Session 3: Coding Workflows

GIT'ER DONE & ALL THE THINGS THAT ARE FIT TO GIT

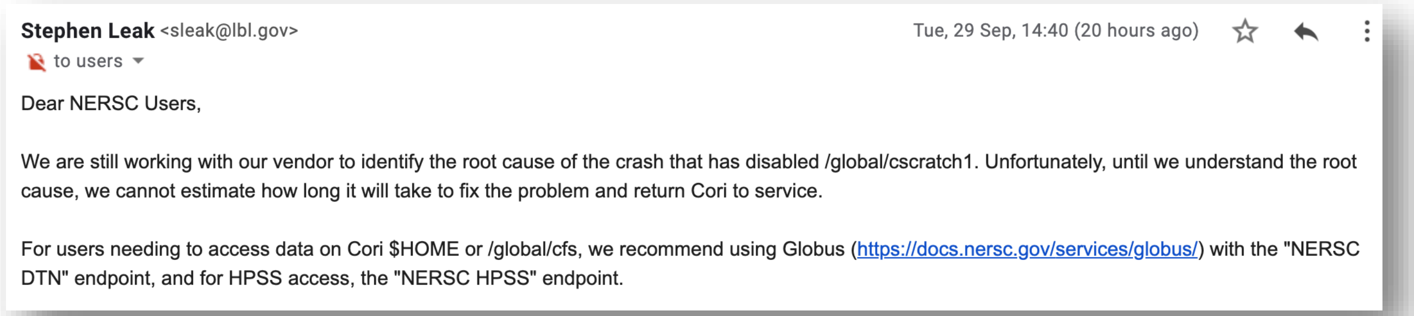
STARFISH SCHOOL 2022

Version control (but why??)

Why would you want to use Version Control?

- Multiple versions without screwing working code up
- Keep track of changes
- Use in a paper – and can cite to a specific version
- Open access and changes made by other users
- Rollback – when you screw up, you can fix!
- Collaboration

Version control



Stephen Leak <sleak@lbl.gov>
to users

Tue, 29 Sep, 14:40 (20 hours ago)

Dear NERSC Users,

We are still working with our vendor to identify the root cause of the crash that has disabled /global/cscratch1. Unfortunately, until we understand the root cause, we cannot estimate how long it will take to fix the problem and return Cori to service.

For users needing to access data on Cori $HOME or /global/cfs, we recommend using Globus (https://docs.nersc.gov/services/globus/) with the "NERSC DTN" endpoint, and for HPSS access, the "NERSC HPSS" endpoint.

COOL CATCH

Losing access to your code when a big super computer is down isn't cool at all...

Git

The basics of git

- Initializing/Cloning
- Committing
- Making a branch/Checking Out
- Pulling/Pushing
- Merging
- Conflicts

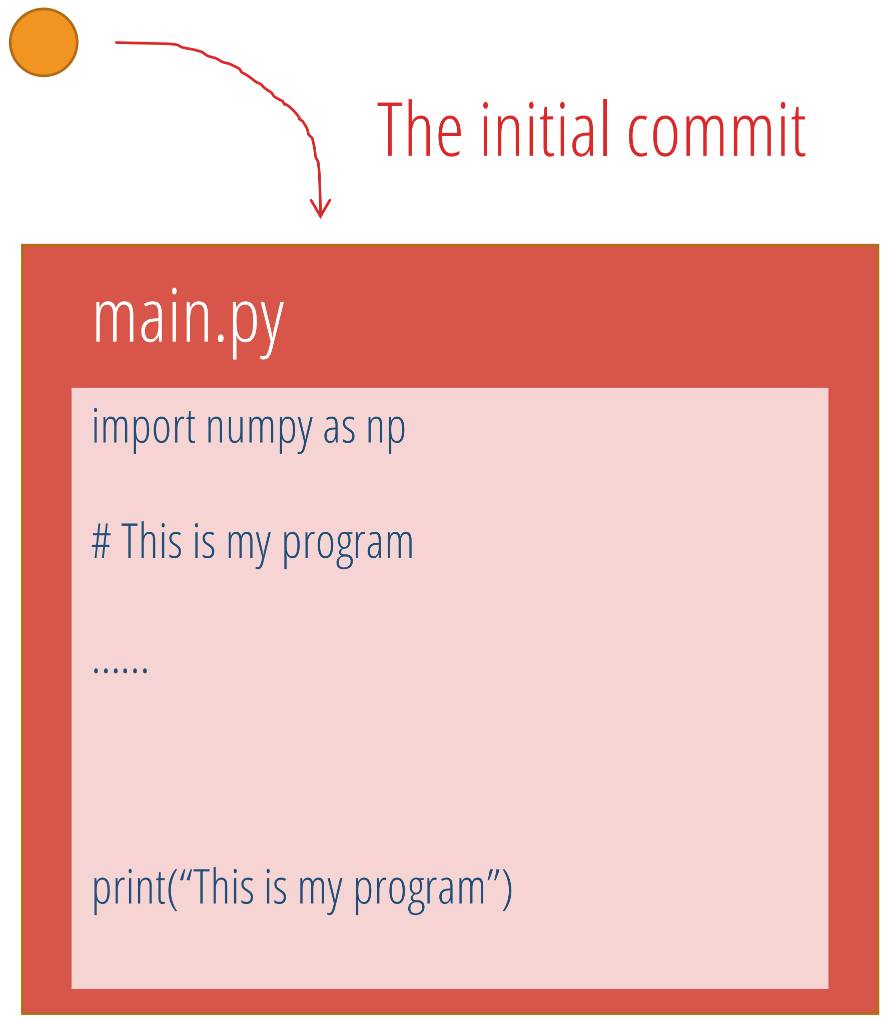
The basics of git

- Initializing/Cloning
- Committing
- Making a branch/Checking Out
- Pulling/Pushing
- Merging
- Conflicts

HOT TIP

When you start an RStudio project you can make a new git repo by default

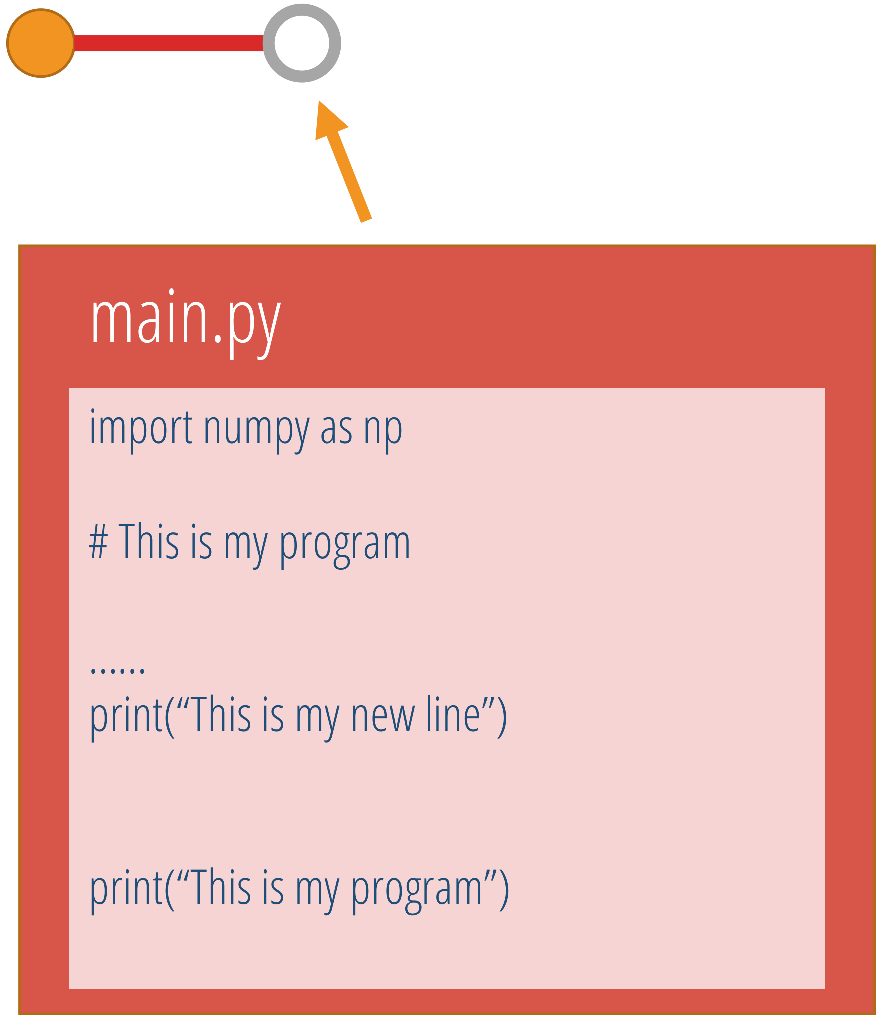
Your Git Tree



main.py

```
import numpy as np

# This is my program

......



print("This is my program")
```

Your Git Tree

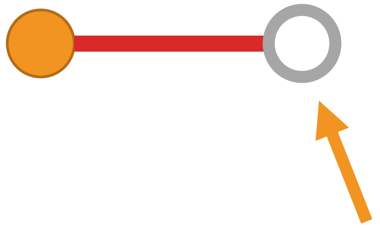


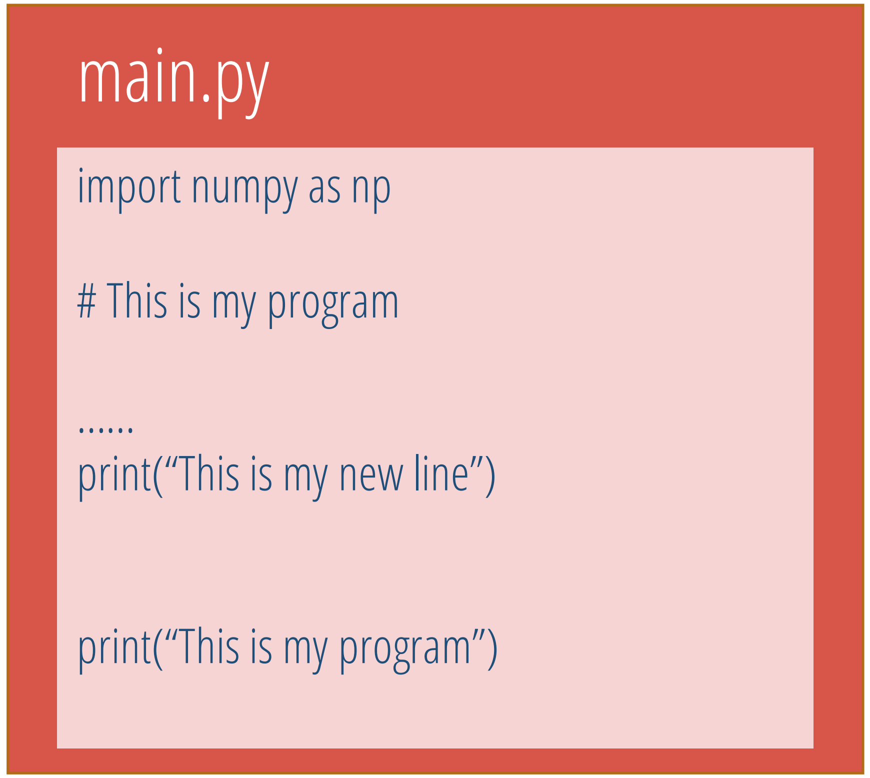
main.py

```
import numpy as np

# This is my program

......
print("This is my new line")


print("This is my program")
```

Making a change
(not staged)

Your Git Tree



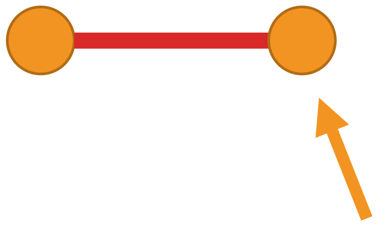


main.py

```
import numpy as np

# This is my program


......
print("This is my new line")



print("This is my program")
```

Making a change

Your Git Tree

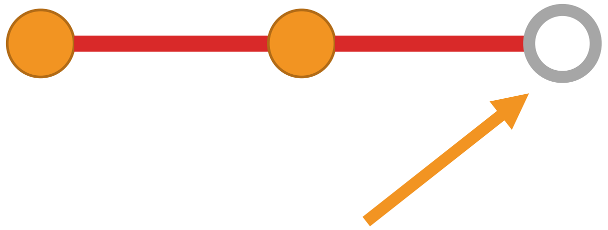


main.py

```
import numpy as np

# This is my program


......
print("This is my new line")



print("This is my program")
```

