Project Primer

Dan Spalla

2020-06-30

Contents

Preface

This documentation is a primer for the COVID-19 DCTT BCS-2032203 project. It covers an introduction to Git, using Git in RStudio, an introduction to RMarkdown, the project repository, and project helper scripts.

This documentation assumes an introductory understanding of the R programming language and RStudio. It also assumes zero knowledge of Git, using Git in RStudio, and RMarkdown; consequently, some explanations may be structured or phrased in a way that are conducive to understanding but not perfectly accurate.

Chapter 1

Git Introduction

1.1 What is Git?

Git is a distributed version control system that was developed by Linus Torvalds for working on the Linux kernel. Despite its original use case, it is used for many software projects. It has become a popular choice for version control systems and becomes a powerful tool once the basics are learned.

Users traditionally interface with Git through a Unix shell, or commandline environment, but there are several graphical or GUI options for interfacing with the tool. More information about this can be found in the Appendix section of this documentation.

Git, and version control systems more generally, enable users to store different versions of files and keep track of changes between those versions. Git helps users document a history of changes, including mistakes, to their projects and enables them to switch to any past version. The result is essentially a list of different versions of a project that represent its state during a given time.

1.2 What is a Repository?

A Git repository is a directory, or folder, where version tracking and history has been initialized. One can check if version tracking is initialized in a directory by looking for a hidden ".git" subdirectory (you may have to enable a, "show hidden files" feature in your file manager if you wish to do this). This hidden directory stores metadata about a project, metadata of the files in that project, and the detailed history of that project. Note that the actual project files reside in the directory that contains the hidden ".git" directory, not the ".git" directory itself.

A repository is an entity that is local in nature, meaning it resides on your machine and you treat it the same way you would any other file. As a result, any changes you make are yours and you must deliberately choose to share those changes if you want others to see them.

1.3 What Is A Commit?

A commit is a change to a repository and you do this when you implement a change to your project. This change can be as simple as the addition of a new file to your project or as complex as a series of changes across multiple files. A user "commits" to a repository when they want to store a new version of a project. When this is done, they now have a distinct version of the project that they can go back to and view the changes from one version to another.

Generally, commits should contain a change that is singular in purpose. If two separate parts of a project were to be changed, they should be "committed" to the repository separately. A commit represents a point in time, a specific version of your project, and if two changes representing separate parts of a project were to end up in the same commit, then it would make the history of your project confusing or not clear. It is desirable to be able to understand the evolution of a project as changes are made and this becomes more difficult if changes are not distinct.

A commit will have a message associated with it. This message should describe why changes were made and what purpose these changes fulfill. Each commit will have a log of what aspects of your project have changed and a commit message should also briefly describe what changes were made if they are not self-explanatory.

For the sake of argument, let's say you change a readme file to include a link to a funding page for your project. The following is an example of a poor commit message.

> Changed the readme.

A better message would be:

> Added a link to the funding page.

An even better message would be:

> Added a link to the funding page in the reference section of the readme. Those who want to fund the project no longer have to email asking how to support it.

1.4 Understanding Distributed

As was stated previously, Git is a distributed version control system which means that every repository, including ones on your local machine, do not require some centralized service to function.

When you share changes with others, you are simply merging histories or syncing changes rather than performing some sort of special transaction with a centralized service. It is important to have a conceptual understanding of this because this is how you use Git when working on a project. You make and commit changes locally, then you can share those changes with the rest of the world. You cannot share changes with the rest of the world if you have not committed those changes locally.

1.5 Push

The act of "push", or "pushing", is when changes are uploaded to remote locations. Once a change is made and committed to your local repository, you can then send a copy of your changes to a remote location (push). This remote location is simply housing another copy of that repository. Anyone with the proper permissions can push to a remote repository so that someone else can download (pull) their changes.

1.6 Pull

The act of "pull", or "pulling", is when changes are downloaded from remote locations. If someone pushes a change into a remote repository, you can download (pull) that change and sync it with your local repository. Anyone can pull changes from a remote repository and sync it with their local version. This is also how repositories are initially downloaded if they are not present on your machine.

1.7 The Log

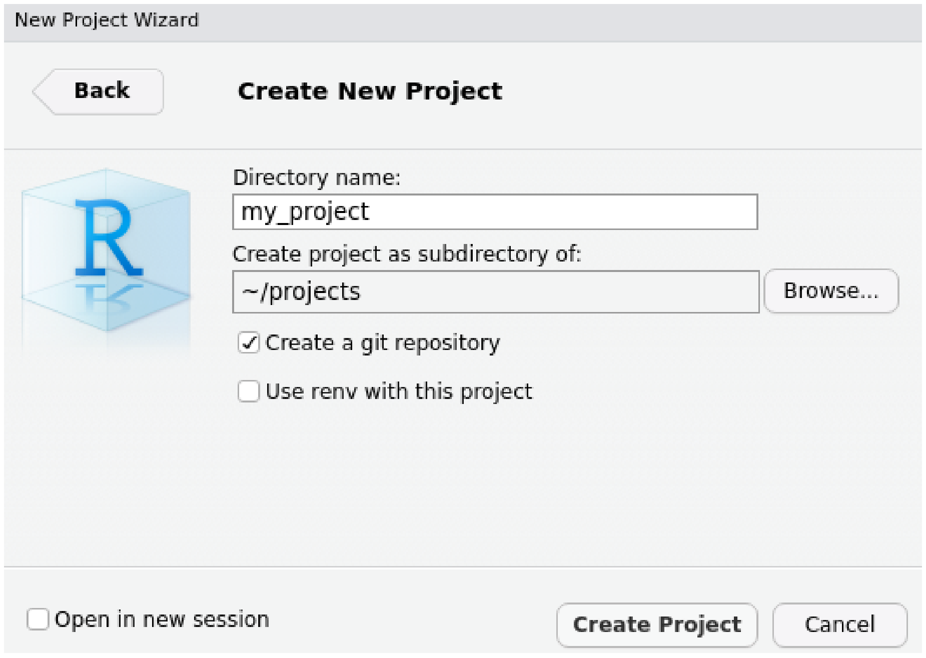
Each Git repository has its own history that details which files have changed, who changed them, and their stated reasons for changing them. This can be viewed by using the log functionality of Git. The log will give you an overview of each project change by showing all the aforementioned information as well as detailing the exact changes. This can be used to track project changes and understand its history.

Chapter 2

Using Git in Rstudio

2.1 Initializing a Repository For a New Project

If you haven't yet created a new project in RStudio, you can check the "Create a git repository" box in the new project wizard to automatically create one for you.



