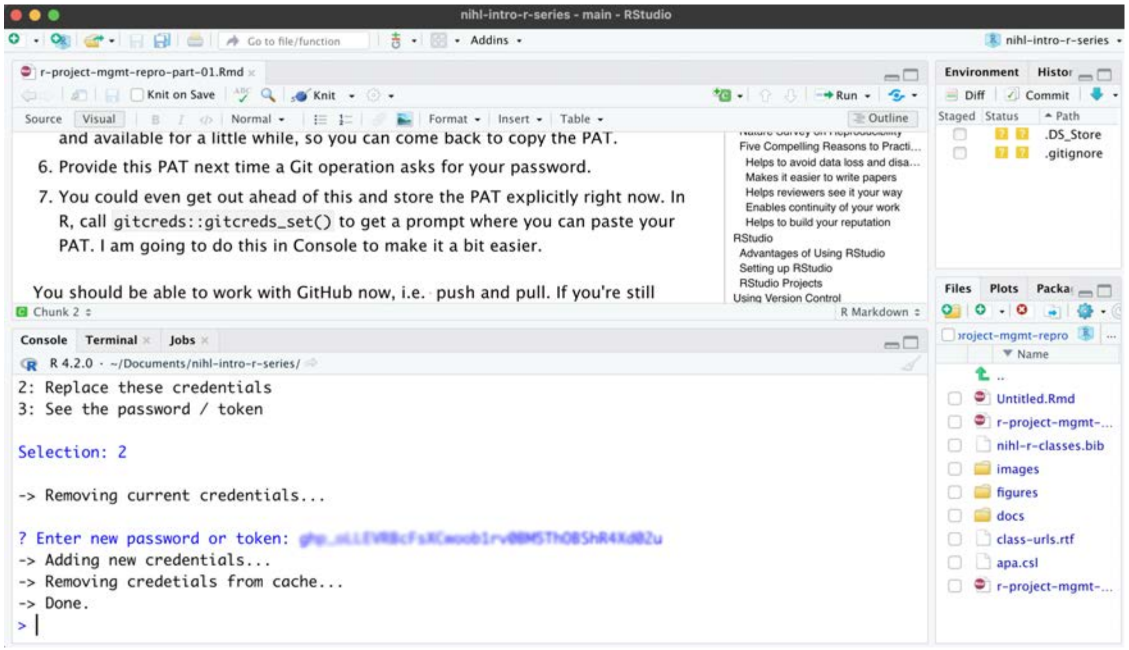
7. You could even get out ahead of this and store the PAT explicitly right now. In R, call `gitcreds::gitcreds_set()` to get a prompt where you can paste your PAT. I am going to do this in Console to make it a bit easier.

You should be able to work with GitHub now, i.e. push and pull. If you're still

```
  protocol: https
  host    : github.com
  username: [redacted]
  password: <-- hidden -->

-> What would you like to do?

1: Keep these credentials
2: Replace these credentials
3: See the password / token

Selection: 
```

Figure 6: Replacing the current PAT.

10. Follow the prompts and enter the PAT that you just created.



and available for a little while, so you can come back to copy the PAT.

6. Provide this PAT next time a Git operation asks for your password.

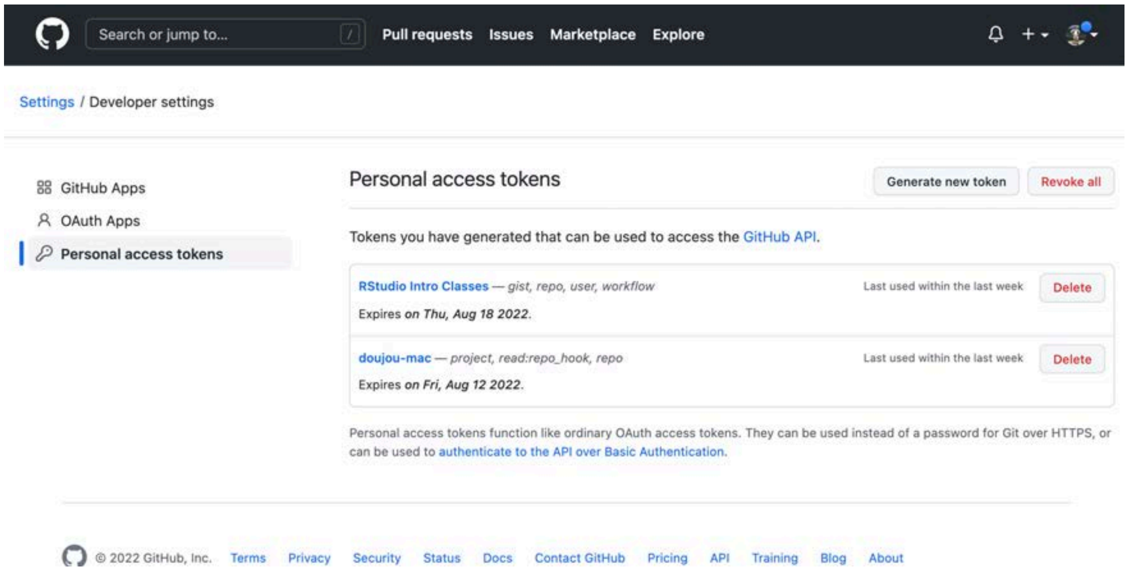
7. You could even get out ahead of this and store the PAT explicitly right now. In R, call `gitcreds::gitcreds_set()` to get a prompt where you can paste your PAT. I am going to do this in Console to make it a bit easier.

You should be able to work with GitHub now, i.e. push and pull. If you're still

```
R 4.2.0 · ~/Documents/nihl-intro-r-series/
2: Replace these credentials
3: See the password / token

Selection: 2

-> Removing current credentials...

? Enter new password or token: [redacted]
-> Adding new credentials...
-> Removing credetials from cache...
-> Done.
>
```

Figure 7: Final output from storing the PAT.

11. If you go back to GitHub you should see a list of all of the PATs that you have created and when they expire [Figure 8].



Settings / Developer settings

Personal access tokens

Tokens you have generated that can be used to access the GitHub API.

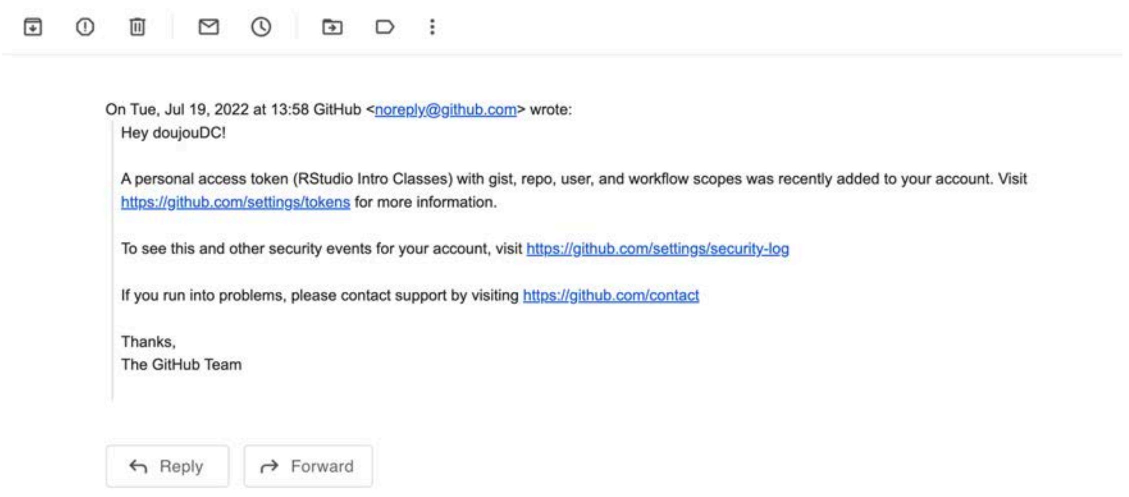
RStudio Intro Classes — *gist, repo, user, workflow* Expires on *Thu, Aug 18 2022*.	Last used within the last week	Delete
doujou-mac — *project, read:repo_hook, repo* Expires on *Fri, Aug 12 2022*.	Last used within the last week	Delete

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

© 2022 GitHub, Inc.

Figure 8: List of GitHub PATs.

You will also receive a message from GitHub (default email address) letting you know that a PAT has been created [Figure 9].


On Tue, Jul 19, 2022 at 13:58 GitHub <noreply@github.com> wrote:

Hey doujouDC!

A personal access token (RStudio Intro Classes) with gist, repo, user, and workflow scopes was recently added to your account. Visit https://github.com/settings/tokens for more information.

To see this and other security events for your account, visit https://github.com/settings/security-log

If you run into problems, please contact support by visiting https://github.com/contact

Thanks,
The GitHub Team

Figure 9: PAT alert email from GitHub.

You should be able to work with GitHub now, i.e. push and pull.

Remember, if you are already syncing on another computer, you will have to enter this new PAT before you push or pull.