Now committed

Your Git Tree

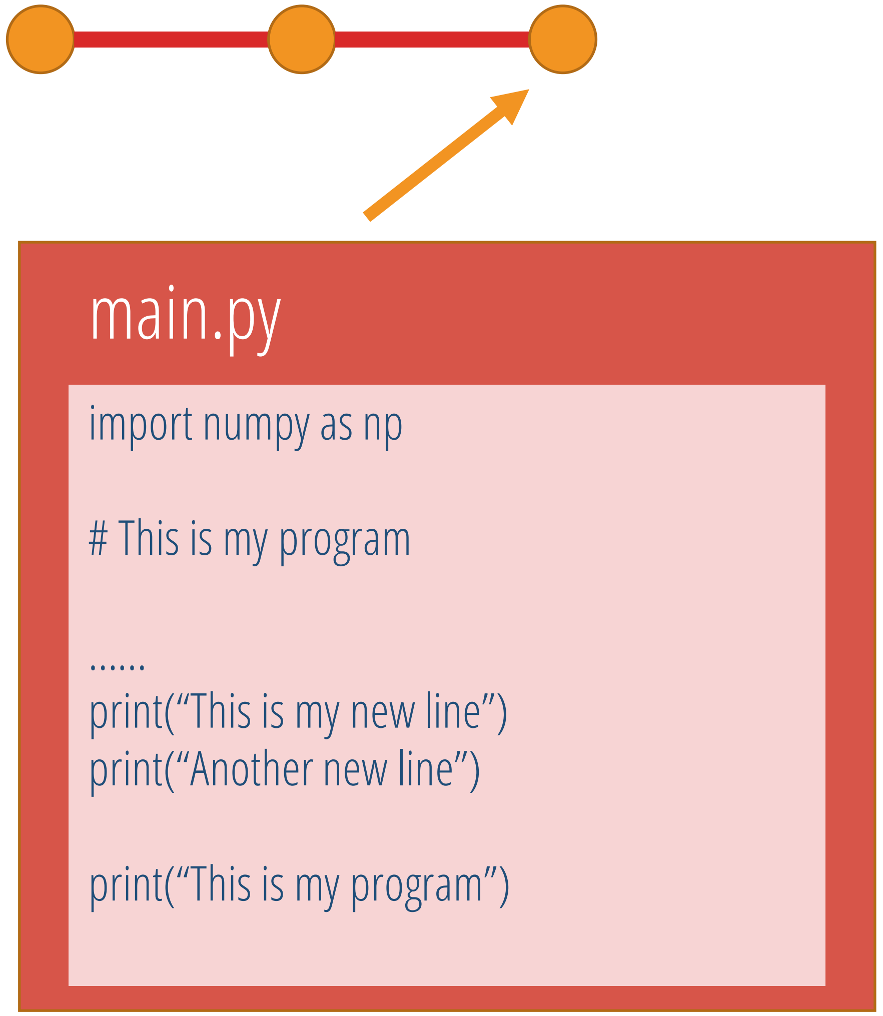


main.py

```
import numpy as np

# This is my program


......
print("This is my new line")
print("Another new line")

print("This is my program")
```

Additional change

Your Git Tree

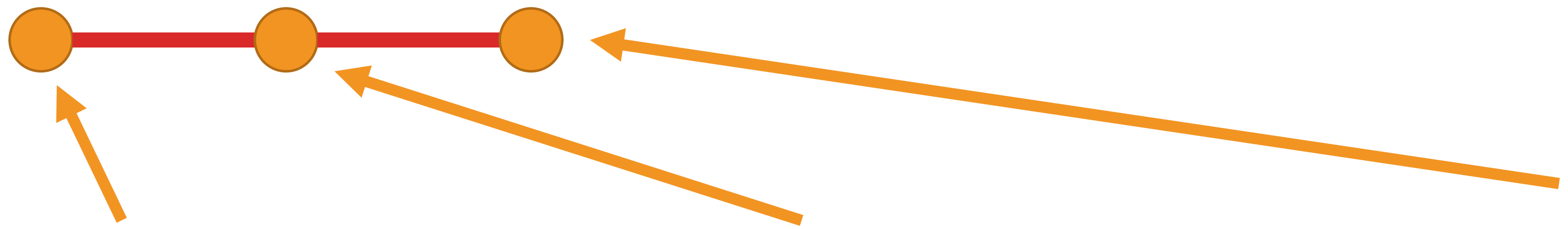


```
import numpy as np

# This is my program


......
print("This is my new line")
print("Another new line")

print("This is my program")
```

Now committed

Your Git Tree

All of the individually committed versions are stored



main.py

```
import numpy as np

# This is my program

......



print(“This is my program”)
```

main.py

```
import numpy as np

# This is my program

......
print(“This is my new line”)


print(“This is my program”)
```

main.py

```
import numpy as np

# This is my program

......
print(“This is my new line”)
print(“Another new line”)

print(“This is my program”)
```

Your Git Tree

- When you push those changes those committed changes become part of the branch.
- And the magic (or the murderous rage) happens.
- Don't leave branches unpushed.

Back to basics...

Creating a repository

From an already existing directory:

```
    git init .
```

Creates a local repository with all your local code.

Creating a repository

From an already existing directory:

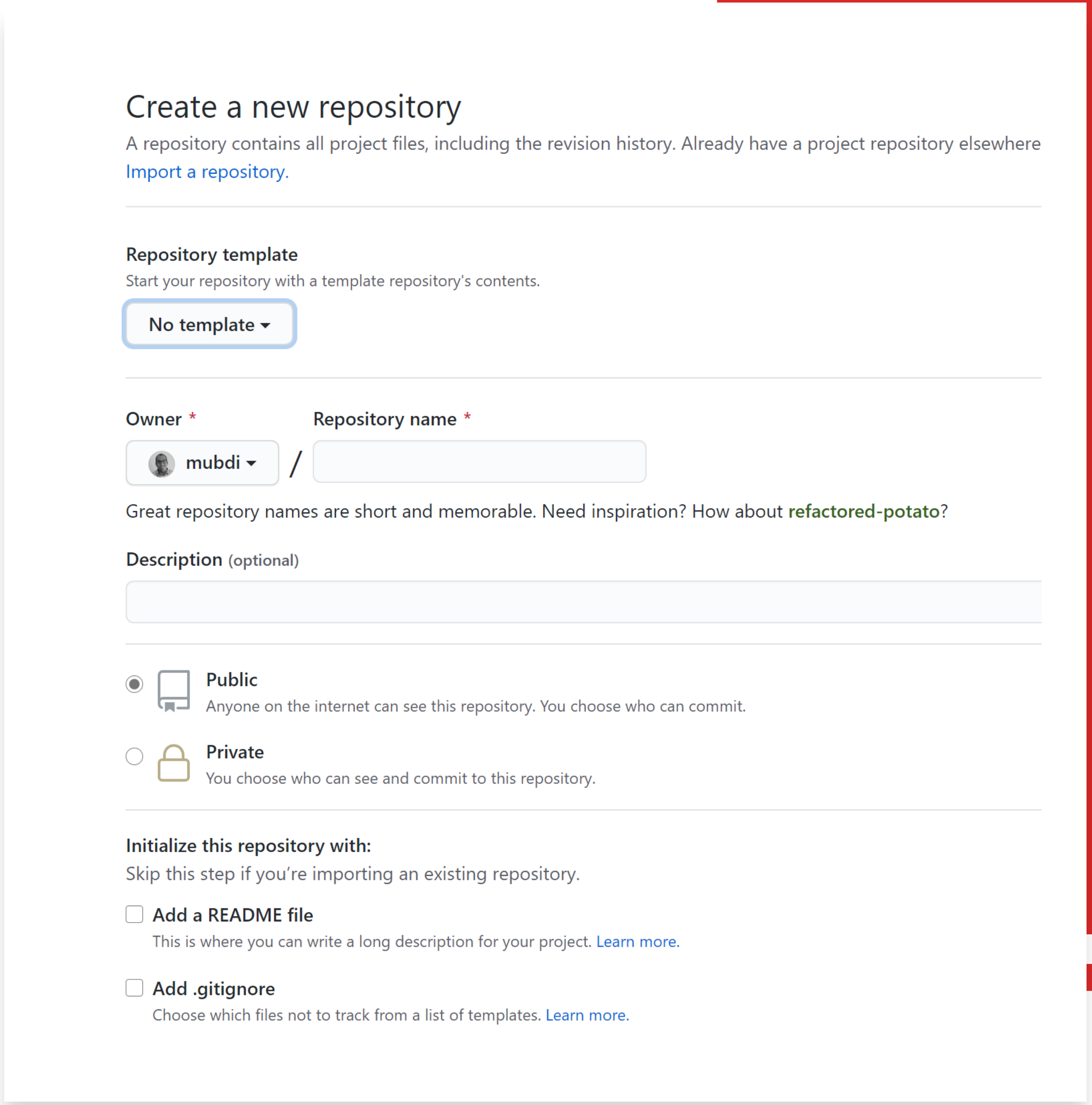
```
git init .
```

Creates a local repository with all your local code.

HOT TIP
If you have a bunch of code you've been using for years and you want to start tracking it, initialize the git repo this way

Creating a repository

Can create a new repo directly on github:



Create a new repository

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

Repository template

Start your repository with a template repository's contents.

No template

Owner * Repository name *

mubdi /

Great repository names are short and memorable. Need inspiration? How about refactored-potato?

Description (optional)

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

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

Initialize this repository with:

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

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

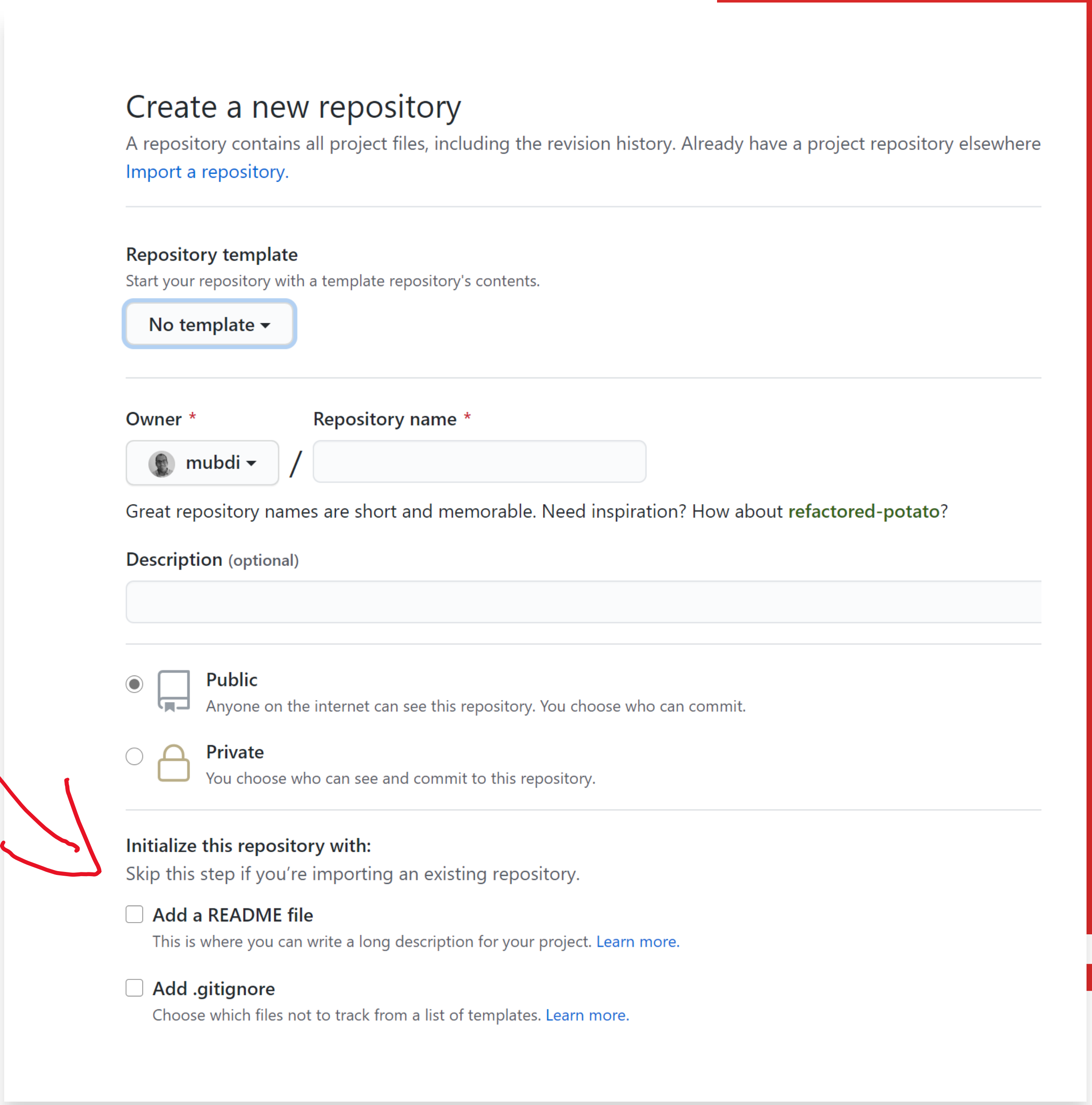
Add .gitignore
Choose which files not to track from a list of templates. Learn more.