Directory name: my_project

Create project as subdirectory of: ~/projects

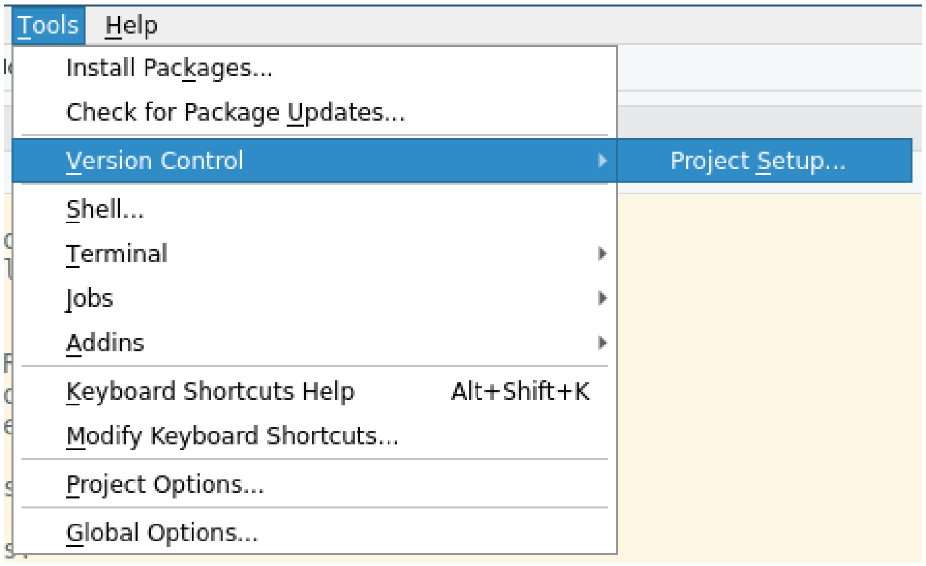
[✓] Create a git repository

[ ] Use renv with this project

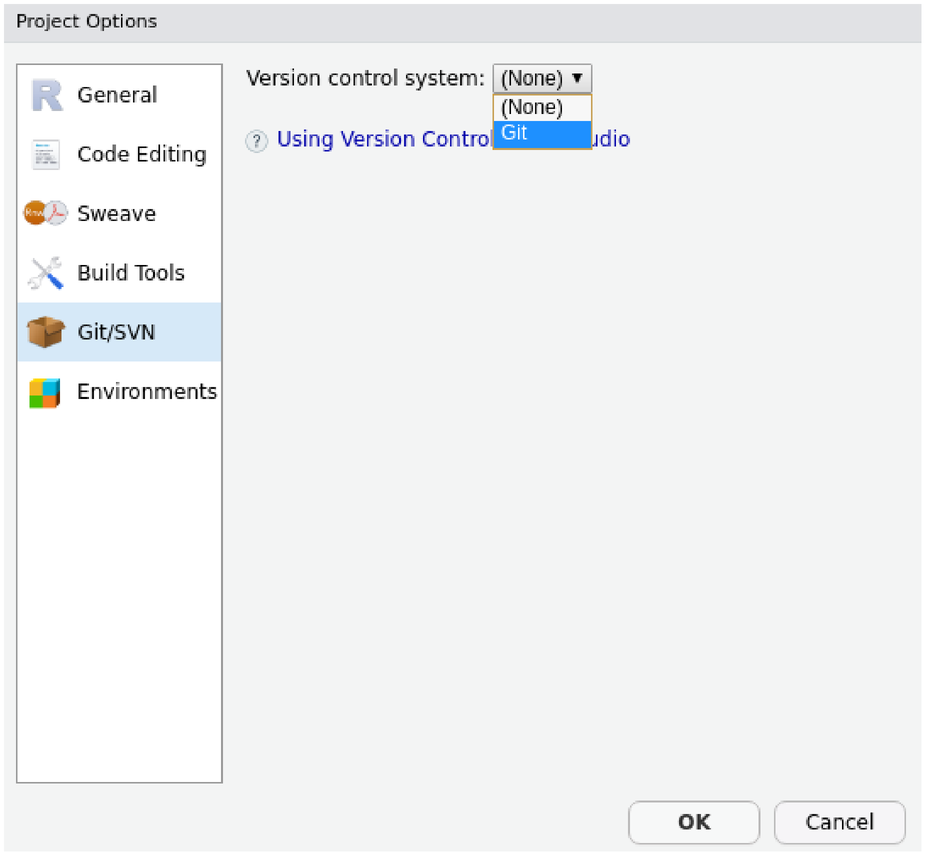
[ ] Open in new session

If you already have an existing project but wish to initialize a Git repository for it, its quite simple.

1. Select “Tools” from the toolbar, “Version Control”, and click on “Project Setup…”

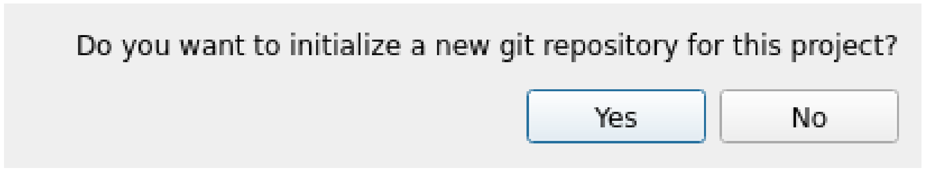


2. Select “Git” from the dropdown box

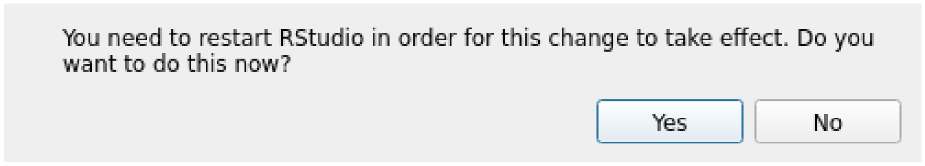

Project Options

Version control system: (None)

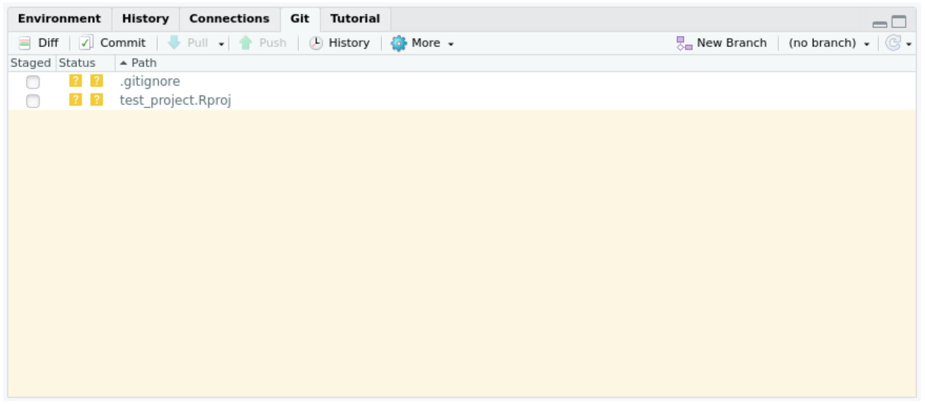
3. Confirm that you want to create a new Git repository


Do you want to initialize a new git repository for this project?

4. It may ask you to restart RStudio; restart RStudio


You need to restart RStudio in order for this change to take effect. Do you want to do this now?