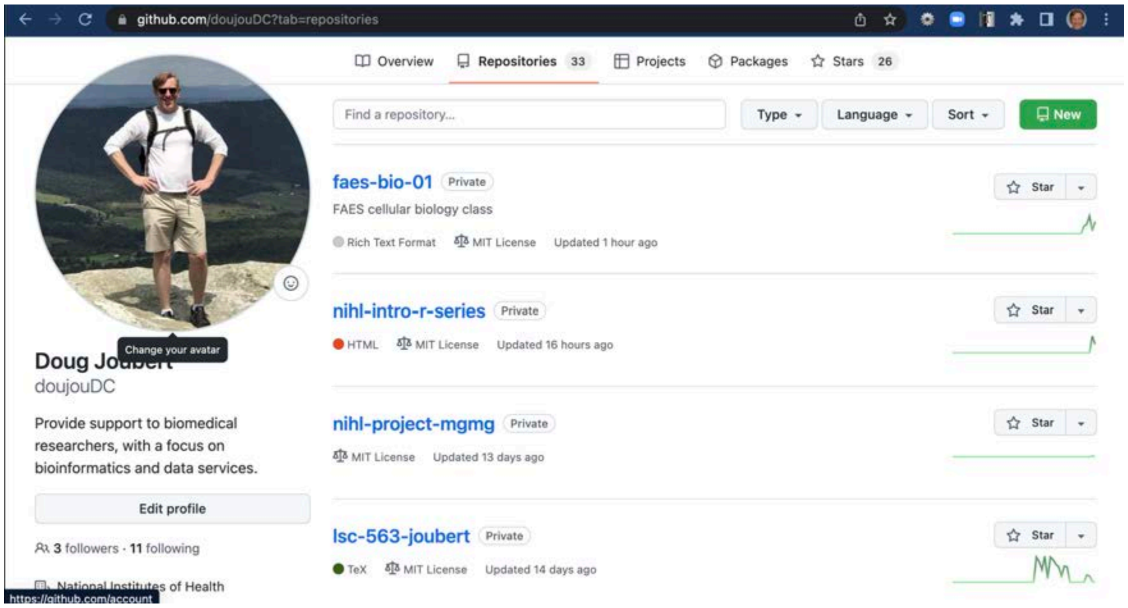
Read on to learn more about:

- How to decide between the HTTPS and SSH protocols
- PAT scopes, names, and expiration
- PAT storage
- Troubleshooting

Creating a New Repository in GitHub

GitHub's collaborative approach to development depends on publishing commits from your local repository to GitHub for other people to view, fetch, and update.

At this link, you may sign into your GitHub account or create one if you have not already. Figure 10 is displaying the repositories in my GitHub account.



github.com/doujouDC?tab=repositories

Overview | Repositories 33 | Projects | Packages | Stars 26

Find a repository... | Type | Language | Sort | New

faes-bio-01 Private
FAES cellular biology class
Rich Text Format · MIT License · Updated 1 hour ago

nihl-intro-r-series Private
HTML · MIT License · Updated 16 hours ago

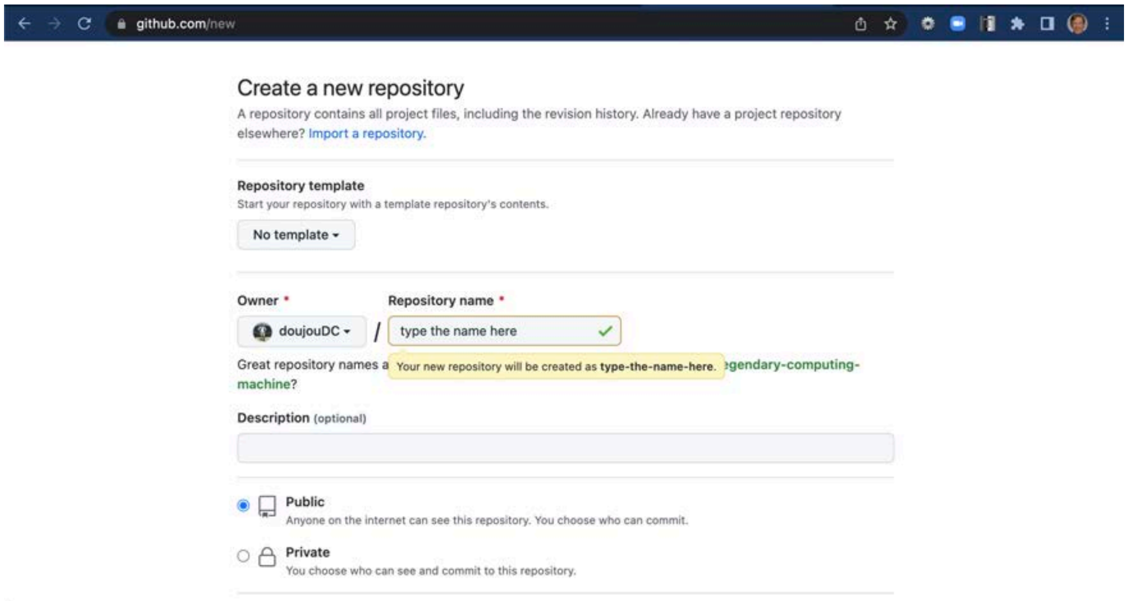
nihl-project-mgmg Private
MIT License · Updated 13 days ago

lsc-563-joubert Private
TeX · MIT License · Updated 14 days ago

Doug Joubert
doujouDC
Provide support to biomedical researchers, with a focus on bioinformatics and data services.
Edit profile
3 followers · 11 following
National Institutes of Health

Figure 10: My repository homepage in GitHub.

I have already created a repository for this class. However, Figure 11 is displaying the Create a New Repository screen. [Figure 11].



github.com/new

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 * doujouDC / Repository name * type the name here
Your new repository will be created as type-the-name-here.
Great repository names a... ...legendary-computing-machine?

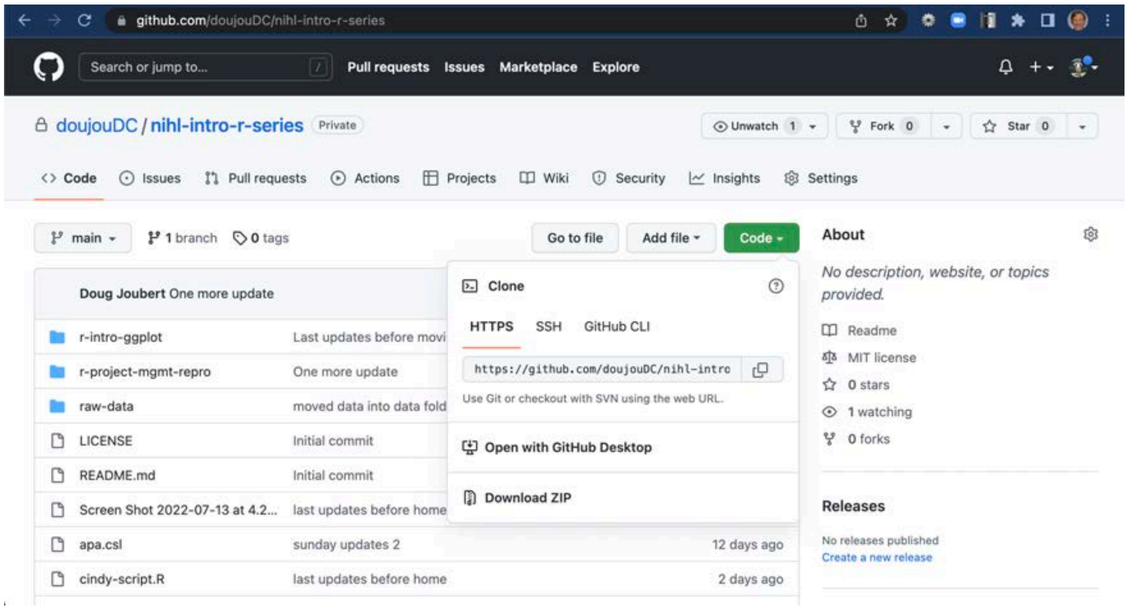
Description (optional)

● 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.

Figure 11: Creating a new repository on GitHub.

Once your repository is created, you will need to copy the repository URL before you create the project in RStudio. Figure 11 shows the 3 different methods for creating a URL: (1) https, (2) SSH, or (3) GitHub CLI. We have already created our PAT to use with the https option [Figure 12].



Doug Joubert One more update

r-intro-ggplot	Last updates before movi	
r-project-mgmt-repro	One more update	
raw-data	moved data into data fold	
LICENSE	Initial commit	
README.md	Initial commit	
Screen Shot 2022-07-13 at 4.2...	last updates before home	
apa.csl	sunday updates 2	12 days ago
cindy-script.R	last updates before home	2 days ago

```
https://github.com/doujouDC/nihl-intro
```

Figure 12: Creating a link to a GitHub repository.

RStudio

RStudio is an integrated development environment (IDE) for R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging, collaboration, and workspace management. It is a powerful tool which supports research by weaving the principles of reproducibility throughout the entire research lifecycle, from data gathering to the statistical analysis, presentation, and publication of results.