HOT TIP

If you want to create a repo on github for an existing repository (i.e. one you have already made on your computer), make sure you don't initialize it.

Creating a repository

Can create a new repo directly on github:



Create a new repository

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

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

No template ▾

Owner * **Repository name** *

mubdi ▾ /

Great repository names are short and memorable. Need inspiration? How about **refactored-potato**?

Description (optional)

[✓ **Public**]
Anyone on the internet can see this repository. You choose who can commit.

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

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

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

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

Cloning an existing repository

Make a “local” copy of a repository:

```
git clone <URL>
```

This makes a local copy of your “remote” repository.

Cloning an existing repository

Make a “local” copy of a repository:

```
    git clone <URL>
```

This makes a local copy of your “remote” repository.

HOT TIP

You can also clone a local path or network path this way as well (i.e., if you’re storing your repo on a Department/CITA computer)

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

COOL CATCH

Git is really meant to track "small" files (think text files and code, not really large datasets.) In general, don't add large datasets to your git repo.

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

COOL CATCH

Git is really meant to track "small" files (think text files and code, not really large datasets.) In general, don't

repo.

HOT TIP

If you accidentally added large files to your git repo, check out conflict resolution that we'll talk about in a bit.

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

HOT TIP

To see what state your repository is in, use the command `git status`

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename>
```

or to track everything:

```
git add *
```

HOT TIP

Sometimes you have files in the directory. You can tell git that you never want to add them by adding them to a **.gitignore** file

Adding and Tracking Files

For git to track any files, you need to **add** the files to the repository:

```
git add <filename
```

or to track everything:

```
git add *
```

HOTTER TIP

You don't even need to make your own .gitignore file! You can get pre-made templates for most languages here: https://github.com/github/gitignore

Sometimes you have files in the directory. You can tell git that you never want to add them by adding them to a **.gitignore** file

Committing Changes

Once you've added files, you can commit that change using

```
    git commit <filename>
```

or for all files that have been changed/added:

```
    git commit -a
```

which will open your default editor to add a message to describe your change.

Committing Changes

Once you've added files, you can commit that change using

```
git commit <filename>
```

or for all files that have been changed/added:

```
git commit -a
```

which will open your default editor to add a message to describe your change.

HOT TIP

You can specify your message on the command line using

```
git commit -a –m
"Your Message"
```

Committing Changes

Once you've added files, you can commit that change using

```
git commit <filename>
```

or for all files that have been changed/added:

```
git commit -a
```

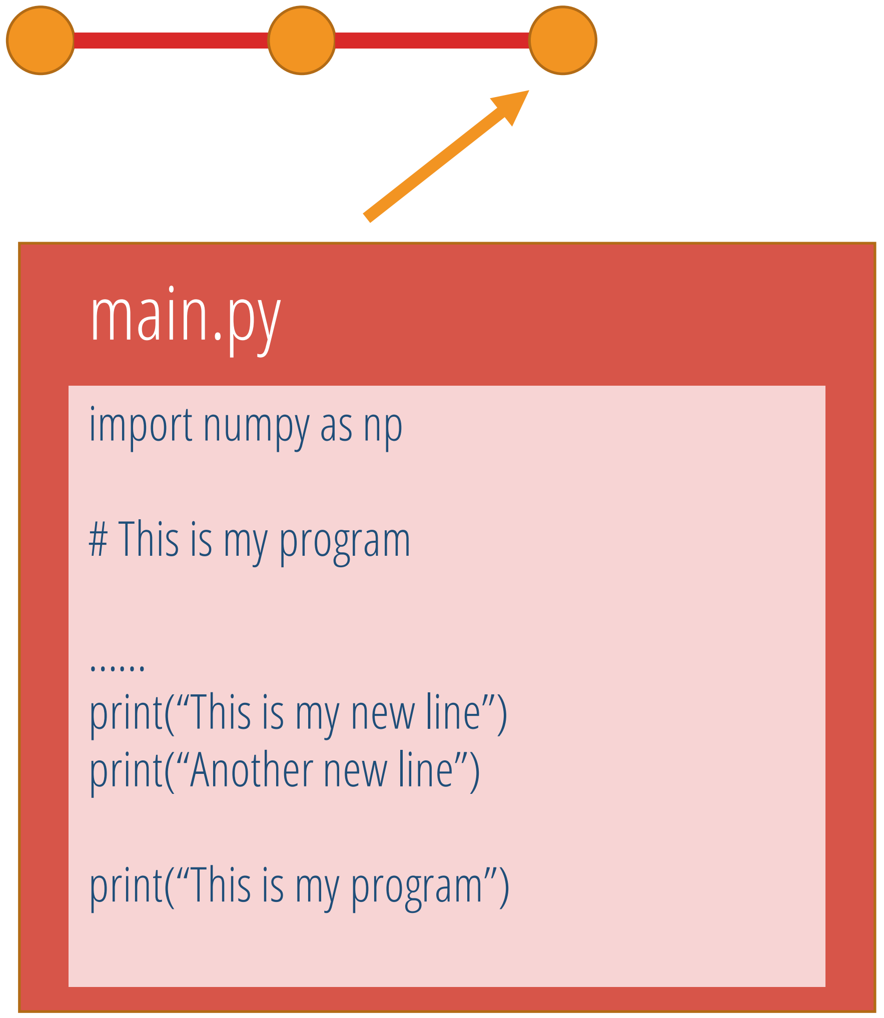
which will open your default editor to add a message to describe your change.

HOT TIP

Commit often! Just do it! Don't worry if things aren't perfect!

Git Branches

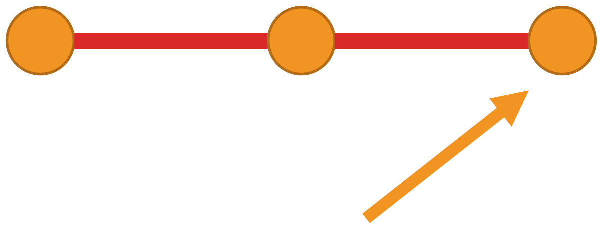
Main Branch



main.py

```
import numpy as np

# This is my program


......
print("This is my new line")
print("Another new line")

print("This is my program")
```

Git Branches



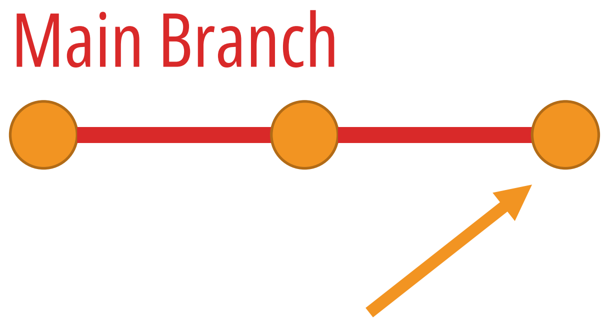
main.py

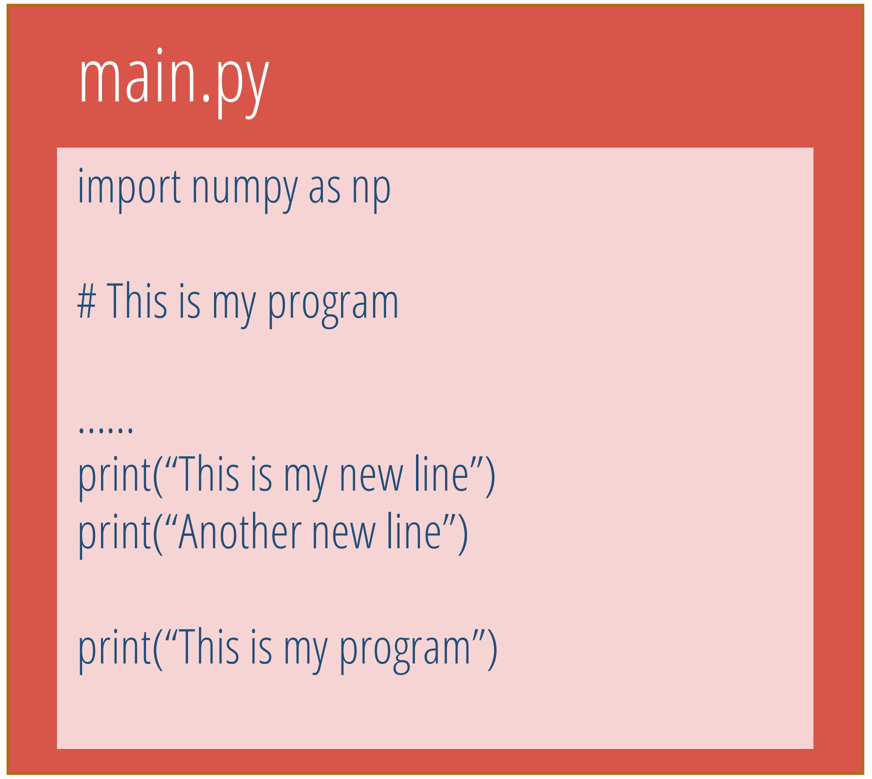
```
import numpy as np

# This is my program


......
print("This is my new line")
print("Another new line")

print("This is my program")
```

COOL CATCH

Previously, the primary branch in git used to be referred to by the problematic term "master". You may see this terminology still every once in a while.

Git Branches



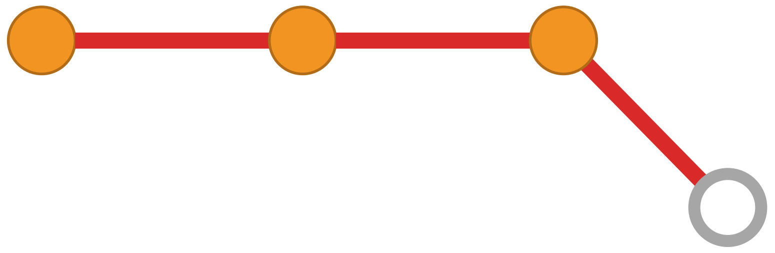


```
import numpy as np

# This is my program


……
print("This is my new line")
print("Another new line")