5. The Git tab will appear on the top-right window and show the status of your new repository

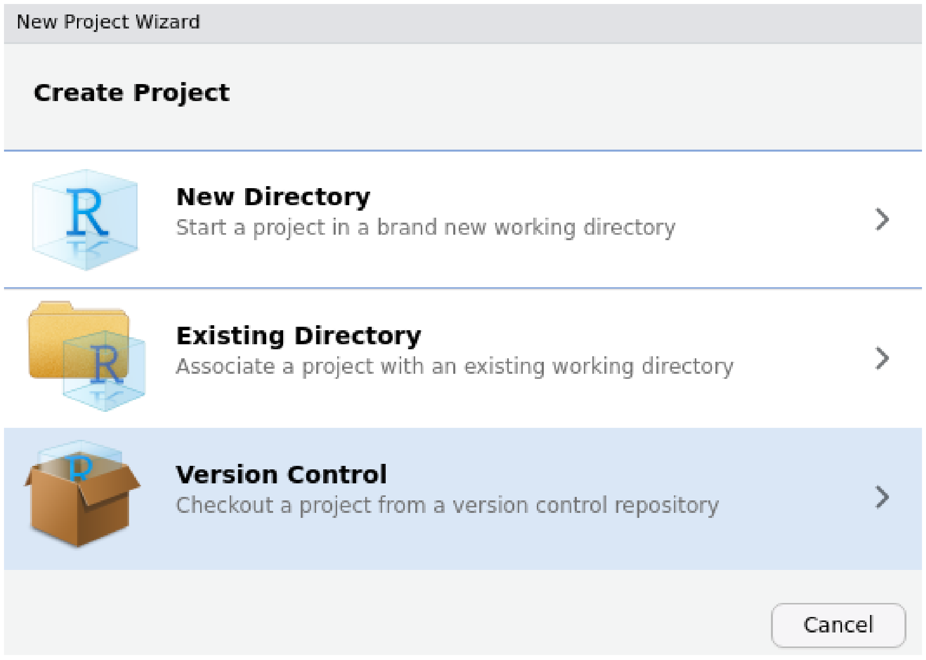


.gitignore
test_project.Rproj

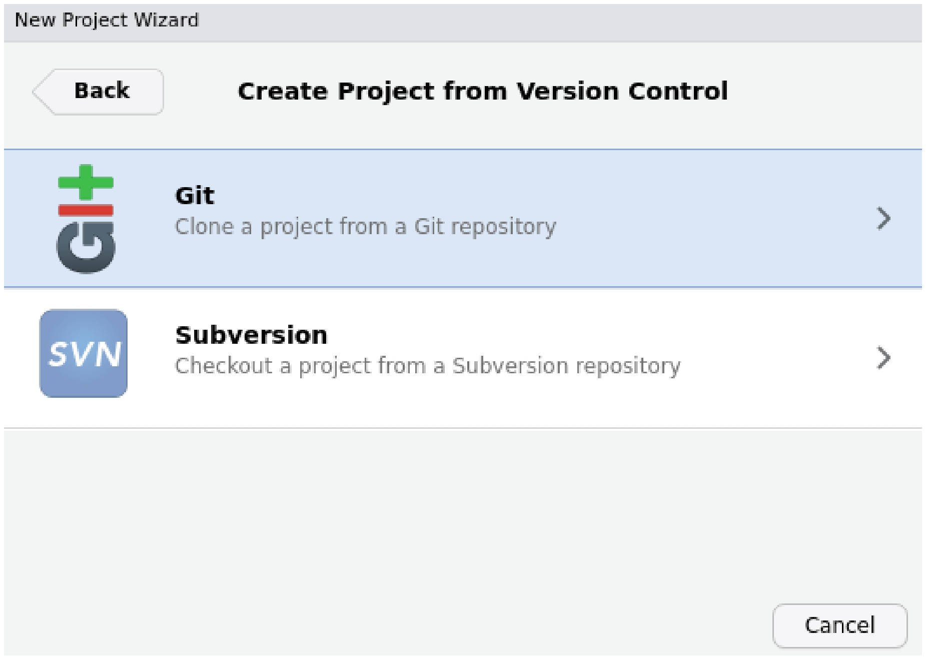
2.2 Creating a Project From a Git Repository

Creating a new RStudio project from a Git repository, such as one hosted on GitHub, is quite simple.

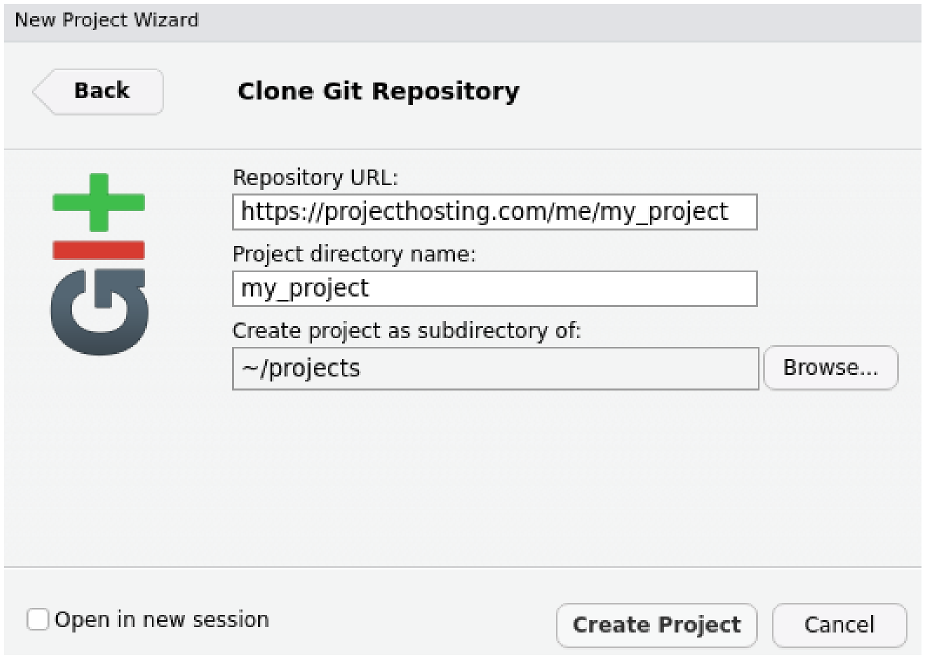
1. Create a new project
2. Select "Version Control"


New Project Wizard
Create Project
New Directory
Start a project in a brand new working directory
Existing Directory
Associate a project with an existing working directory
Version Control
Checkout a project from a version control repository
Cancel

3. Select "Git"


New Project Wizard
Back
Create Project from Version Control
Git
Clone a project from a Git repository
SVN
Subversion
Checkout a project from a Subversion repository
Cancel

4. Enter the repository URL, give the project a name, and select the directory where project files shall reside



Repository URL: https://projecthosting.com/me/my_project

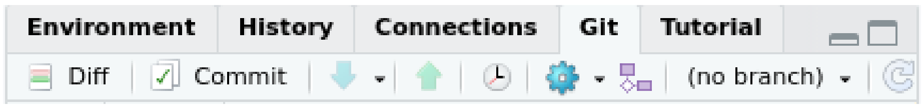
Project directory name: my_project

Create project as subdirectory of: ~/projects

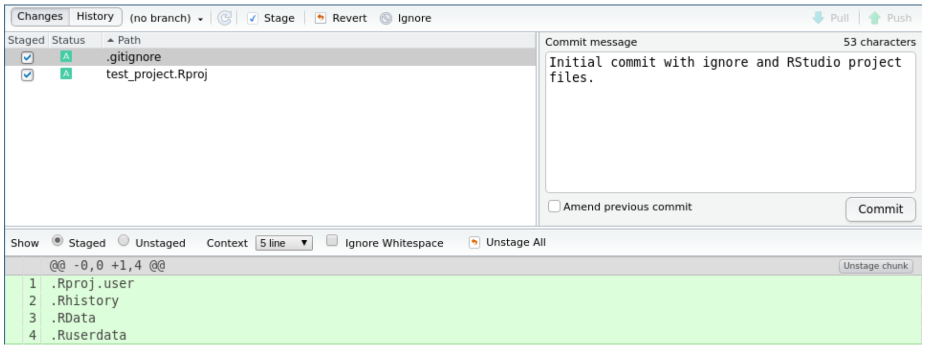
2.3 Making Changes and Committing

To make changes to files in a git repository, modify them you would any other file. When you are ready to commit your changes, do the following.

1. Hit the commit button under the Git tab and a new window will appear



2. Select, stage, the changed files that you wish to add and commit

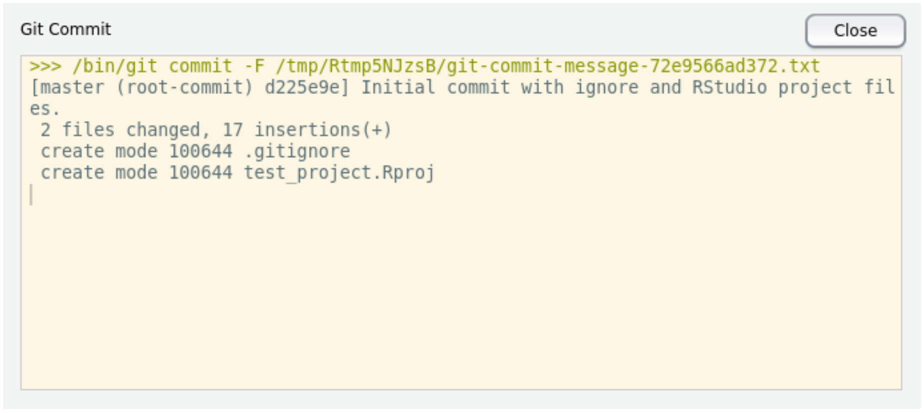


.gitignore
test_project.Rproj

```
Initial commit with ignore and RStudio project
files.
```

```
@@ -0,0 +1,4 @@
.Rproj.user
.Rhistory
.RData
.Ruserdata
```

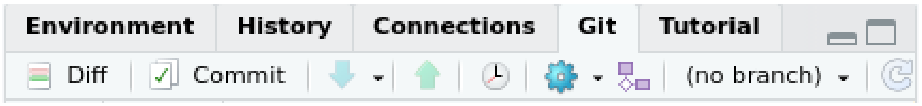
3. Write a commit message that succinctly describes the "what" and "why" of your changes and hit the commit button. A status window will show whether or not it was successful



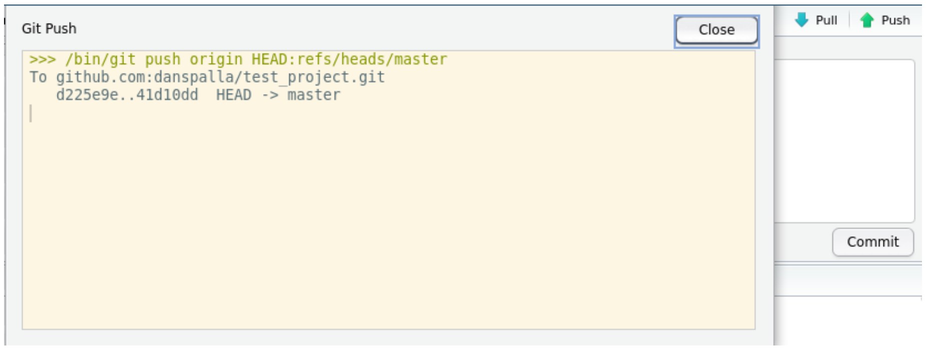
```
>>> /bin/git commit -F /tmp/Rtmp5NJzsB/git-commit-message-72e9566ad372.txt
[master (root-commit) d225e9e] Initial commit with ignore and RStudio project fil
es.
 2 files changed, 17 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 test_project.Rproj
```

2.4 Sharing Those Changes, Push

When you are ready to share your changes with the rest of the world, press the push button under the Git tab or on the commit window.