Advantages of Using RStudio

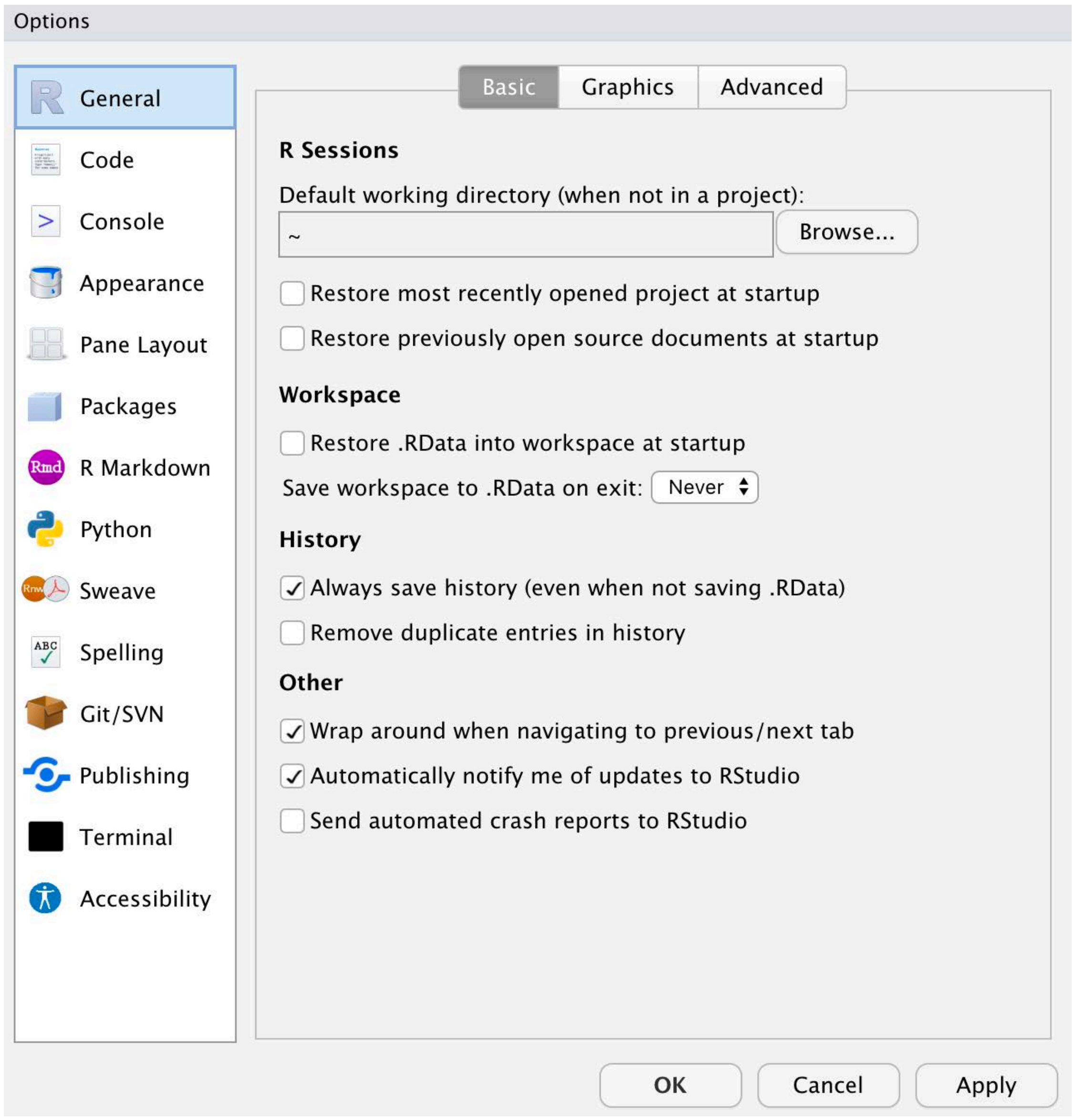
- It is free and open-source
- It is designed to make it easy to write and reuse code
- Makes it convenient to view and interact with the objects stored in your environment
- Makes it easy to set your working directory and access files on your computer
- Integrates with Collaboration and Publishing Tools
- Creates documents using R Markdown

Setting up RStudio

There are a number of RStudio settings that are recommended. These settings will force RStudio to start "fresh" each time you open RStudio.

This means that RStudio will not remember the code that you ran in your previous session. This forces you to document your work in your code and not in the RStudio workspace [which is temporary].

Figure 13 list the preferred RStudio settings. To change these options, go to **Tools>Global Options>General.**


Options

General
Code
Console
Appearance
Pane Layout
Packages
R Markdown
Python
Sweave
Spelling
Git/SVN
Publishing
Terminal
Accessibility

Basic | Graphics | Advanced

R Sessions

Default working directory (when not in a project):

~ Browse...

☐ Restore most recently opened project at startup

☐ Restore previously open source documents at startup

Workspace

☐ Restore .RData into workspace at startup

Save workspace to .RData on exit: Never

History

☑ Always save history (even when not saving .RData)

☐ Remove duplicate entries in history

Other

☑ Wrap around when navigating to previous/next tab

☑ Automatically notify me of updates to RStudio

☐ Send automated crash reports to RStudio

OK | Cancel | Apply

Figure 13: Preferred RStudio options.

You might want to also choose UTF-8 as the default text encoding. This setting is available from **Tools>Global Options>Code>Saving**.

RStudio Projects

An RStudio Project provides a set of tools that allow you to organize your scripts, files, and output:

1. Allows you to quickly navigate to your working directory

2. Is a great way to keep all your data organized

3. Can preserve custom settings and open files to make it easier to resume work after a break.

An RStudio project is different from an R session. Meaning that an R session is all the objects and work done in R. Sessions are usually kept in working memory, until you restart R.

There are three options for creating a project in RStudio:

1. New Directory - start a brand new R project (with the option of version control).
2. Existing Directory - add existing work to a R project (with the option of setting up version control).
3. Version Control Continue an existing R project that already uses version control (i.e. download from GitHub).

We will be using option 3, creating a repository on GitHub, and then importing this into RStudio.

Creating a New Project in RStudio

1. Open RStudio
2. Choose New Project from the Tools Menu.
3. There are three options when creating a new project: (1) Creating a project in a new directory, (2) Creating a project in an existing directory, and (3) Creating a project using version control [Figure 14].

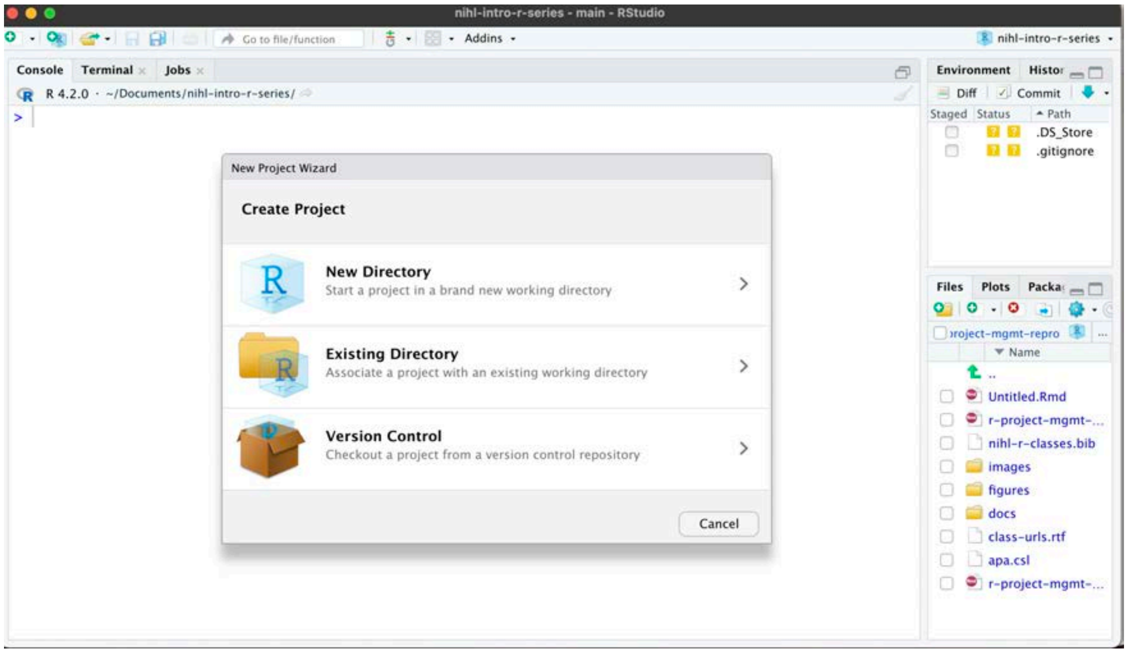


Figure 14: Creating a new project in RStudio.