print("This is my program")
```

Sometimes, you'll want to work on something separate from your working code. You can create a "branch"

Git Branches

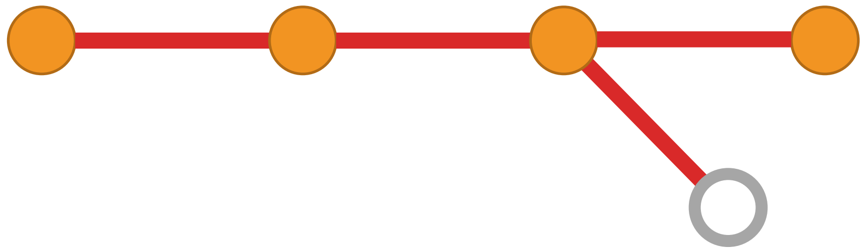
Main Branch



New Branch

Git Branches

Main Branch

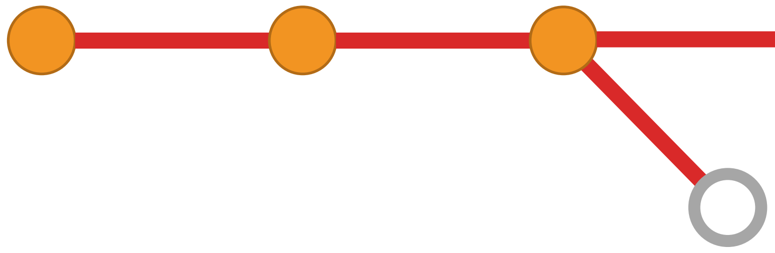


New Branch

Life continues on the main branch

Git Branches

Main Branch

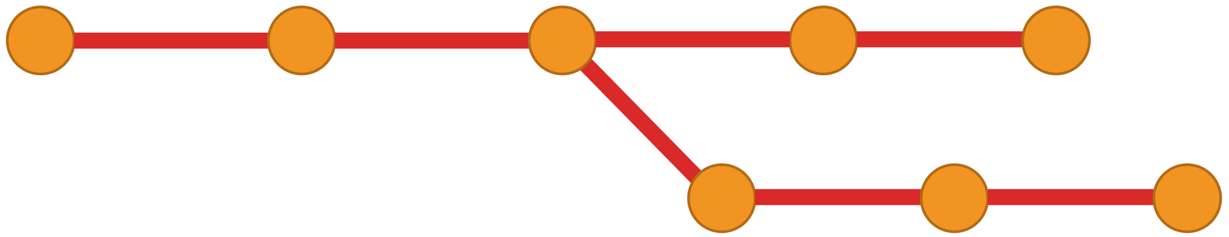


New Branch

Life continues on the main branch

Git Branches

Main Branch

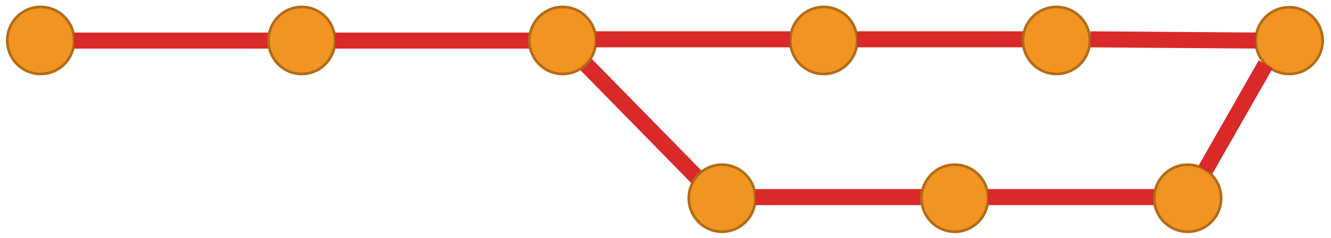


New Branch

But you can continue developing on the new branch by “checking it out” and committing as usual

Git Branches

Main Branch

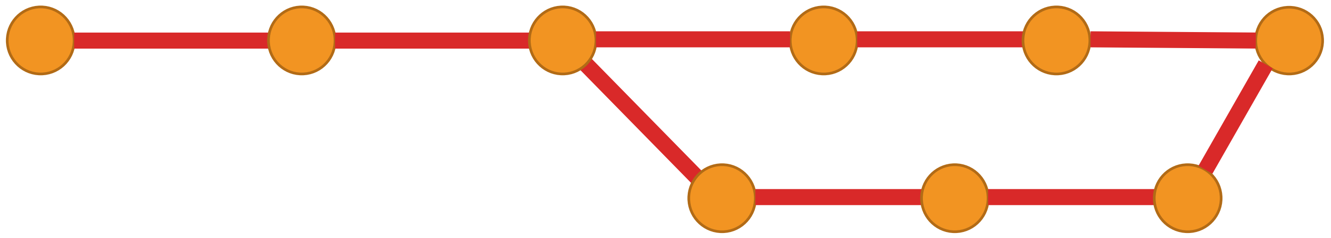


New Branch

Once you're done with your new branch, you can **merge** it back to the main branch

Git Branches

Main Branch



New Branch

Once you’re done with your new branch, you can **merge** it back to the main branch

If there are changes that can’t be merged together, there’s a conflict!

Creating a new branch and checking it out

You can create a new branch using the command line:

```
git branch <branch_name>
```

once you've created the branch, you can move to it by checking it out:

```
git checkout <branch_name>
```

Creating a new branch and checking it out

You can create a new branch using the command line:

```
git branch <branch_name>
```

once you've created the branch, you can move to it by checking it out:

```
git checkout <branch_name>
```

HOT TIP

You can see which branch you're on with `git status`

Creating a new branch and checking it out

You can create a new branch using the command line:

```
git branch <branch_name>
```

once you've created the branch, you can move to it by checking it out:

```
git checkout <branch_name>
```

HOT TIP

You can see all of your available branches by

```
git branch -a
```

Merging back to the main

You can change back to the main branch:

```
git checkout main
```

and you can merge your old branch:

```
git merge <branch_name>
```

Break!

Mergin th

You can chang

and

Dealing with conflicts

When you merge, on occasion the branches will have a conflict. Git will tell you about it. What do you do?

1. Git will tell you about it, and save both versions in the same file
2. Fix the file and save it
3. Commit the new version.
4. Be proud that you defeated the conflict!

COOL CATCH
If you are using a terminal rather than an IDE you *might* get the conflict message as a vi file.

Dealing with conflicts

When you merge, on occasion the branches will have a conflict. Git will tell you about it. What do you do?

1. Git will tell you about it, and save both versions in the same file
2. Fix the file and save it
3. Commit the new version.
4. Be proud that you defeated the conflict!

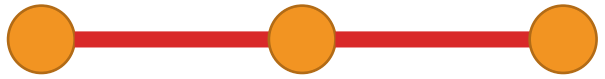
Demo Time!
Real Life Git Conflict

COOL CATCH
If you are using a terminal rather than an IDE you *might* get the conflict message as a vi file.

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

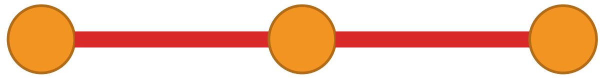
Local Repository

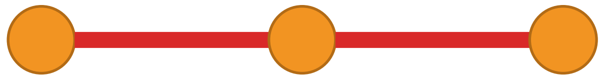


Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.

Local Repository

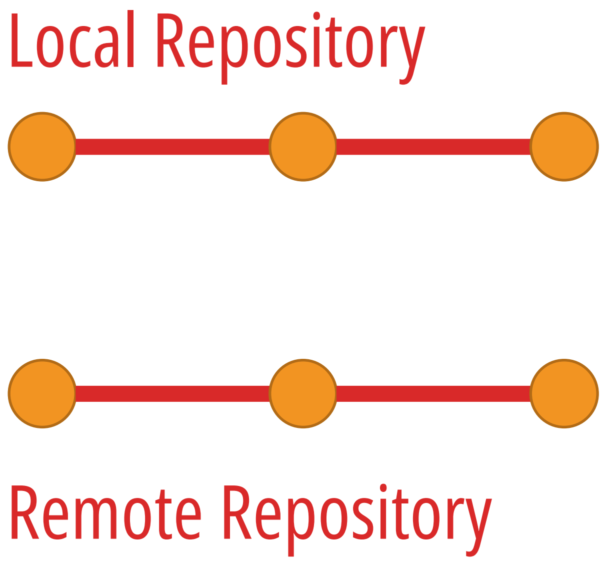




Remote Repository

Github and Remote Git Repositories

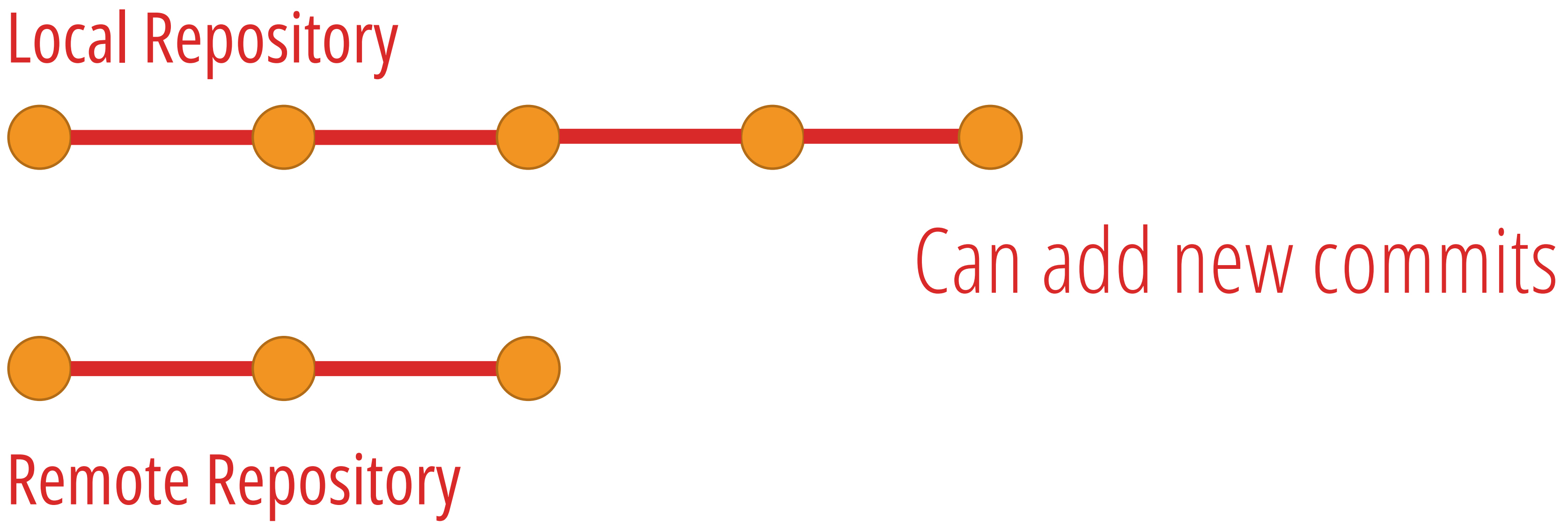
Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.



COOL CATCH
GitHub, while popular, isn't the only remote/cloud git service. You may also see people using BitBucket or GitLab, amongst others

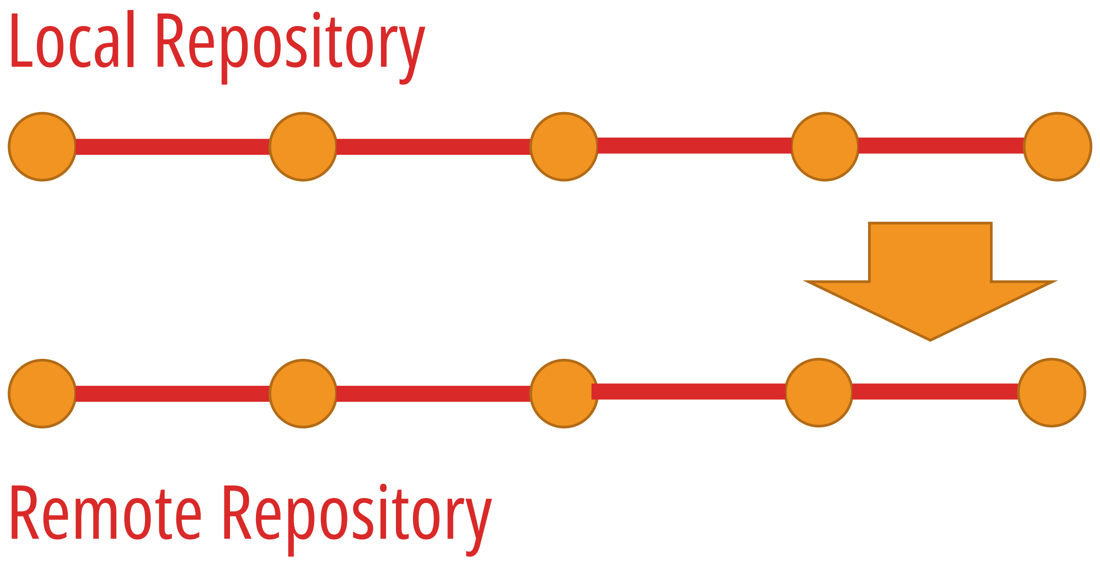
Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.



Github and Remote Git Repositories

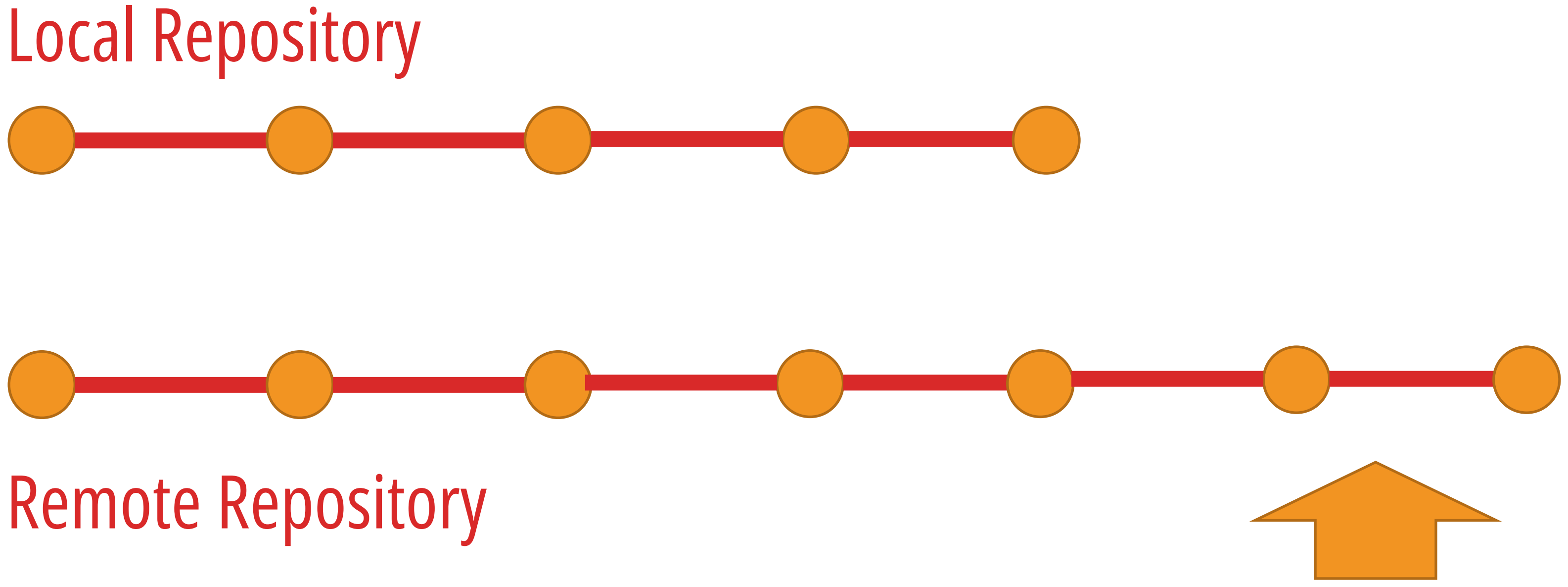
Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.



Can then **`git push`** them back to the remote

Github and Remote Git Repositories

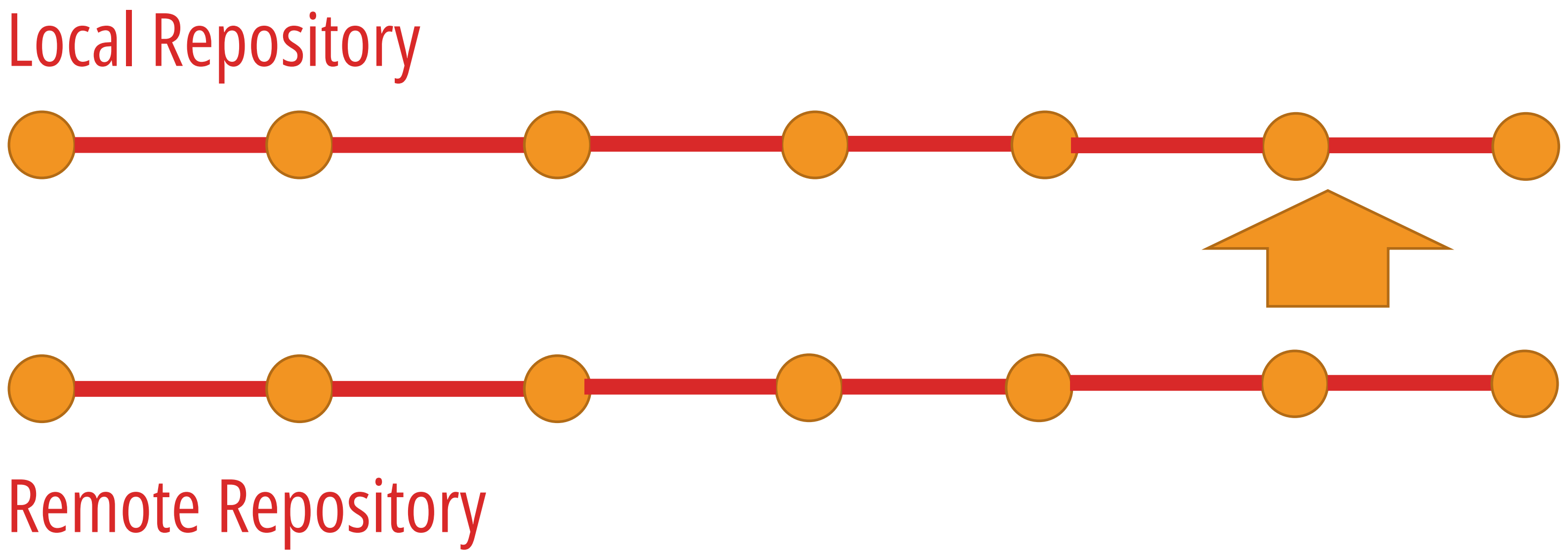
Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.



Other people can push to the remote repo

Github and Remote Git Repositories

Every git repo keeps a full history of all commits. But you can create a centralized location for the repository, where multiple people can contribute.



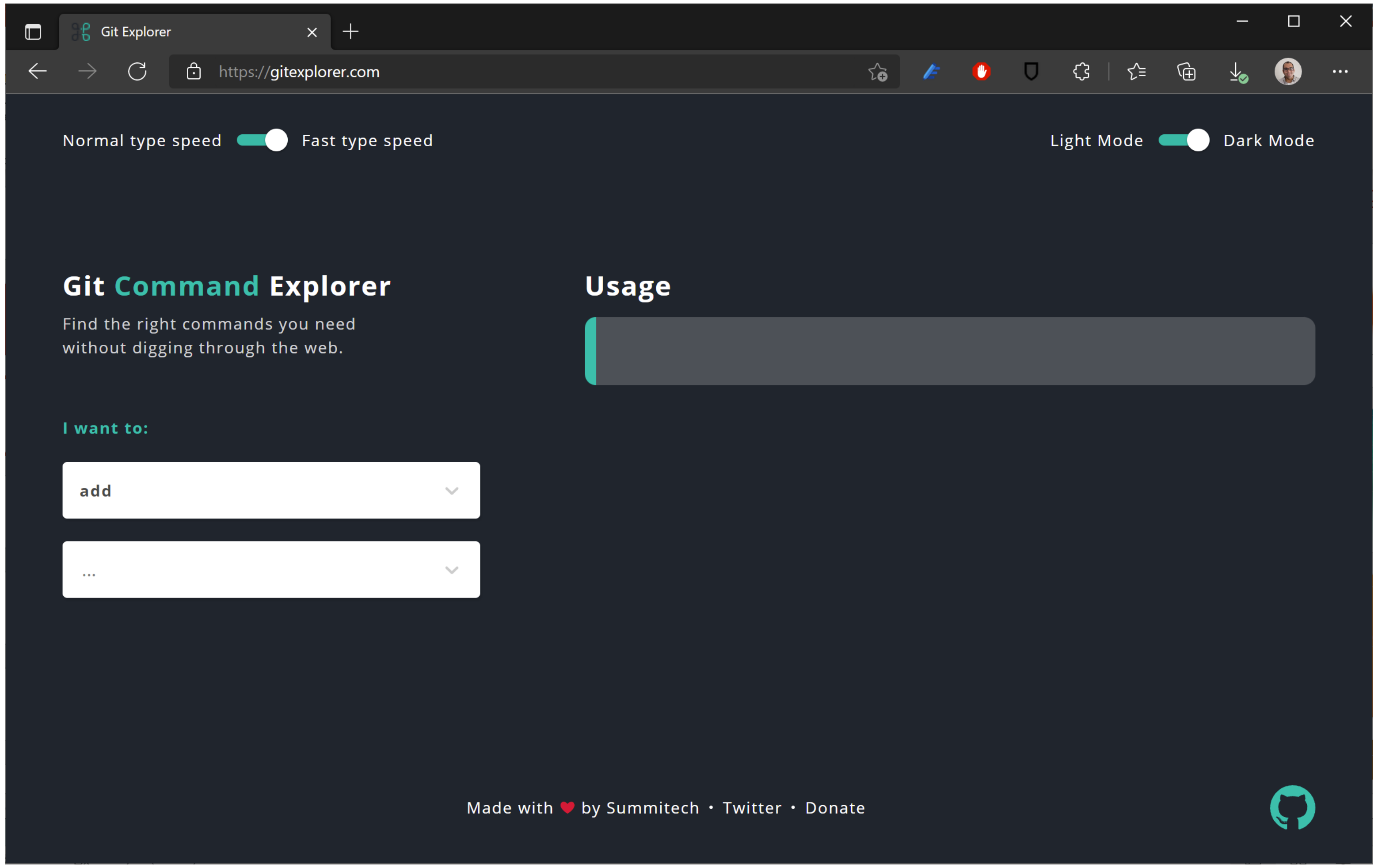
Can then **`git pull`** them back to your local repo

Useful References for Git

- Software Carpentry (intro to version control with git)
  - https://swcarpentry.github.io/git-novice/
- Atlassian Tutorials
  - https://www.atlassian.com/git/tutorials/what-is-version-control
- Git Cheat Sheet from Atlassian
  - https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- The Simple Guide
  - https://rogerdudler.github.io/git-guide/

I don’t usually remember all git commands. It’s a bit of a waste of time.

Git Command Explorer



Git Explorer

https://gitexplorer.com

Normal type speed · Fast type speed

Light Mode · Dark Mode

Git Command Explorer

Find the right commands you need without digging through the web.

Usage

I want to:

add

...

Made with ♥ by Summitech • Twitter • Donate

http://gitexplorer.com

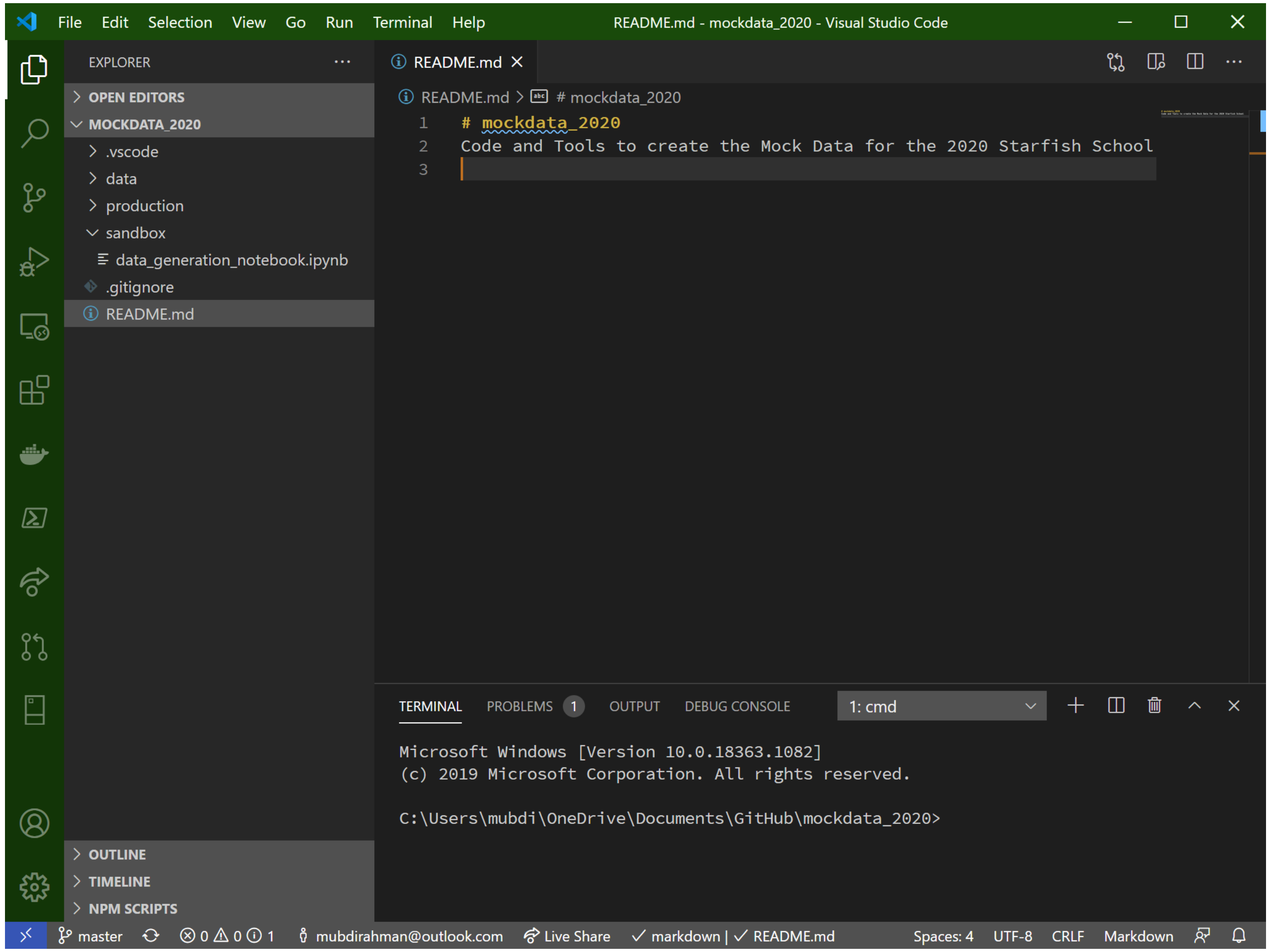
Using a Code Editor: VS Code

Made for efficient coding practices

Takes care of git right through the environment

Linting, autocomplete, error checking

Helps with debugging (come back to this place next week)


File Edit Selection View Go Run Terminal Help
README.md - mockdata_2020 - Visual Studio Code
EXPLORER
OPEN EDITORS
MOCKDATA_2020
.vscode
data
production
sandbox
data_generation_notebook.ipynb
.gitignore
README.md
OUTLINE
TIMELINE
NPM SCRIPTS
README.md
README.md > # mockdata_2020
TERMINAL PROBLEMS 1 OUTPUT DEBUG CONSOLE
1: cmd
master ⊗ 0 ⚠ 0 ⓘ 1 mubdirahman@outlook.com Live Share ✓ markdown | ✓ README.md Spaces: 4 UTF-8 CRLF Markdown