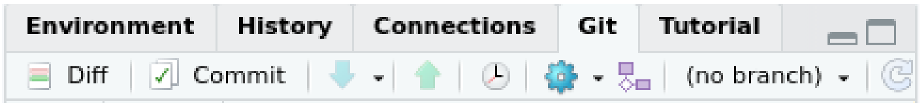
A status window will appear and show its success.



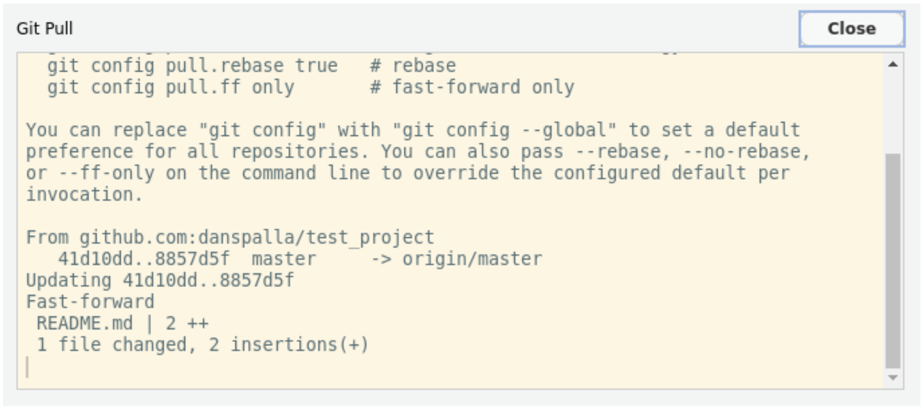
```
>>> /bin/git push origin HEAD:refs/heads/master
To github.com:danspalla/test_project.git
   d225e9e..41d10dd  HEAD -> master
```

2.5 Getting Changes From Others, Pull

To receive updates that others have pushed, hit the pull button under the Git tab.



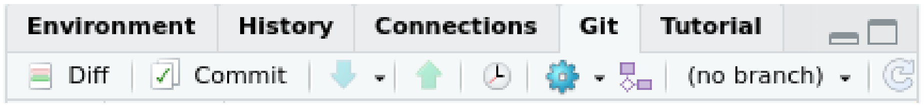
A status window will appear and show its success.



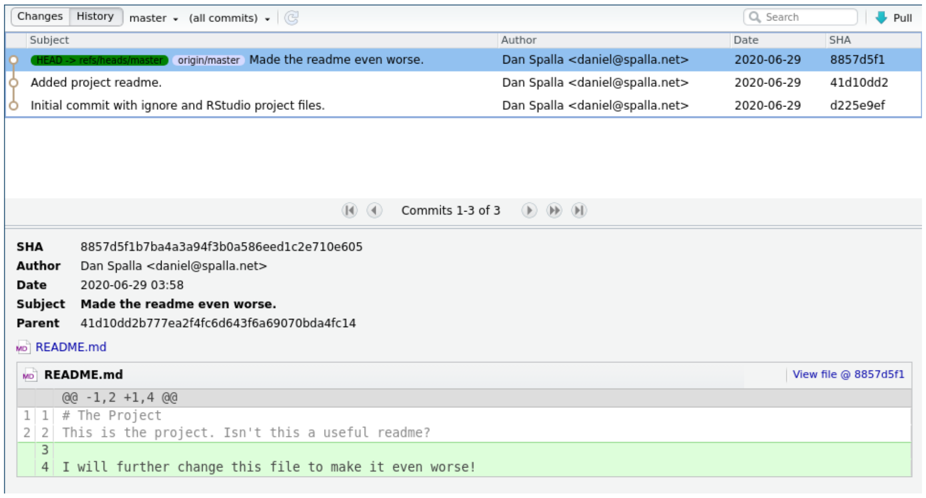
```
  git config pull.rebase true   # rebase
  git config pull.ff only       # fast-forward only

You can replace "git config" with "git config --global" to set a default
preference for all repositories. You can also pass --rebase, --no-rebase,
or --ff-only on the command line to override the configured default per
invocation.

From github.com:danspalla/test_project
   41d10dd..8857d5f  master     -> origin/master
Updating 41d10dd..8857d5f
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
```

2.6 Viewing the Log

You can view the git log for a repository by clicking the history button.



You can click on each commit, read each commit changes, and see what files in the project were changed for that commit.



Subject	Author	Date	SHA
HEAD -> refs/heads/master origin/master Made the readme even worse.	Dan Spalla <daniel@spalla.net>	2020-06-29	8857d5f1
Added project readme.	Dan Spalla <daniel@spalla.net>	2020-06-29	41d10dd2
Initial commit with ignore and RStudio project files.	Dan Spalla <daniel@spalla.net>	2020-06-29	d225e9ef

Commits 1-3 of 3

SHA	8857d5f1b7ba4a3a94f3b0a586eed1c2e710e605
Author	Dan Spalla <daniel@spalla.net>
Date	2020-06-29 03:58
Subject	**Made the readme even worse.**
Parent	41d10dd2b777ea2f4fc6d643f6a69070bda4fc14

README.md

README.md View file @ 8857d5f1

```
    @@ -1,2 +1,4 @@
1 1 # The Project
2 2 This is the project. Isn't this a useful readme?
  3
  4 I will further change this file to make it even worse!
```

Chapter 3

RMarkdown Introduction

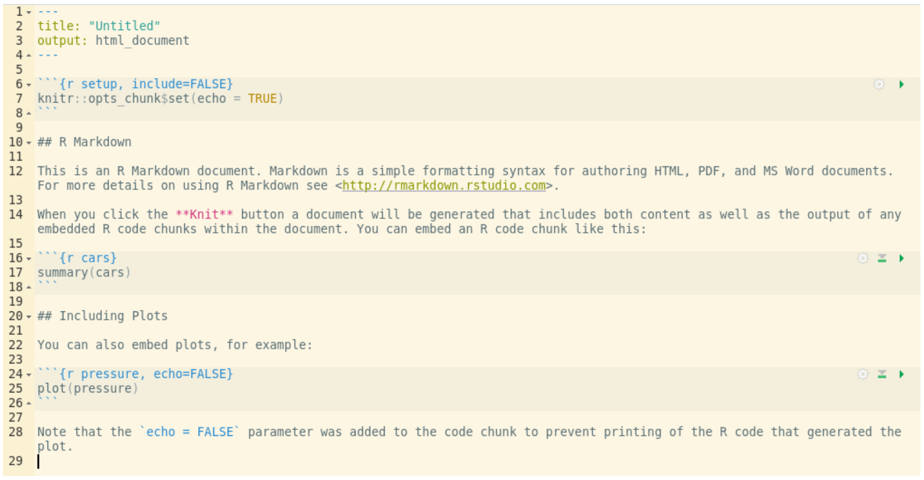
3.1 What is Markdown?

Markdown is a markup language that enables a user to type a certain syntax and always generate the same output documents. There are no hidden characters or special formatting, that you would inevitably encounter using a word processor, which could cause unwanted changes to appearance of a document. Please see the, "Markdown Introductory Tutorial" in the appendix for a basic interactive tutorial.