4. Choose the Version Control option
5. Chose Git as the option, RStudio will automatically open the next screen [Figure 15]

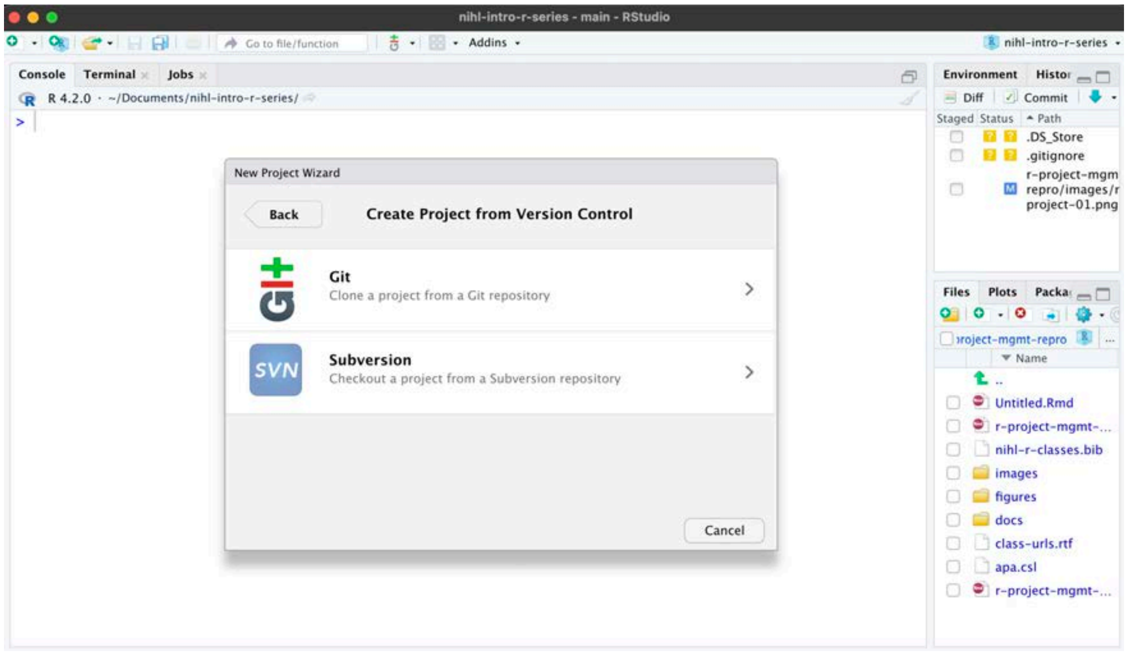

nihl-intro-r-series - main - RStudio
New Project Wizard
Back
Create Project from Version Control
Git
Clone a project from a Git repository
Subversion
Checkout a project from a Subversion repository
Cancel

Figure 15: Choosing the version control option in RStudio.

6. This screen is where you enter the repository URL that you copied from GitHub [Figure 16].

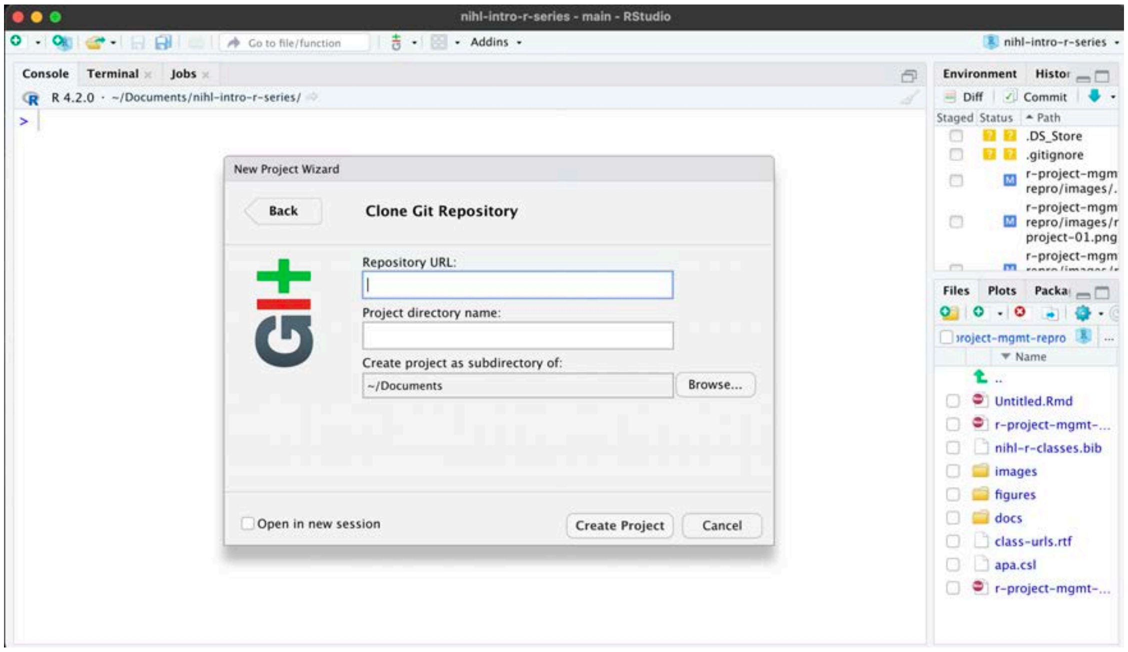


Figure 16: New project wizard - entering the repository URL.

7. Open up the repository that we created in GitHub and copy the `https` URL [Figure 17].

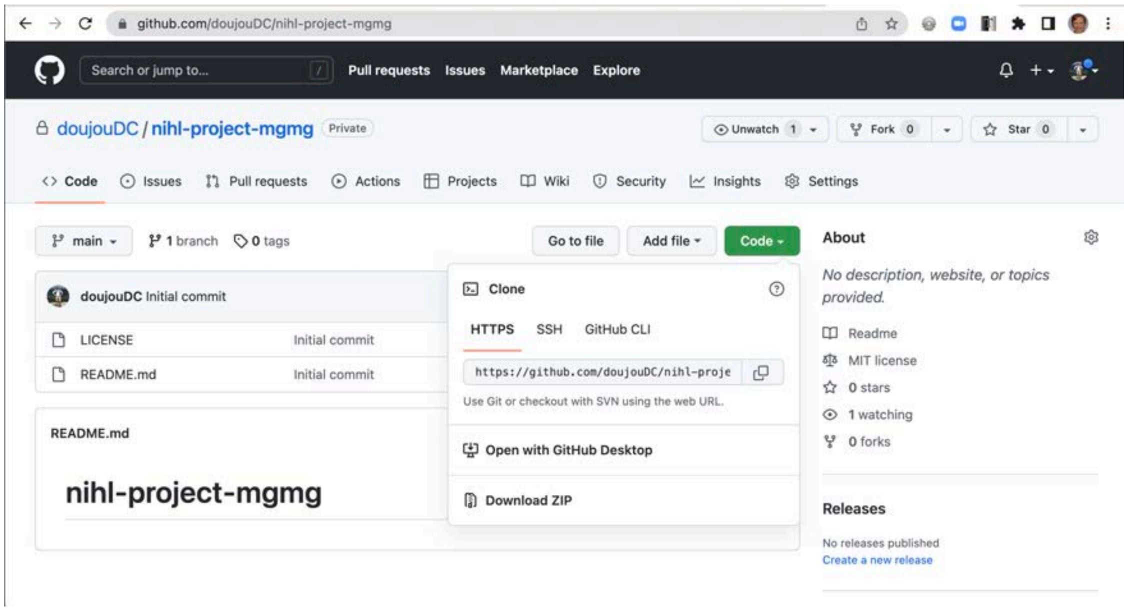


Figure 17: Copying a `https` repository URL from GitHub.

8. Paste the repository URL into the URL box. Once you do this, you will be prompted to choose a subdirectory for the new project. I usually created mine within my Documents directory.

9. Click **Create Project**. The screen will refresh as RStudio switches over to project view and downloads data from GitHub

Good Practices for Managing Projects in RStudio

Project Organization Issues

We often have stress points in our research that may become breaking points, especially when it comes to working with collaborators, returning to a project after a hiatus, or dealing with data and scripts. Let's discuss three of those common stress points:

File/folder disorganization

- You cannot find your files on your computer (or your cloud storage)
- Multiple versions of files with names such as "finaldraft_4.txt"
- Path issues when trying to run code
- Reviewers or colleagues cannot re-run your code/analyses

Storage and sharing issues

- Files are only saved to your computer and are vulnerable (or have already succumbed to computer/hard drive failure
- When working with collaborators, they (or you) don't share the files needed
- Files are shared via email attachments

- Difficult to know if you have the latest version of documents

Losing track of project status

- You cannot remember where you are in a project after being away for an extended period (or what you worked on the previous day…no judgement)
- You aren't sure what you should be working on next
- You have various to-do notes spread across your office or home (or never write them down in the first place)

Discussion Points

To what extent do these stress points affect your research projects?

Are there additional issues that you've encountered that slow down or derail your work due to issues with project management?

Practice Good File Organization

Although there is no "best" way to lay out a project, there are some general principles to adhere to that will make project management easier.

Good Enough Practices for Scientific Computing gives the following recommendations for project organization (Wilson et al., 2017):

1. Each project in its own directory, which is named after the project
2. Text documents associated with the project in the doc directory
3. Raw data and metadata in the data directory (raw-data)
4. Files generated during cleanup and analysis in a results directory
5. Project's scripts and programs in the src directory
6. Programs brought in from elsewhere or compiled locally in the bin directory
7. Name all files to reflect their content or function

Additional filles to include:

README file, to communicate important information about your project LICENSE file, so that others are free to use, change, and distribute the software CITATION.cff file, to let others know how you would like them to cite your work

Additional recommendations for projects and creating folders is covered in the Introduction to R and RStudio class. Class handouts are available upon request.