```
# mockdata_2020
Code and Tools to create the Mock Data for the 2020 Starfish School
```

```
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\mubdi\OneDrive\Documents\GitHub\mockdata_2020>
```

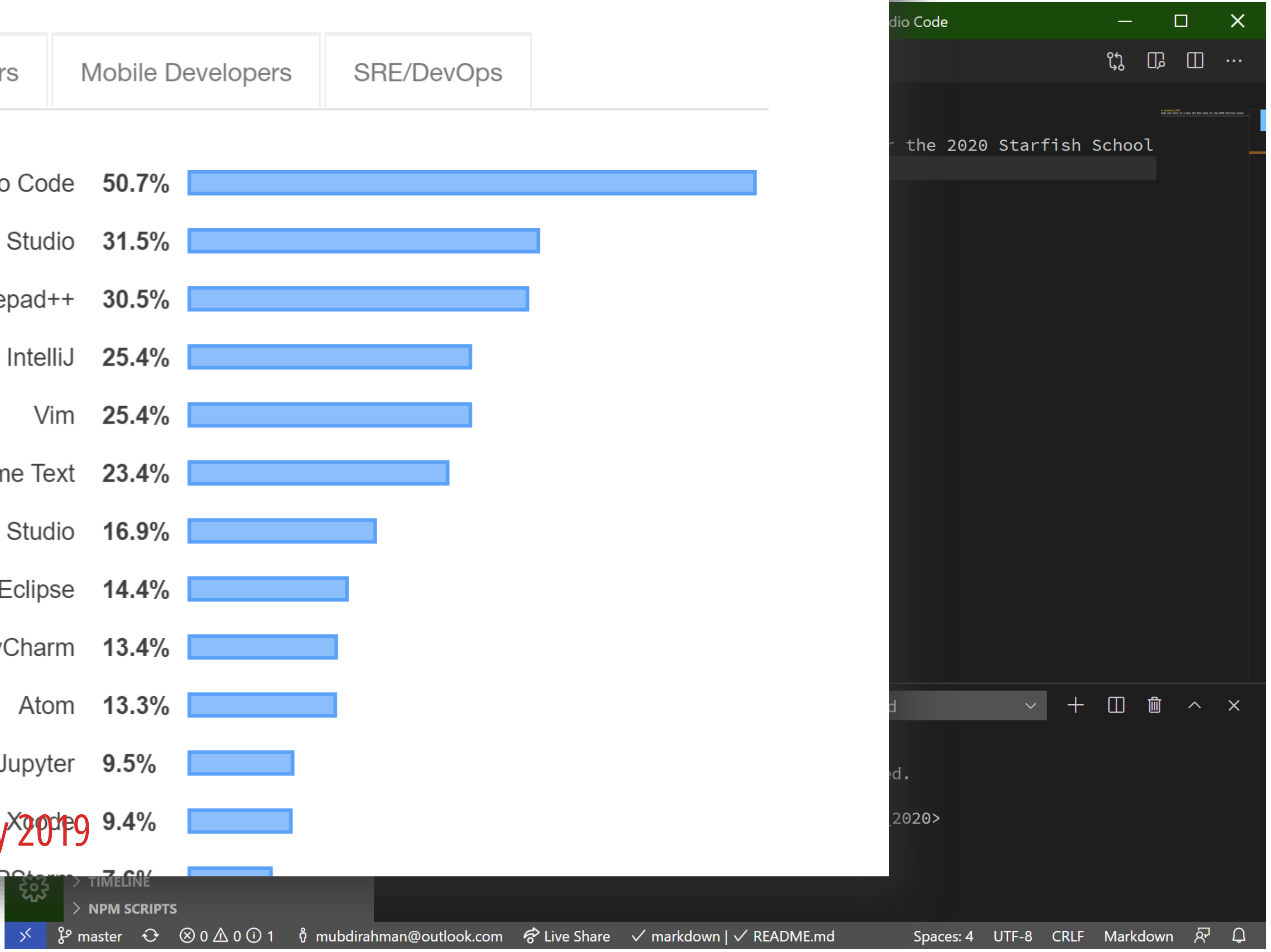
Using a Code Editor: VS Code

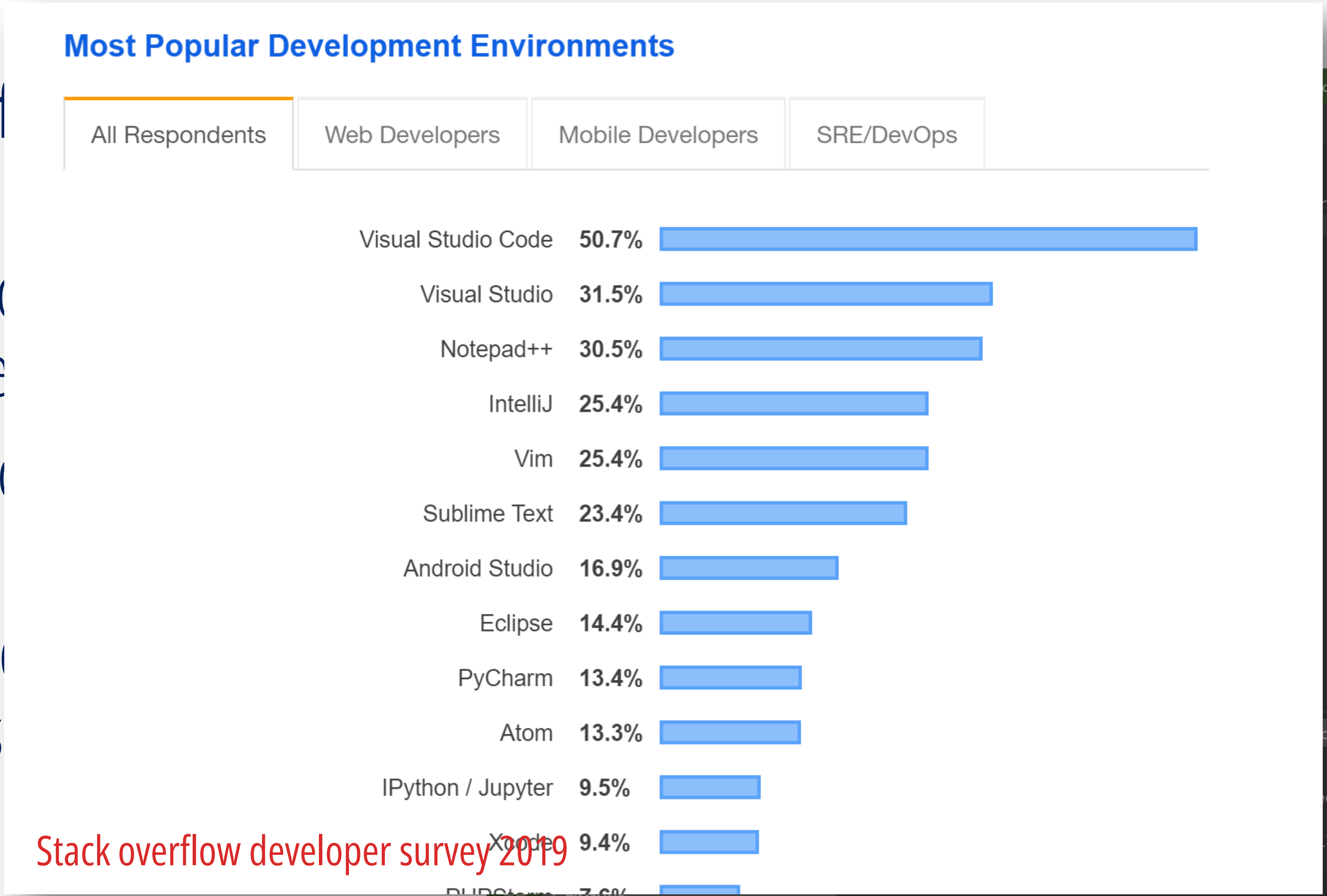
Made for ef
practices

Takes care (
through the

Linting, auto
checking

Helps with (
back to this





Most Popular Development Environments

All Respondents | Web Developers | Mobile Developers | SRE/DevOps

Visual Studio Code	50.7%
Visual Studio	31.5%
Notepad++	30.5%
IntelliJ	25.4%
Vim	25.4%
Sublime Text	23.4%
Android Studio	16.9%
Eclipse	14.4%
PyCharm	13.4%
Atom	13.3%
IPython / Jupyter	9.5%
Xcode	9.4%

Stack overflow developer survey 2019

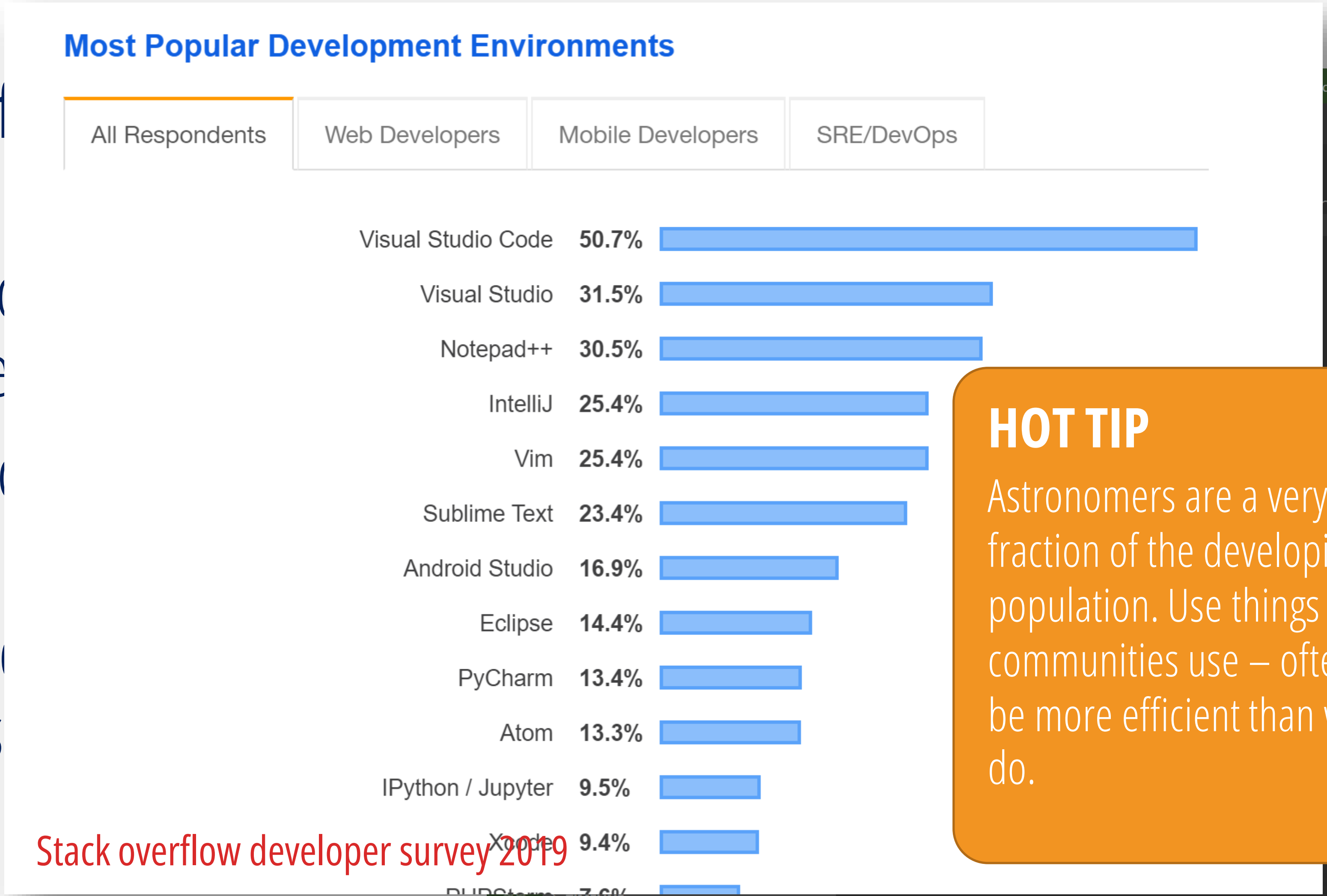
Using a Code Editor: VS Code

Made for ef
practices

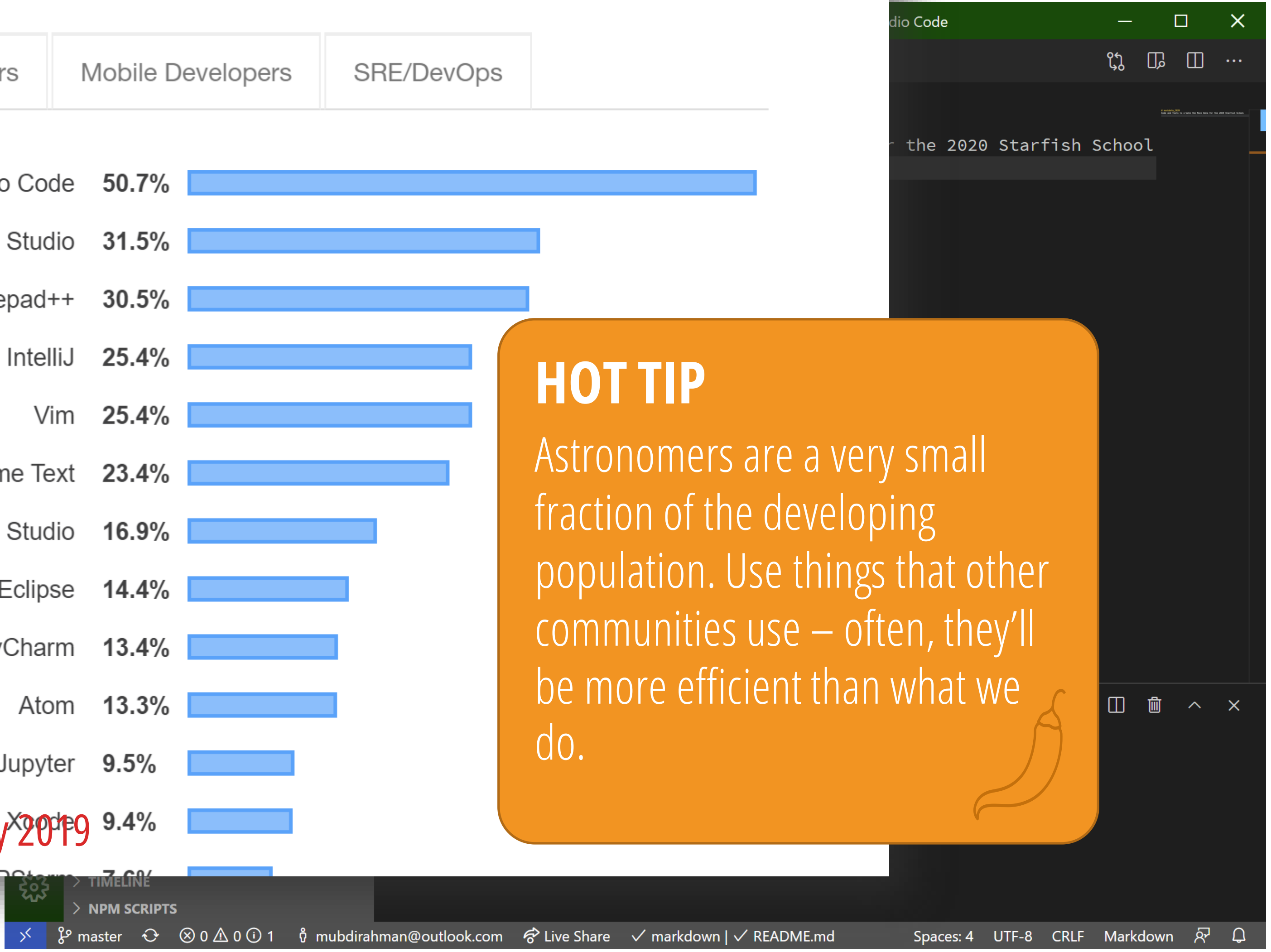
Takes care o
through the

Linting, aut
checking

Helps with o
back to this



Stack overflow developer survey 2019



HOT TIP

Astronomers are a very small fraction of the developing population. Use things that other communities use – often, they'll be more efficient than what we do.

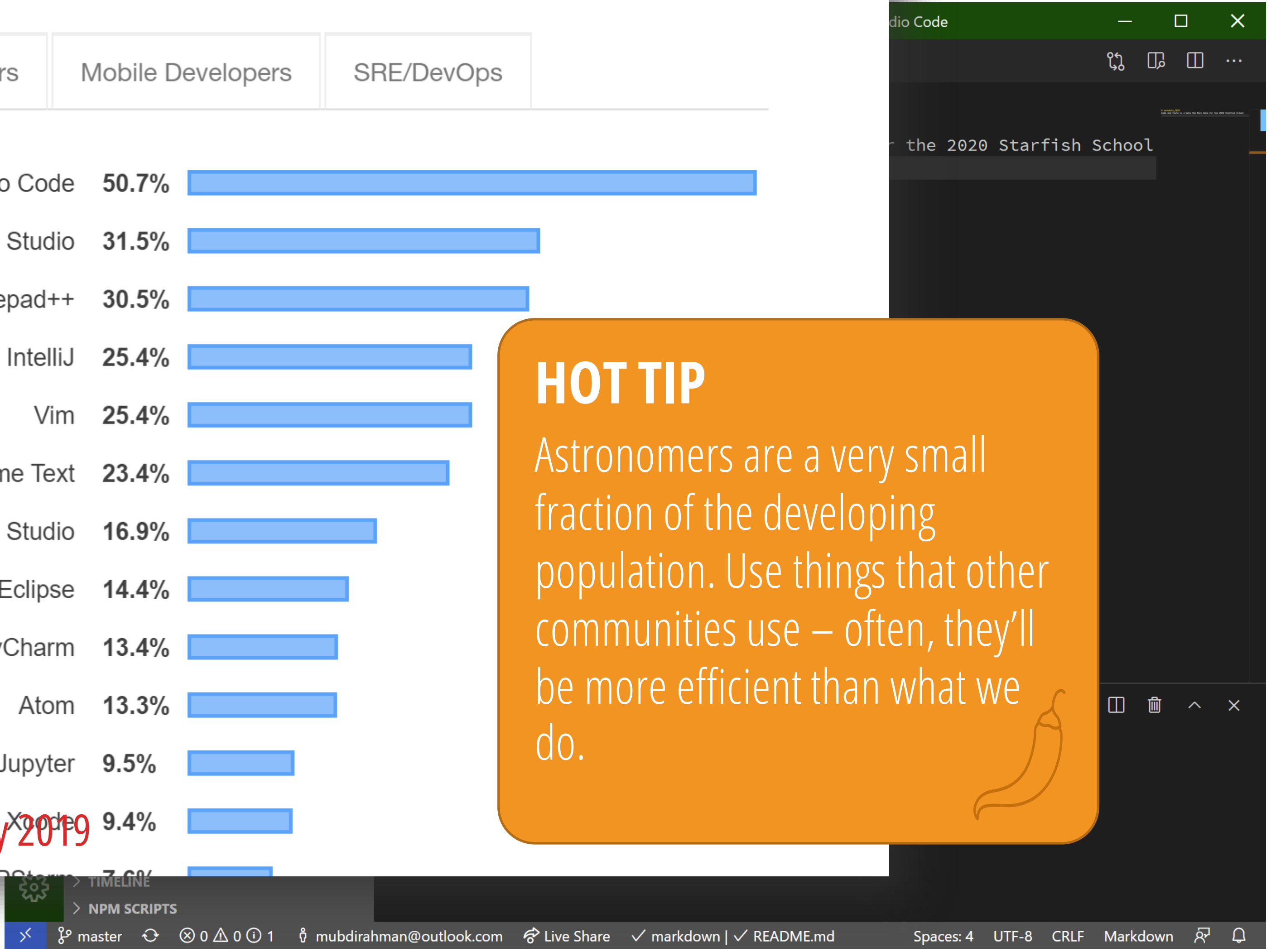
Using a Code Editor: VS Code

Made for ef
practices

Takes care
through the

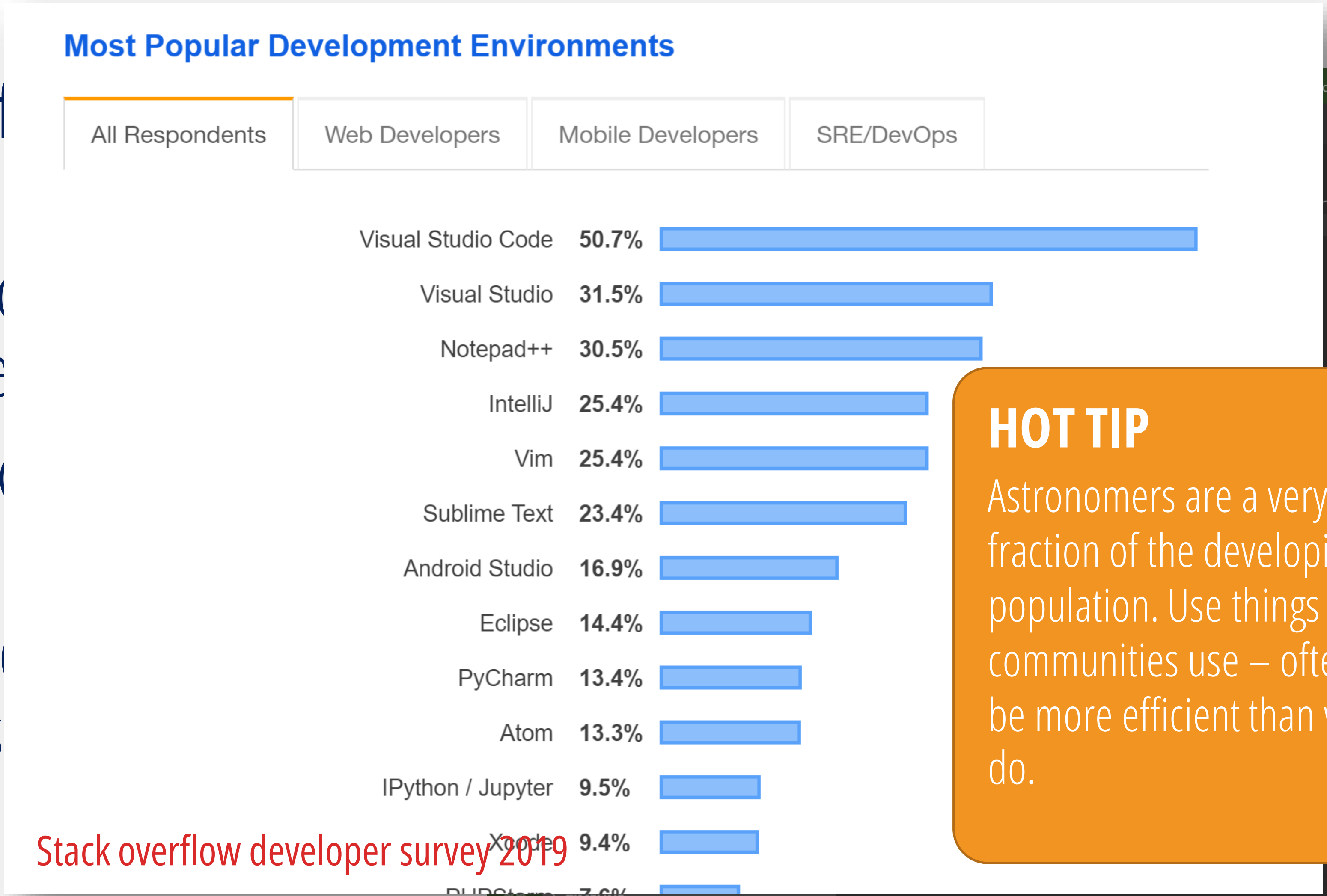
Linting, aut
checking

Helps with
back to this



the 2020 Starfish School