3.2 What is RMarkdown?

RMarkdown is an extension of the Markdown markup language that allows the user to embed R code in Markdown documents. A markup language is a simple set of syntax that when written will always produce the same output document (as opposed to WYSIWYG editors). RMarkdown extends this by embedding R code into the documents, showcasing the source code, a R script's output, and any figures generated by the R script.

Example of source code and output below.



~~~
---
title: "Untitled"
output: html_document
---

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

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents.
For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any
embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

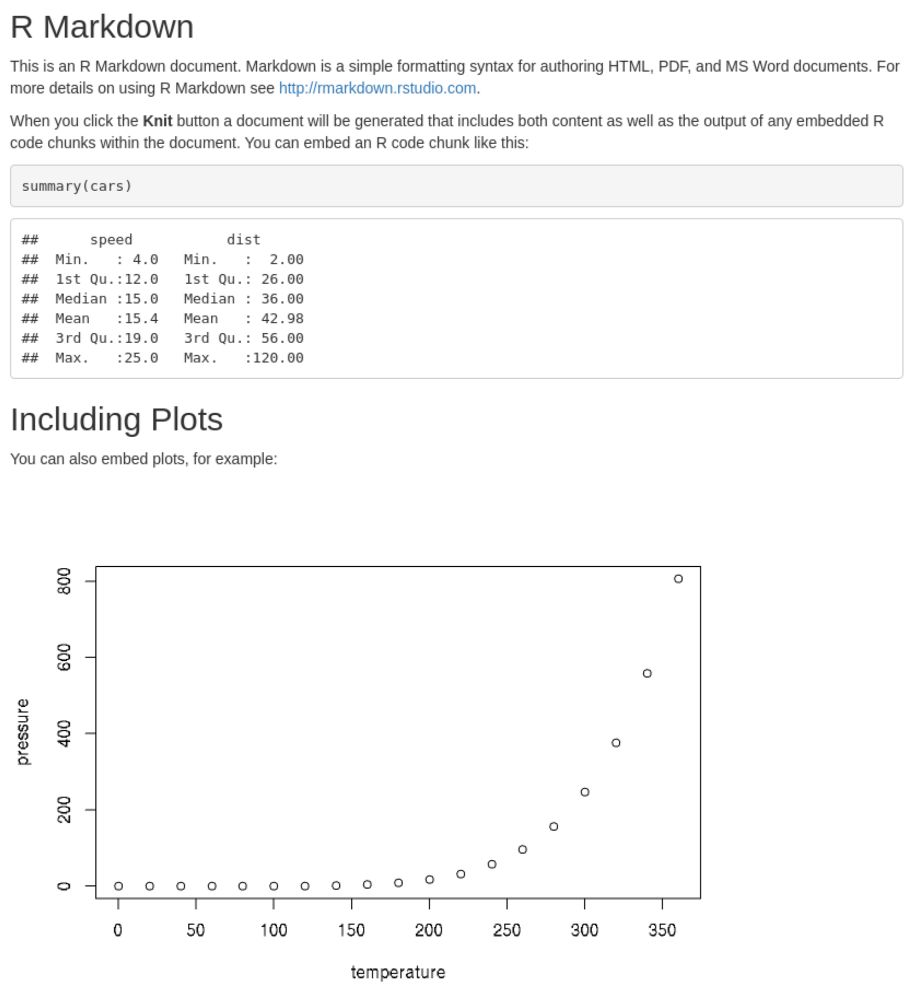
Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the
plot.

```
```



R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

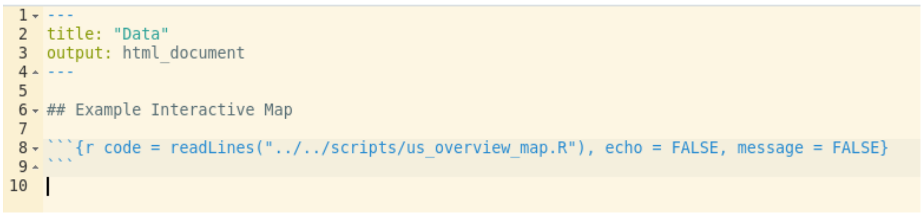
Including Plots

You can also embed plots, for example:

3.3 R Code in RMarkdown

There are two ways to include R code in an RMarkdown document. The first is to simply inline the code in the document itself and compile the document. The second way is to load the code from a file instead of inlining in the document. Loading the code from a file provides a nice separation between the data processing and data presentation of a R script. The second method should be used if the R code is lengthy, complicated, or needs to be worked on independent of successful RMarkdown document compilation.

To load a R script from a file, simply add "code = readlines("location_to_file")" and specify the path to the file.



~~~
---
title: "Data"
output: html_document
---

## Example Interactive Map

```{r code = readLines("../../scripts/us_overview_map.R"), echo = FALSE, message = FALSE}
```

~~~

Chapter 4

Contribution to Project

4.1 Project Repository

The project repository can be found at GitHub. The repository provides the website, website source code, geodatabase, and helper scripts.

4.2 Overview of Project Technologies

The project makes use of a number of technologies.

- R: used for scripts that help manage website content updates, interactive map, and analysis.
- RMarkdown: used to create website pages and embed R code
- RStudio: (optional), used to facilitate easy project contribution and management

4.3 Repository Directory Hierarchy

The repository uses the following directory hierarchy:

- /website: Folder where the website source code is stored; new content for the website is created and added here.
- /docs: "Mount point" for GitHub pages; the project can be configured to serve the website from this folder (GitHub provides no flexibility on this), requiring no further intervention from team members.
- /data: Folder where any data files are stored for use in R scripts or RMarkdown files.
- /scripts: Folder where any R scripts, or other software scripts/tools, are stored. Ex: Helper scripts for building and local previewing of the website are stored here.

4.4 How To Use The Helper Scripts

The following helper scripts are currently available for use in the "/scripts" directory.

- "project_install_dependencies.R": This script installs all the dependencies needed on the project. This should be used when the project has been pulled for the first time.
- "website_preview.R": This script will automatically build the website locally and launch a local web server that can be used to preview the site. This mode will continuously check website files for update and automatically rebuild the site so that changes can be easily previewed.
- "website_build.R": This script builds the website without providing a preview to the user.

4.5 Getting Started With Contribution

To get started contributing to the project, use the following steps.

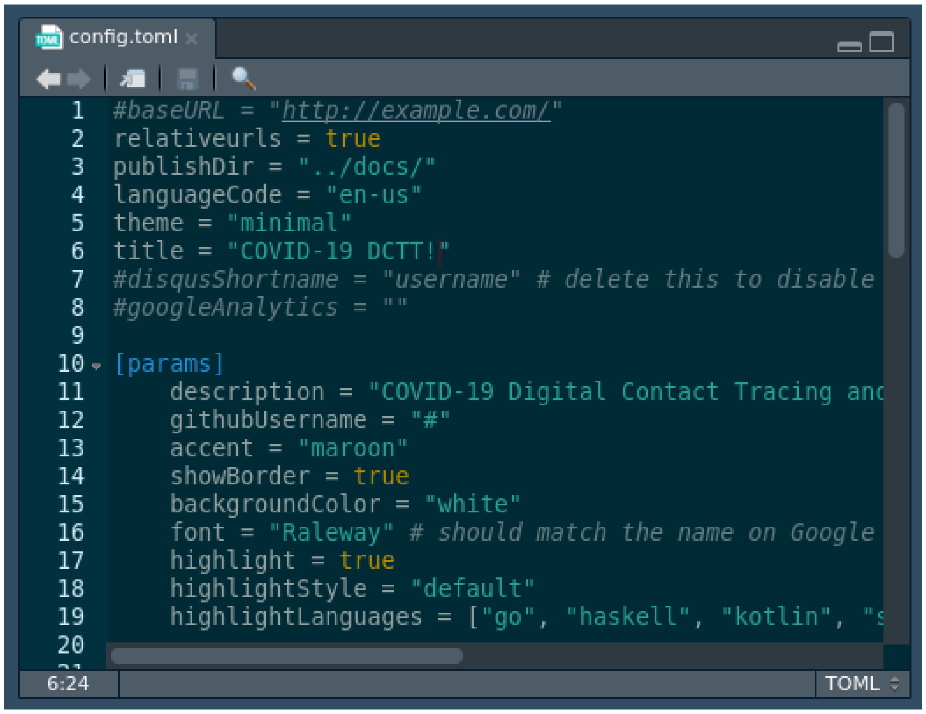
Using RStudio:

1. Open RStudio
2. Create a new project, select the version control option, select Git as the version control system, and enter the GitHub repository URL
3. Wait for RStudio to automatically pull the latest project files from GitHub
4. Navigate to the scripts folder and run the, "project_install_dependencies.R" script; this will install all dependencies and may take some time.

4.6 Contribution Example 1

Scenario: Update the website title to better reflect the project.

1. Navigate to the "/website" directory
2. Open the, "config.toml" file
3. Change the line that says, 'title = "COVID-19 DCTT" ' and change it to, 'title = "COVID-19 DCTT!" '; save the file

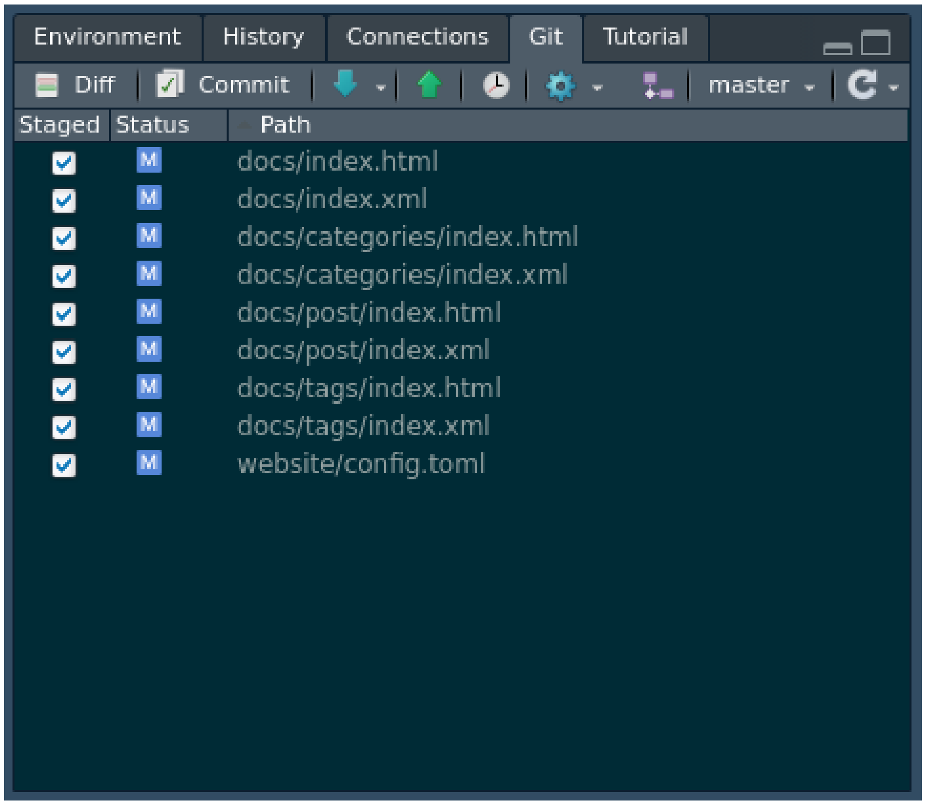


config.toml