Practices for Naming Files

Three principles for file names 1. Machine readable 2. Human readable 3. Plays well with default ordering

Why This is important

- Globbing: using wildcard characters to request or evaluate sets of files with the same partial names or sets of character
- Regular expression friendly
- “Findability”
- Global name changes

Machine-readable Files

- No spaces, unsupported punctuation, accented characters, or case-sensitive file names
- Deliberate use of delimiters (i.e. for splitting file names)
- Consistently use the same delimiters: `data-analyses-fig1.R` as an example

Human-readable

- Name contains brief description of content `anova-analyses-control.R` `marshall-data-raw-data` `marshall-data-filtered` `marshall-data-load-counts.r`

Ordering Files

With **chronological ordering**, file name starts with date:

2022-06-26-BRAFWTNEGASSAY-Plasmid-Cellline-100-MutantFraction_H01.csv

2022-06-26-BRAFWTNEGASSAY-Plasmid-Cellline-100-MutantFraction_H02.csv

2022-06-26-BRAFWTNEGASSAY-Plasmid-Cellline-100-MutantFraction_H03.csv

2022-06-26-BRAFWTNEGASSAY-Plasmid-CeIIIine-100-MutantFraction-platefile.csv

2022-02-26-BRAFWTNEGASSAY-FFPEDNA-CRC-1-41_A01.csv

2022-02-26-BRAFWTNEGASSAY-FFPEDNA-CRC-1-41_A02.csv

2022-02-26-BRAFWTNEGASSAY-FFPEDNA-CRC-1-41_A03.csv

2022-02-26-BRAFWTNEGASSAY-FFPEDNA-CRC-1-41_A04.csv

Consider using ISO 8601 date standard

With **logical ordering**, the filename starts with a number or keyword/number combo.

01-marshall-data.r *see code directory*

02-pre-dea-filtering.r *see code directory*

03-explore-dea-limma-voom.r

04-exploe-dea-results.r helper

01-load-counts.rmd

helper02-load-exp-des.r

helper03-extract-and-tidy.r

As illustrated above, left-pad your numbers to facilite sorting. If you do not do this, your data sorts like this…which is really sad

01-marshall-data.r *see code directory*

04-exploe-dea-results.r

2-explore-dea-limma-voom.r

3-helper-extract-and-tidy.r

helper01-load-counts.rmd

helper2-load-exp-des.r

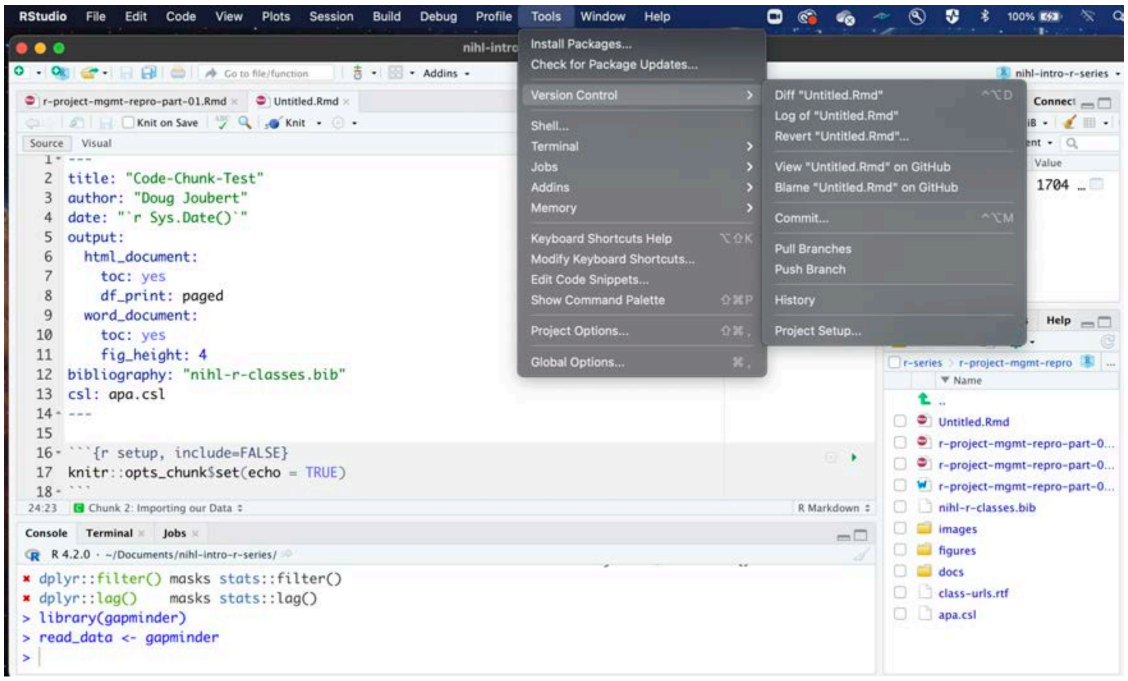
pre-dea-filtering-1.r *see code directory*

Adapted from https://datacarpentry.org/rr-organization1/01-file-naming/index.html. For more tips on file naming, check: The Dos and Don'ts of File Naming.

Using Version Control in RStudio

There are two places we can interact with Git in the RStudio interface.

1. Menu bar [Figure 18]

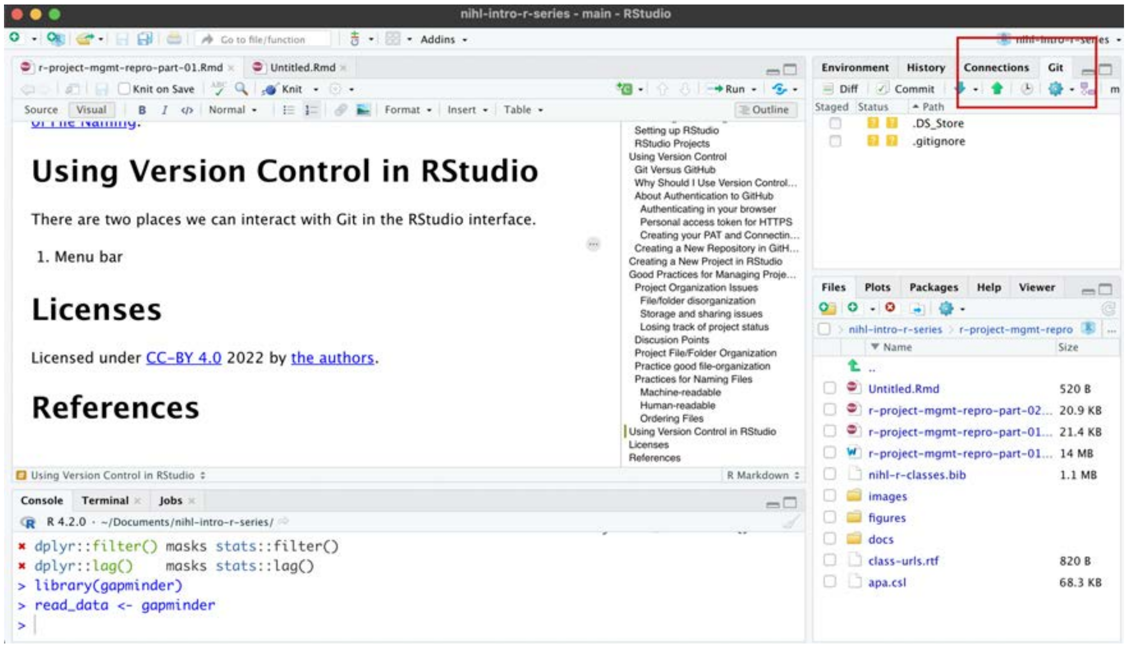


~~~
---
title: "Code-Chunk-Test"
author: "Doug Joubert"
date: "`r Sys.Date()`"
output:
  html_document:
    toc: yes
    df_print: paged
  word_document:
    toc: yes
    fig_height: 4
bibliography: "nihl-r-classes.bib"
csl: apa.csl
---

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

```
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
> library(gapminder)
> read_data <- gapminder
>
```

Figure 18: Menu bar options for working with GIT

2. Git Panel [Figure 19]



Using Version Control in RStudio

There are two places we can interact with Git in the RStudio interface.

1. Menu bar

Licenses

Licensed under CC-BY 4.0 2022 by the authors.

References