```
master  ⊗ 0 ⚠ 0 ⓘ 1  mubdirahman@outlook.com  Live Share  ✓ markdown | ✓ README.md  Spaces: 4  UTF-8  CRLF  Markdown
```



Stack overflow developer survey 2019

HOT TIP

Astronomers are a very small fraction of the developing population. Use things that other communities use – often, they'll be more efficient than what we do.

Demo Time!
VS Code

Exercise

You've tried these before, but let's do it again:

- Create a directory with a couple of new python files.
- Initialize a git repository within the directory
- Make changes to the files and commit them to your repository
- Make a new branch and commit a new change to your python files
- Merge the new branch down to the main branch
- Check out the git log to see all of your commits

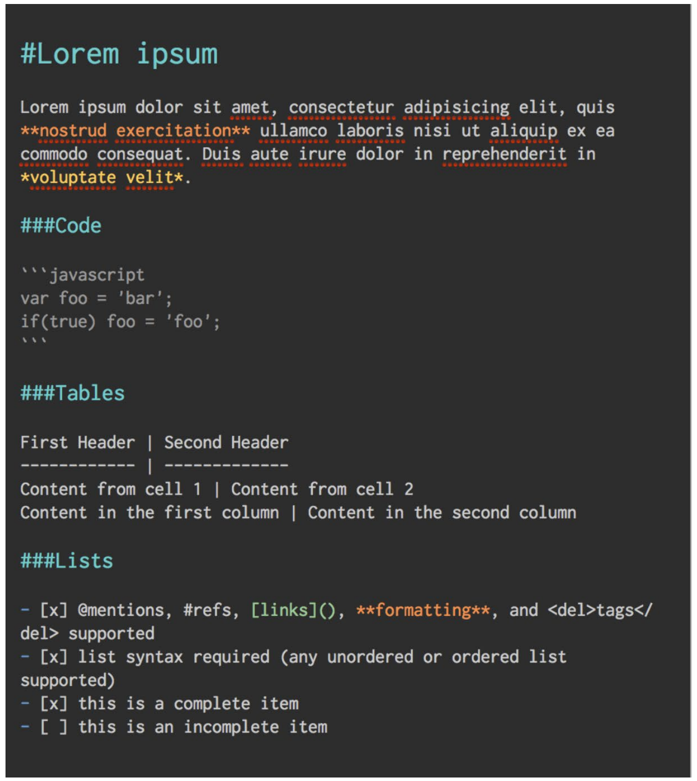
Exercise

- Create a new uninitialized git repository on github
- Push your repository to the new repository
- Clone the remote repo to a new folder
- Create a conflict: make changes to both the old and new repository
- Fix the conflict and push everything back to the remote

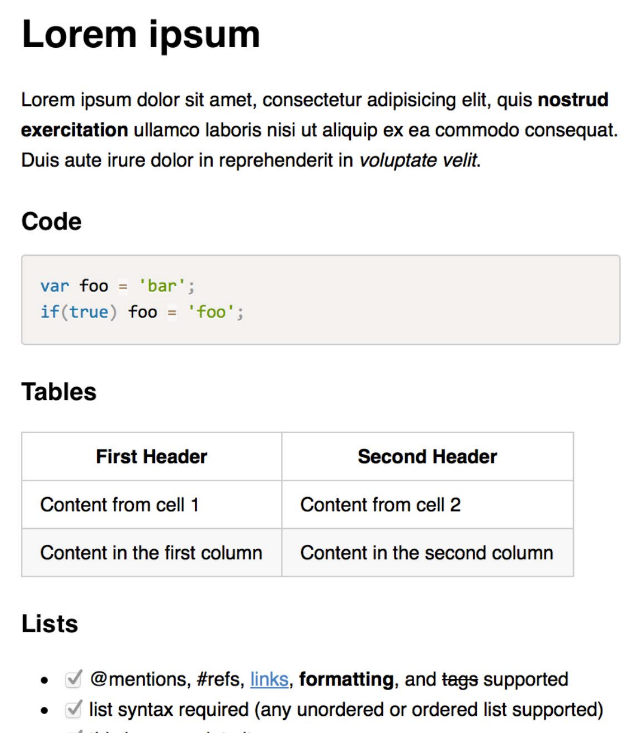
Markdown

Getting down with Markdown

- Common, simple, structured way of writing in plain text files
- Easily interpretable, supports images, tables, links, code
- Natively supported in GitHub, and even Slack!