```
#baseURL = "http://example.com/"
relativeurls = true
publishDir = "../docs/"
languageCode = "en-us"
theme = "minimal"
title = "COVID-19 DCTT!"
#disqusShortname = "username" # delete this to disable
#googleAnalytics = ""

[params]
    description = "COVID-19 Digital Contact Tracing and
    githubUsername = "#"
    accent = "maroon"
    showBorder = true
    backgroundColor = "white"
    font = "Raleway" # should match the name on Google
    highlight = true
    highlightStyle = "default"
    highlightLanguages = ["go", "haskell", "kotlin", "s
```

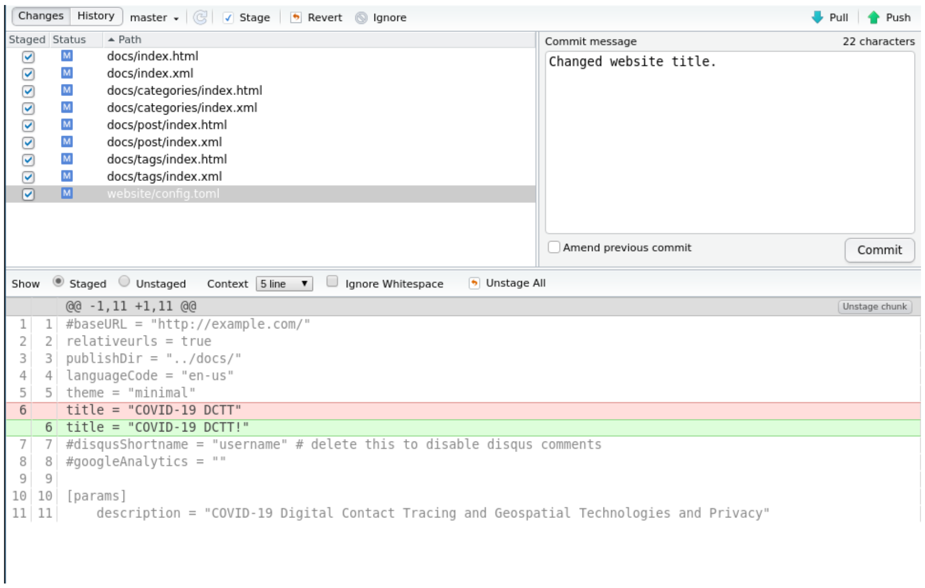
6:24 TOML

4. Navigate to the "/scripts" directory and run the, "website_preview.R" script to preview the website
5. Notice that there are changes in the "/website" directory and the "/docs" directory in the Git tab



Staged	Status	Path
✓	M	docs/index.html
✓	M	docs/index.xml
✓	M	docs/categories/index.html
✓	M	docs/categories/index.xml
✓	M	docs/post/index.html
✓	M	docs/post/index.xml
✓	M	docs/tags/index.html
✓	M	docs/tags/index.xml
✓	M	website/config.toml

6. Commit changes to your local repository stating the change



Staged	Status	Path
✓	M	docs/index.html
✓	M	docs/index.xml
✓	M	docs/categories/index.html
✓	M	docs/categories/index.xml
✓	M	docs/post/index.html
✓	M	docs/post/index.xml
✓	M	docs/tags/index.html
✓	M	docs/tags/index.xml
✓	M	website/config.toml

```
Changed website title.
```

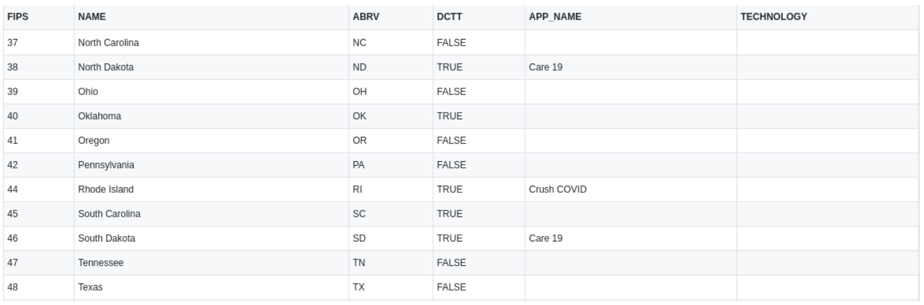
```
@@ -1,11 +1,11 @@
#baseURL = "http://example.com/"
relativeurls = true
publishDir = "../docs/"
languageCode = "en-us"
theme = "minimal"
title = "COVID-19 DCTT"
title = "COVID-19 DCTT!"
#disqusShortname = "username" # delete this to disable disqus comments
#googleAnalytics = ""