```
R 4.2.0 · ~/Documents/nihl-intro-r-series/
× dplyr::filter() masks stats::filter()
× dplyr::lag()    masks stats::lag()
> library(gapminder)
> read_data <- gapminder
>
```

Name	Size
Untitled.Rmd	520 B
r-project-mgmt-repro-part-02...	20.9 KB
r-project-mgmt-repro-part-01...	21.4 KB
r-project-mgmt-repro-part-01...	14 MB
nihl-r-classes.bib	1.1 MB
images	
figures	
docs	
class-urls.rtf	820 B
apa.csl	68.3 KB

Figure 19: Git panel

It is not possible to cover everything you need to successfully use GIT. There are many excellent resources for Git.

Workflows in Git

Git workflows are probably not that different from what you do currently. For example, writing R scripts or authoring reports in LaTeX or R Markdown. But instead of only *saving* individual files, with Git you periodically **commit** your work. This means that you take a snapshot of all the files in the entire project. If you have ever versioned a file by adding your initials or the date, you have effectively made a commit, albeit only for a single file.

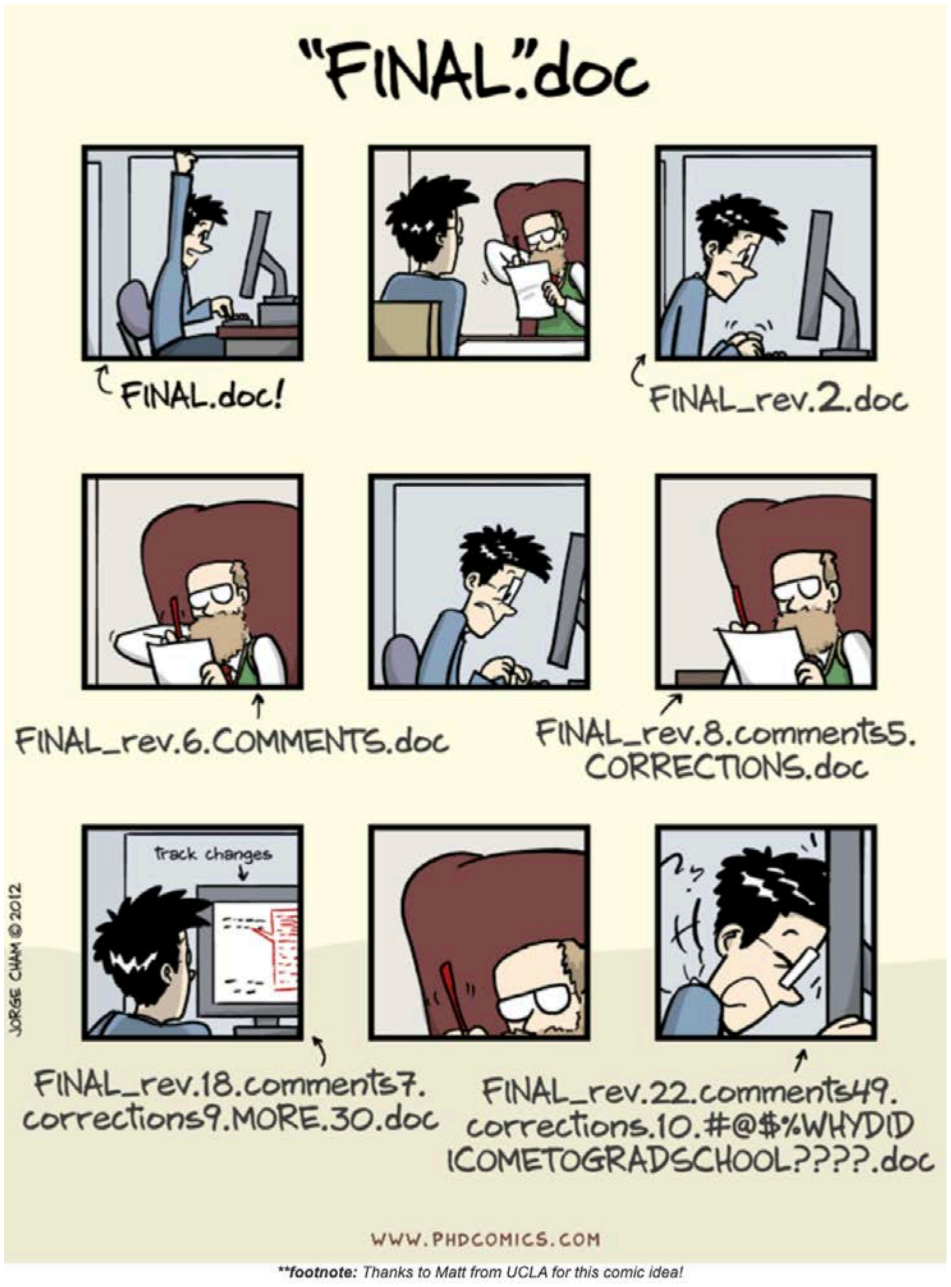


Figure 20: A humorous look at version control.

This is like sharing a document with colleagues on DropBox or sending it out as an email attachment. By pushing to GitHub, you make your work and all your accumulated progress accessible to others.

Commits and diffs

We now connect the fundamental concepts of Git to the data science workflow:

- repository

- commit
- diff

Recall that a repository or repo is just a directory of files that Git manages holistically. A commit functions like a snapshot of all the files in the repo, at a specific moment. Figure 21 is a fictional analysis of the iris data, focusing on the evolution of a script, `iris.R`.

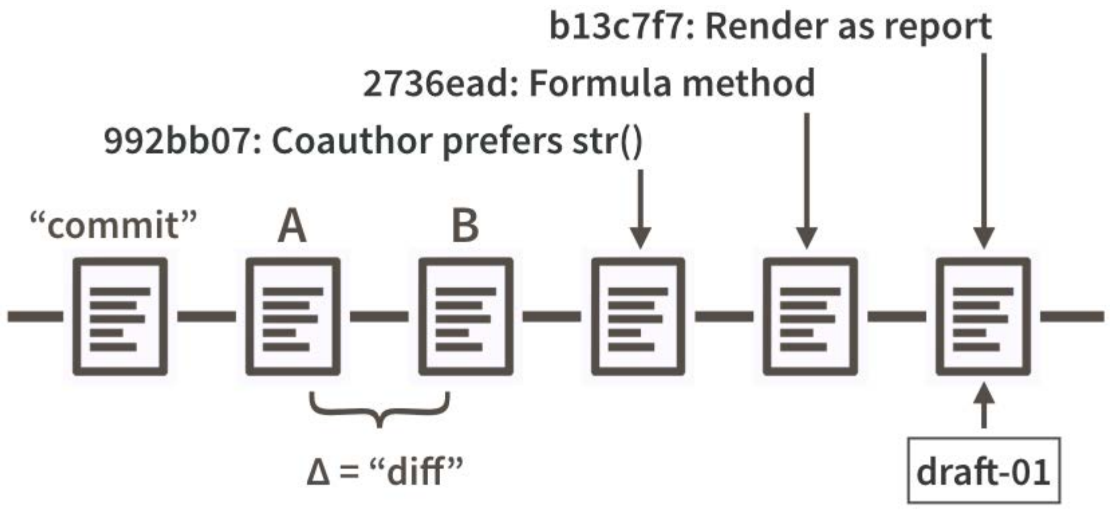
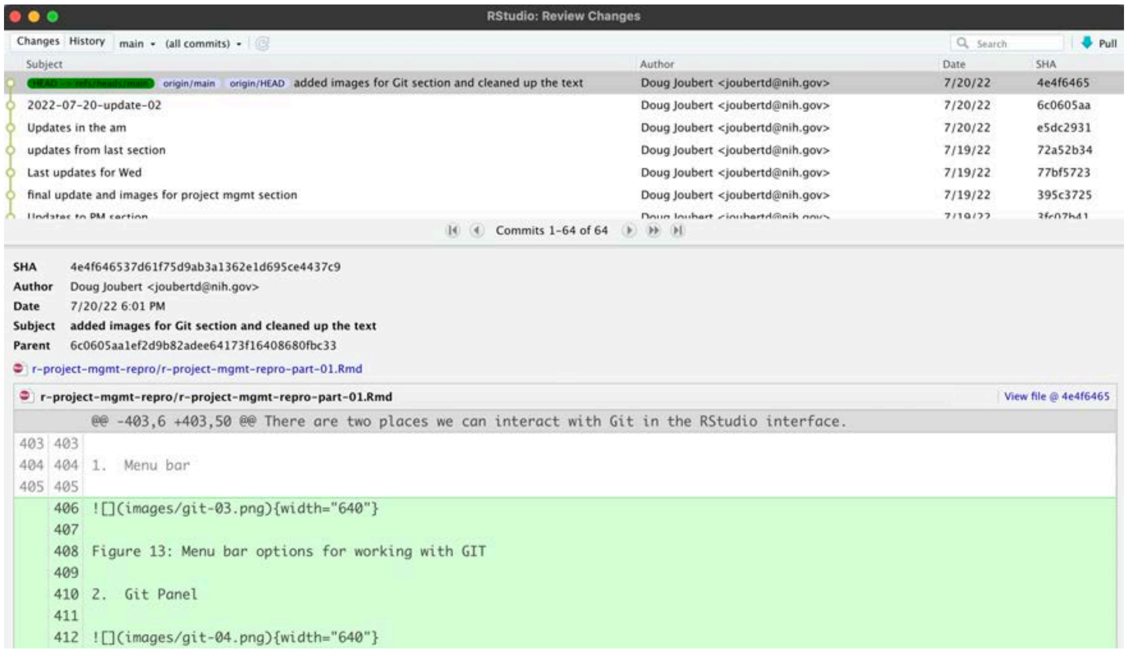


Figure 21: Partial commit history that looks at diffs, commit messages, SHA, and tags.

So what is going on in Figure 21. There are two versions of our document. Version A of this file and a modified version, version B. In this example, let us assume that version A was part of one Git commit and version B was part of the next commit. The set of differences between A and B is called a "diff."

Every time you make a commit you must also write a short **commit message**. Ideally, this conveys the motivation for the change. When you revisit a project after a break or need to digest recent changes made by a colleague, looking at the **history**, by reading commit messages and skimming through diffs, is an extremely efficient way to get up to speed. Figure 21 shows the messages associated with the last three commits.

Every commit needs some sort of nickname, so you can identify it. Git does this automatically, assigning each commit with an SHA, a seemingly random string of 40 letters and numbers (it is not, in fact, random). Though you will be exposed to these, you don't have to handle them directly very often and, when you do, usually the first 7 characters suffice. Figure 17 is showing the history for the training manual that I created for this class. You will notice the SHA column in Figure 22.



RStudio: Review Changes

Subject	Author	Date	SHA
HEAD -> refs/heads/main origin/main origin/HEAD added images for Git section and cleaned up the text	Doug Joubert <joubertd@nih.gov>	7/20/22	4e4f6465
2022-07-20-update-02	Doug Joubert <joubertd@nih.gov>	7/20/22	6c0605aa
Updates in the am	Doug Joubert <joubertd@nih.gov>	7/20/22	e5dc2931
updates from last section	Doug Joubert <joubertd@nih.gov>	7/19/22	72a52b34
Last updates for Wed	Doug Joubert <joubertd@nih.gov>	7/19/22	77bf5723
final update and images for project mgmt section	Doug Joubert <joubertd@nih.gov>	7/19/22	395c3725
Updates to PM section	Doug Joubert <joubertd@nih.gov>	7/19/22	3fc07b41

Commits 1-64 of 64

SHA 4e4f646537d61f75d9ab3a1362e1d695ce4437c9
Author Doug Joubert <joubertd@nih.gov>
Date 7/20/22 6:01 PM
Subject **added images for Git section and cleaned up the text**
Parent 6c0605aa1ef2d9b82adee64173f16408680fbc33

r-project-mgmt-repro/r-project-mgmt-repro-part-01.Rmd

r-project-mgmt-repro/r-project-mgmt-repro-part-01.Rmd View file @ 4e4f6465