[params]
    description = "COVID-19 Digital Contact Tracing and Geospatial Technologies and Privacy"
```

7. Push your changes to the GitHub repository to share the changes with the team.

4.7 Contribution Example 2

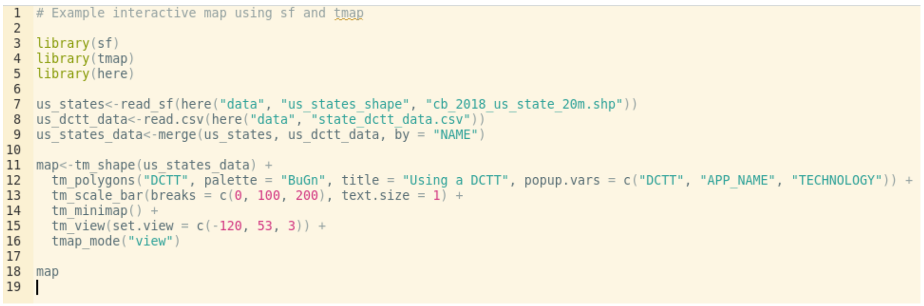
Scenario: Create an interactive overview map for the US to show which states are using digital contact tracing and which states are not; include the technology used and application name if available.

1. Obtain the state data on DCTT, put the file in the "/data" directory



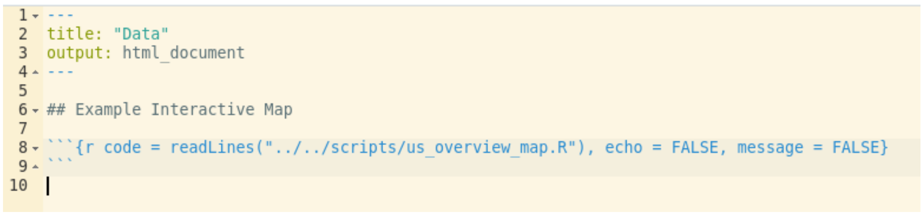
FIPS	NAME	ABRV	DCTT	APP_NAME	TECHNOLOGY
37	North Carolina	NC	FALSE		
38	North Dakota	ND	TRUE	Care 19	
39	Ohio	OH	FALSE		
40	Oklahoma	OK	TRUE		
41	Oregon	OR	FALSE		
42	Pennsylvania	PA	FALSE		
44	Rhode Island	RI	TRUE	Crush COVID	
45	South Carolina	SC	TRUE		
46	South Dakota	SD	TRUE	Care 19	
47	Tennessee	TN	FALSE		
48	Texas	TX	FALSE		

2. Place the shapefile that will be used in the "/data" folder as well 3. Write an R script to generate the interactive map and place it in the "/scripts" directory



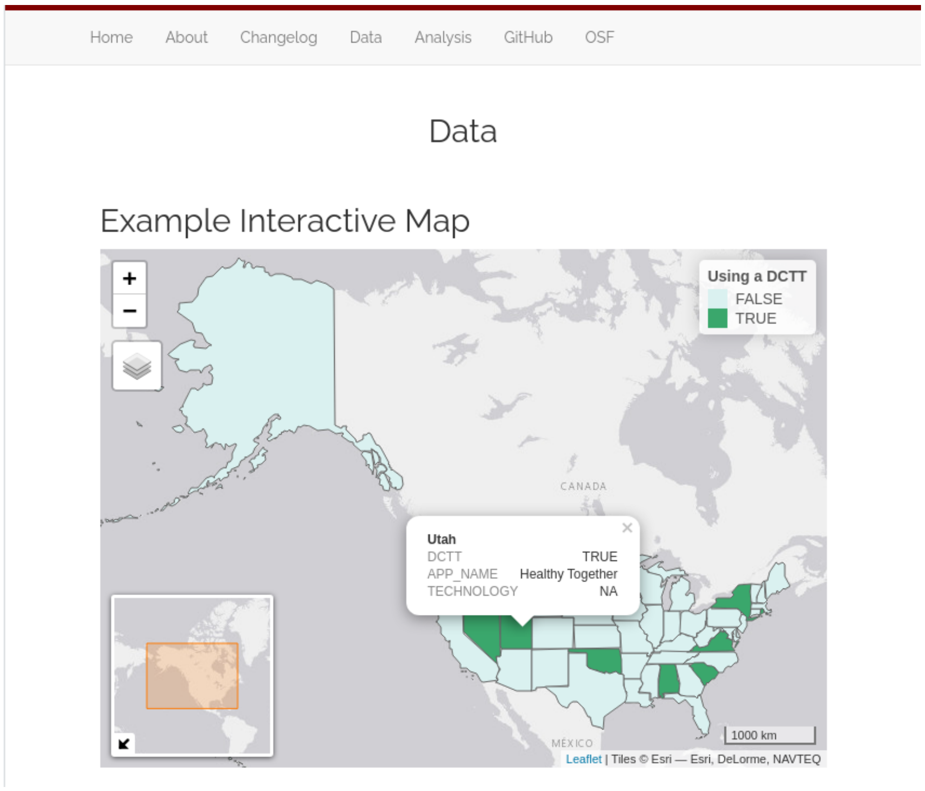
```r
# Example interactive map using sf and tmap

library(sf)
library(tmap)
library(here)

us_states<-read_sf(here("data", "us_states_shape", "cb_2018_us_state_20m.shp"))
us_dctt_data<-read.csv(here("data", "state_dctt_data.csv"))
us_states_data<-merge(us_states, us_dctt_data, by = "NAME")

map<-tm_shape(us_states_data) +
  tm_polygons("DCTT", palette = "BuGn", title = "Using a DCTT", popup.vars = c("DCTT", "APP_NAME", "TECHNOLOGY")) +
  tm_scale_bar(breaks = c(0, 100, 200), text.size = 1) +
  tm_minimap() +
  tm_view(set.view = c(-120, 53, 3)) +
  tmap_mode("view")

map
```

4. Modify the RMarkdown document on the page we want to display the map, and add the location of the script