```
@@ -403,6 +403,50 @@ There are two places we can interact with Git in the RStudio interface.

1.  Menu bar

![](images/git-03.png){width="640"}

Figure 13: Menu bar options for working with GIT

2.  Git Panel

![](images/git-04.png){width="640"}
```

Figure 22: History for training manual.

The Git workflow can be summarized as:

```
Pull > add > commit > push
```

Pull: *download* the most recent version of the repository *from* GitHub *to* your local computer.

Push: *upload* the most recent version of the repository *to* GitHub *from* your local computer.

Tips for working with Git

- You should pull each time you start working on your project after a hiatus, or before each edit if you know a team member is working at the same time.
- Commit frequently, each commit should be a distinct set of edits which you can summarize in 50 characters or less. Don't add a bunch of unrelated edits to the same commit, it makes it harder to look back through your "snapshots" and find the right one if you need to.
- At the end of your work session (or more frequently if you are working at the same time as team members), "push" your commits to the remote repository - this is the only way your local changes get added to your team's remote repository.

This pull, add, commit, push routine will become second nature. Pulling at the beginning and pushing at the end of your work session becomes a sort of ritual that marks the beginning and end of your work session.

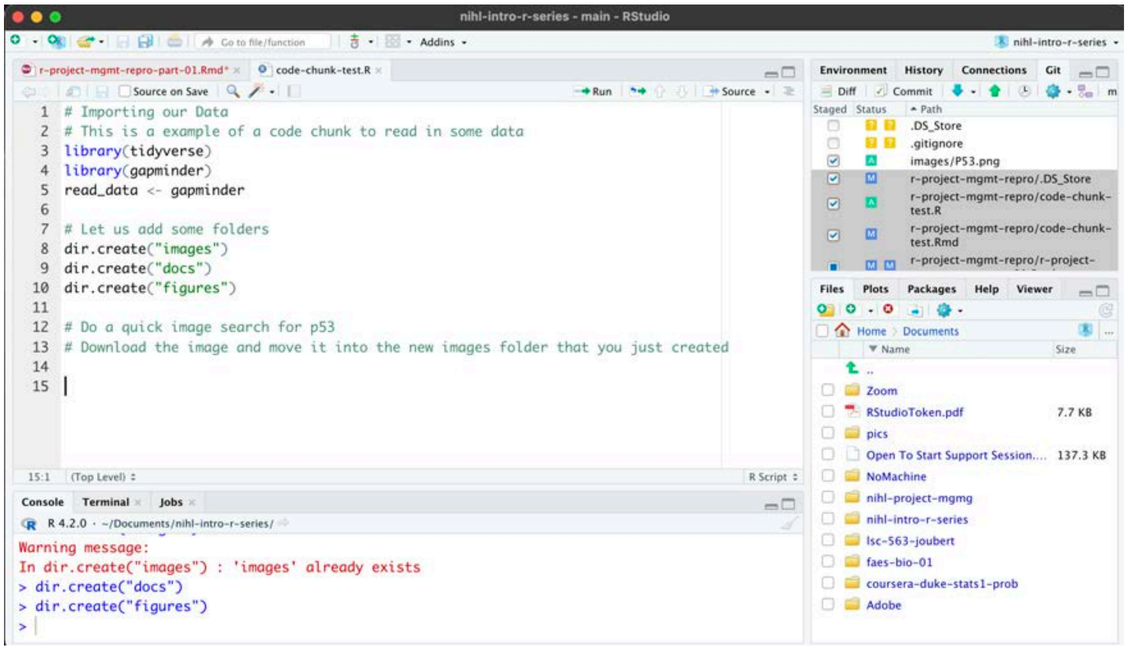
Pushing our Edits to Git

Create a new R-script named "proj-mgmt-01" Add the following lines of code

```
# Importing our Data

# This is an example of a code chunk to read in some data

library(tidyverse)

library(gapminder)

read_data <- gapminder

# Let us add some folders

dir.create("images")

dir.create("docs")

dir.create("figures")

# Do a quick image search for p53

# Download the image and move it into the new images folder that you just
created
```

We are now ready to send these changes to our remote repository

1. Select each file that has been modified from the list in the Git panel except. [Figure 23]



```
# Importing our Data
# This is a example of a code chunk to read in some data
library(tidyverse)
library(gapminder)
read_data <- gapminder