~~~
---
title: "Data"
output: html_document
---

## Example Interactive Map

```{r code = readLines("../../scripts/us_overview_map.R"), echo = FALSE, message = FALSE}
```
~~~

5. Run the "website_preview.R" from the "/scripts" directory to preview the map locally

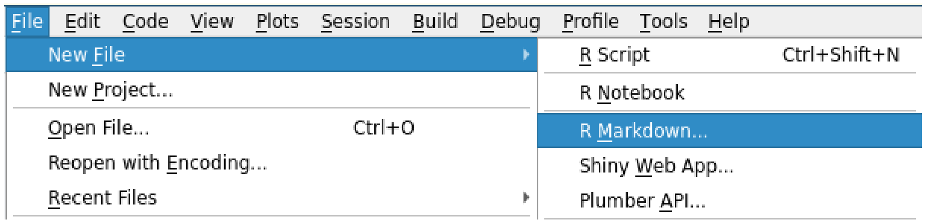


6. Commit the changes to the "/data", "/scripts", "website", and "/docs" directories to your local repository 7. Push the changes to the GitHub repository to update the website and share the interactive map with the team

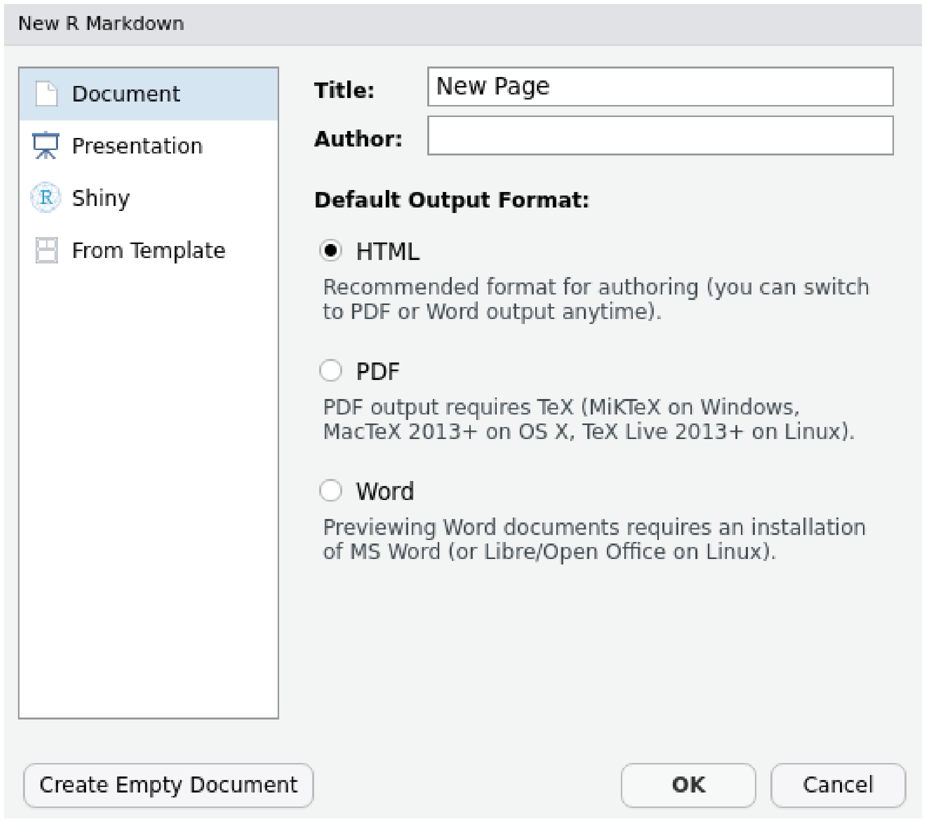
4.8 Creating a New Page

Creating a new page for the website requires a number of changes across the website source files.

1. Create a new RMarkdown file from the "File -> New File" menu.



2. Give the file an appropriate title and make sure HTML is the output format


New R Markdown

Document
Presentation
Shiny
From Template

Title: New Page
Author:

Default Output Format:

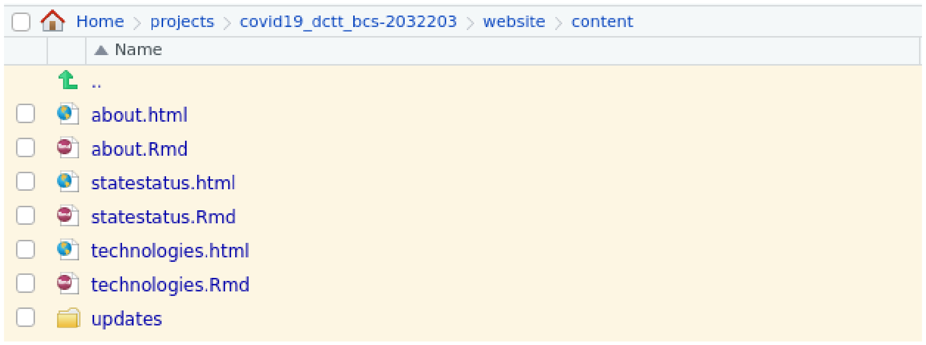
● HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

○ PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

○ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

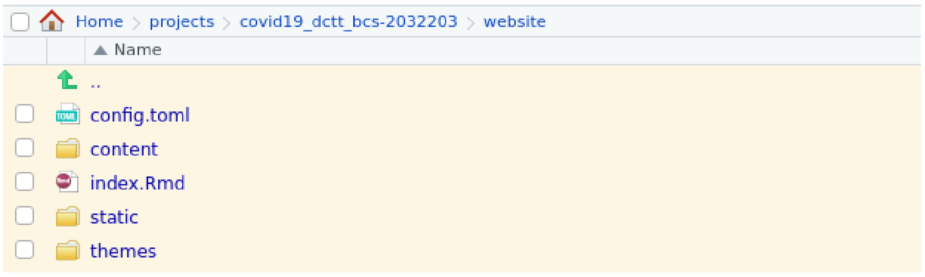
Create Empty Document | OK | Cancel

3. Save the file in the content directory: "/website/content/"


Home > projects > covid19_dctt_bcs-2032203 > website > content

▲ Name
..
about.html
about.Rmd
statestatus.html
statestatus.Rmd
technologies.html
technologies.Rmd
updates

4. Open the "config.toml" configuration file located in "/website/"


Home > projects > covid19_dctt_bcs-2032203 > website

```
Name
..
config.toml
content
index.Rmd
static
themes
```

5. Add the following entry to the configuration file to add the page to the navigation bar

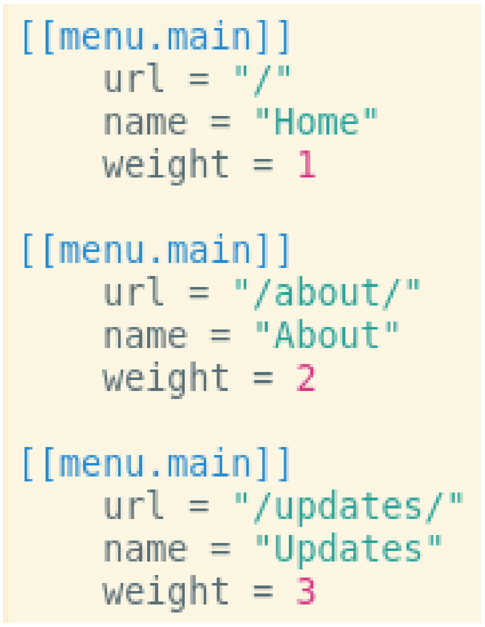
   [[menu.main]]

   url = "/page_file_name/"

   name = "Pretty Page Name"

   weight = 5

- Note: The "url" should match the RMarkdown file name exactly, the "name" will be the name that appears on the website, and the weight (relative to other pages) will specify the order that the page will appear on the navigation bar



```
[[menu.main]]
    url = "/"
    name = "Home"
    weight = 1

[[menu.main]]
    url = "/about/"
    name = "About"
    weight = 2

[[menu.main]]
    url = "/updates/"
    name = "Updates"
    weight = 3
```

6. Run the "website_preview.R" script to build and preview the changes

4.9 Creating a New Update Post

Creating a new update post for the website consists of a few steps.

1. Change your working directory to the "/website/" directory if you haven't already

- Note: Using the R package 'here' makes this even easier

  setwd(here("website"))

  or

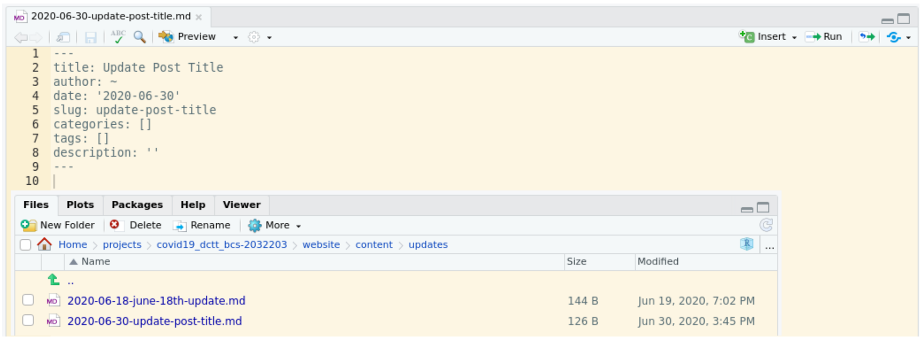
  setwd("./website") # assumes you are in project root

2. Use the blogdown function "new_post" to generate the new post

   blogdown::new_post("Update Post Title", subdir = "updates")

- Note: Blogdown is a layer on-top of a static site generator Hugo which uses the "posts" directory for all new posts. You have to specify 'subdir = "updates"' to get the new posts to appear in the "updates" directory; otherwise you will have to move your Markdown post from the "posts" directory to the "updates" directory.

2. A new Markdown file will open with your specified title and a template for the post


2020-06-30-update-post-title.md
Preview
Insert
Run
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home › projects › covid19_dctt_bcs-2032203 › website › content › updates

```
---
title: Update Post Title
author: ~
date: '2020-06-30'
slug: update-post-title
categories: []
tags: []
description: ''
---
```

```
Name                              Size    Modified
..
2020-06-18-june-18th-update.md    144 B   Jun 19, 2020, 7:02 PM
2020-06-30-update-post-title.md   126 B   Jun 30, 2020, 3:45 PM
```

3. Modify the post file to your liking and run the "website_preview.R" script to preview the changes

Appendix & Resources

Git

Download Git.

Additional Git Documentation.

Markdown

Markdown Syntax Cheat sheet

Markdown Syntax

Markdown Introductory Tutorial

R

Download R.

RMarkdown

RMarkdown Guide.

RStudio

Download RStudio.

Project Source Code

GitHub.