# Let us add some folders
dir.create("images")
dir.create("docs")
dir.create("figures")

# Do a quick image search for p53
# Download the image and move it into the new images folder that you just created
```

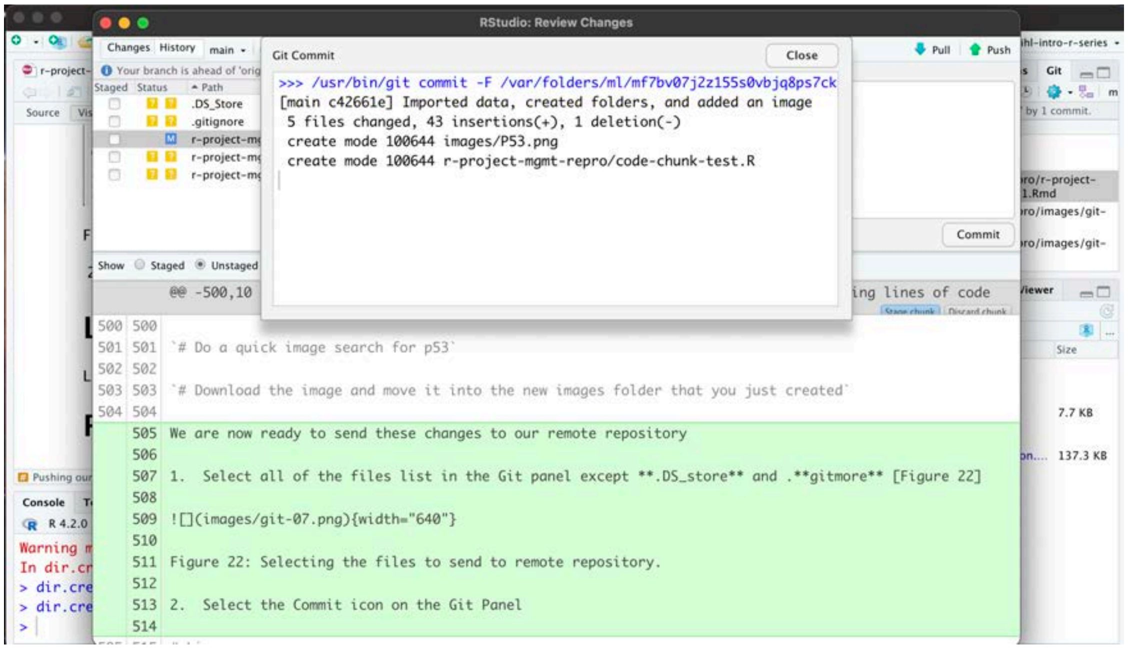
```
Warning message:
In dir.create("images") : 'images' already exists
> dir.create("docs")
> dir.create("figures")
>
```

Staged	Status	Path
☐	? ?	.DS_Store
☐	? ?	.gitignore
✓	A	images/P53.png
✓	M	r-project-mgmt-repro/.DS_Store
✓	A	r-project-mgmt-repro/code-chunk-test.R
✓	M	r-project-mgmt-repro/code-chunk-test.Rmd
■	M M	r-project-mgmt-repro/r-project-

Name	Size
..	
Zoom	
RStudioToken.pdf	7.7 KB
pics	
Open To Start Support Session....	137.3 KB
NoMachine	
nihl-project-mgmg	
nihl-intro-r-series	
lsc-563-joubert	
faes-bio-01	
coursera-duke-stats1-prob	
Adobe	

Figure 23: Selecting the files to send to remote repository.

2. Select the Commit icon on the Git Panel. Once you do this, the commit screen will refresh showing the changes [Figure 24].

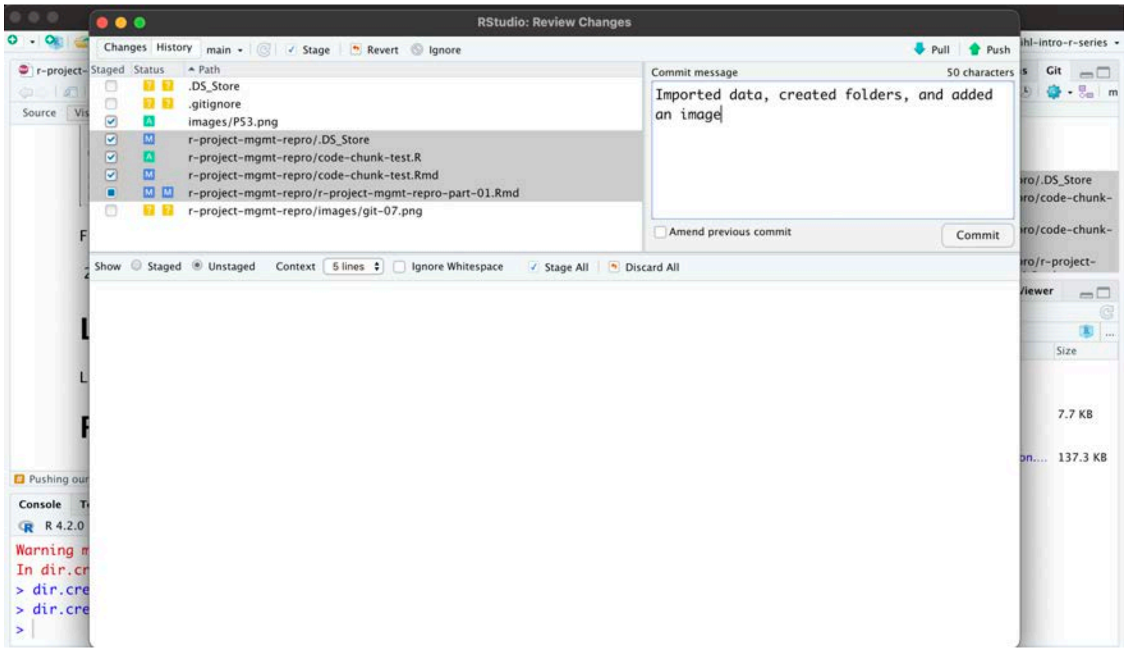


```
>>> /usr/bin/git commit -F /var/folders/ml/mf7bv07j2z155s0vbjq8ps7ck
[main c42661e] Imported data, created folders, and added an image
 5 files changed, 43 insertions(+), 1 deletion(-)
 create mode 100644 images/P53.png
 create mode 100644 r-project-mgmt-repro/code-chunk-test.R
```

```
@@ -500,10                                   ing lines of code

`# Do a quick image search for p53`

`# Download the image and move it into the new images folder that you just created`

We are now ready to send these changes to our remote repository

1.  Select all of the files list in the Git panel except **.DS_store** and .**gitmore** [Figure 22]

![](images/git-07.png){width="640"}

Figure 22: Selecting the files to send to remote repository.

2.  Select the Commit icon on the Git Panel

```

Figure 24: Changes being sent to GitHub.

3. Add a short commit message, and select the commit button [Figure 25].

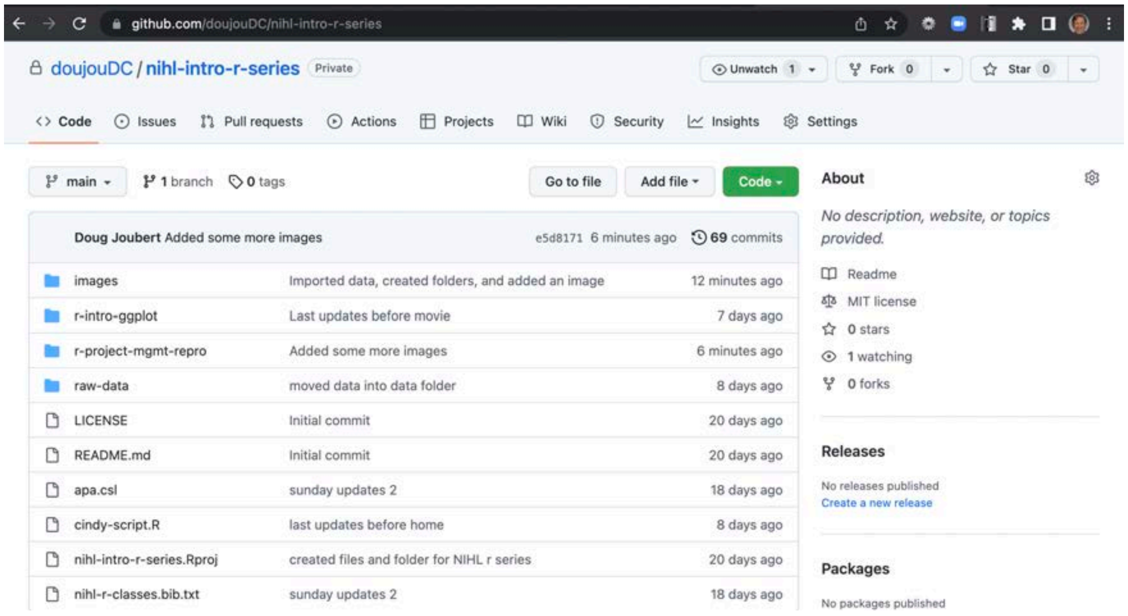


```
.DS_Store
.gitignore
images/P53.png
r-project-mgmt-repro/.DS_Store
r-project-mgmt-repro/code-chunk-test.R
r-project-mgmt-repro/code-chunk-test.Rmd
r-project-mgmt-repro/r-project-mgmt-repro-part-01.Rmd
r-project-mgmt-repro/images/git-07.png
```

Commit message: Imported data, created folders, and added an image

Figure 25: Adding a commit message and committing the changes.

4. Close the Git Commit screen
5. Select the Push button to send the changes to GitHub
6. If we check our repository on GitHub, the changes will be reflected [Figure 26]



Doug Joubert Added some more images e5d8171 6 minutes ago 69 commits

images	Imported data, created folders, and added an image	12 minutes ago
r-intro-ggplot	Last updates before movie	7 days ago
r-project-mgmt-repro	Added some more images	6 minutes ago
raw-data	moved data into data folder	8 days ago
LICENSE	Initial commit	20 days ago
README.md	Initial commit	20 days ago
apa.csl	sunday updates 2	18 days ago
cindy-script.R	last updates before home	8 days ago
nihl-intro-r-series.Rproj	created files and folder for NIHL r series	20 days ago
nihl-r-classes.bib.txt	sunday updates 2	18 days ago

No description, website, or topics provided.

No releases published

No packages published

Licenses

References

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility [Journal Article]. *Nature*, *533*(7604), 452–454. https://doi.org/10.1038/533452a

Bollen, K., Cacioppo, J. T., Kaplan, R. M., Krosnick, J. A., & Olds, J. L. (2015). *Social, behavioral, and economic sciences perspectives on robust and reliable science* [Report]. Subcommittee on Replicability in Science.

Heiss, A. (2022). *Amounts and proportions* (Web Page October 3; Vol. 2022). https://datavizs22.classes.andrewheiss.com/content/04-content/

Markowetz, F. (2015). Five selfish reasons to work reproducibly [Journal Article]. *Genome Biology*, *16*(1), 274. https://doi.org/10.1186/s13059-015-0850-7

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing [Journal Article]. *PLOS Computational Biology*, *13*(6), e1005510–e1005510. https://doi.org/10.1371/journal.pcbi.1005510

Ziemann, M., Eren, Y., & El-Osta, A. (2016). Gene name errors are widespread in the scientific literature [Journal Article]. *Genome Biology*, *17*(1), 177. https://doi.org/10.1186/s13059-016-1044-7