~~~
#Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, quis
**nostrud exercitation** ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in
*voluptate velit*.

###Code

```javascript
var foo = 'bar';
if(true) foo = 'foo';
```

###Tables

First Header | Second Header
------------ | -------------
Content from cell 1 | Content from cell 2
Content in the first column | Content in the second column

###Lists

- [x] @mentions, #refs, [links](), **formatting**, and <del>tags</
del> supported
- [x] list syntax required (any unordered or ordered list
supported)
- [x] this is a complete item
- [ ] this is an incomplete item
~~~



Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, quis **nostrud exercitation** ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in *voluptate velit*.

Code

```
var foo = 'bar';
if(true) foo = 'foo';
```

Tables

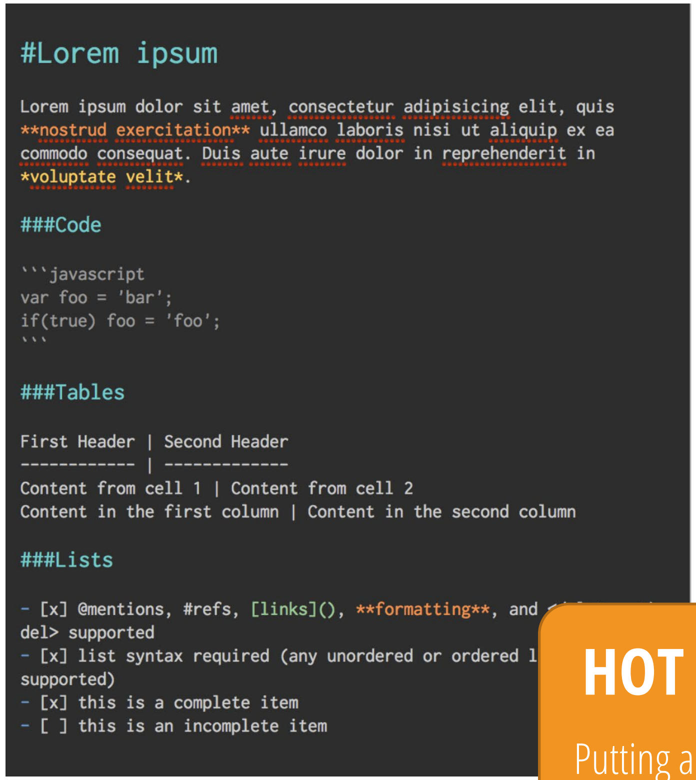
First Header	Second Header
Content from cell 1	Content from cell 2
Content in the first column	Content in the second column

Lists

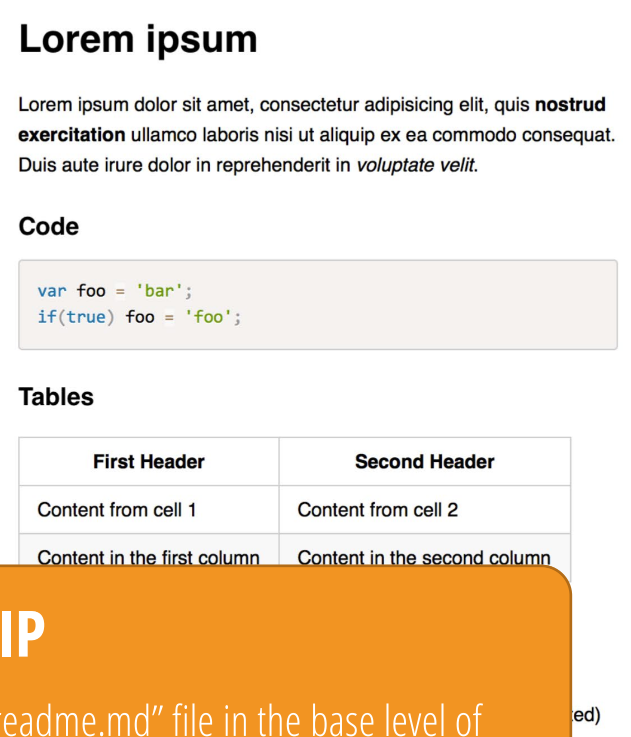
- ✓ @mentions, #refs, links, **formatting**, and ~~tags~~ supported
- ✓ list syntax required (any unordered or ordered list supported)

Getting down with Markdown

- Common, simple, structured way of writing in plain text files
- Easily interpretable, supports images, tables, links, code
- Natively supported in GitHub, and even Slack!



~~~
#Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, quis
**nostrud exercitation** ullamco laboris nisi ut aliquip ex ea
commodo consequat. Duis aute irure dolor in reprehenderit in
*voluptate velit*.

###Code

```javascript
var foo = 'bar';
if(true) foo = 'foo';
```

###Tables

First Header | Second Header
------------ | -------------
Content from cell 1 | Content from cell 2
Content in the first column | Content in the second column

###Lists

- [x] @mentions, #refs, [links](), **formatting**, and
del> supported
- [x] list syntax required (any unordered or ordered l
supported)
- [x] this is a complete item
- [ ] this is an incomplete item
~~~



Lorem ipsum

Lorem ipsum dolor sit amet, consectetur adipisicing elit, quis **nostrud exercitation** ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in *voluptate velit*.

Code

```javascript
var foo = 'bar';
if(true) foo = 'foo';
```

Tables

First Header	Second Header
Content from cell 1	Content from cell 2
Content in the first column	Content in the second column

ed)

HOT TIP

Putting a “readme.md” file in the base level of your GitHub repository makes the contents of that file appear when you look at the repository on the web

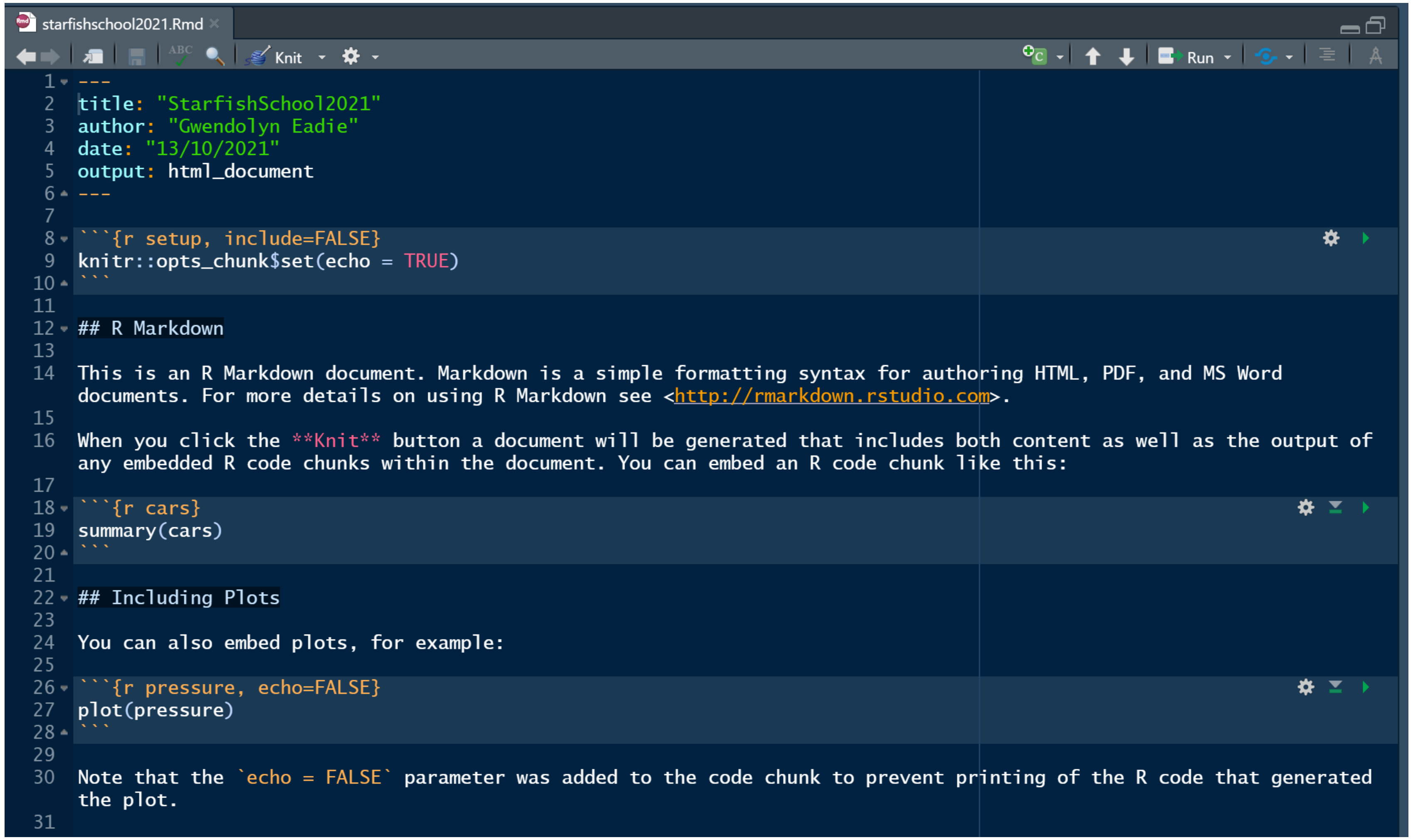
Some Basic Markup

```
# This is a heading
## This is a smaller heading
### This is an even smaller heading

* This is list item 1. _I am underlined_
* This is list item 2. **I Can Be Bold**
* I want to put a [Link In
Here](http://www.link.com)

![Image](image.png)
```

A Little More Markup

~~~
> This is something that’s in a quote


```

A little bit of code can go in here
```


# A Table
| Col 1 | Col2 | Col 3 |
| --- | --- | --- |
| a | b | c |
| a | b | c |
~~~

R Markdown in R Studio

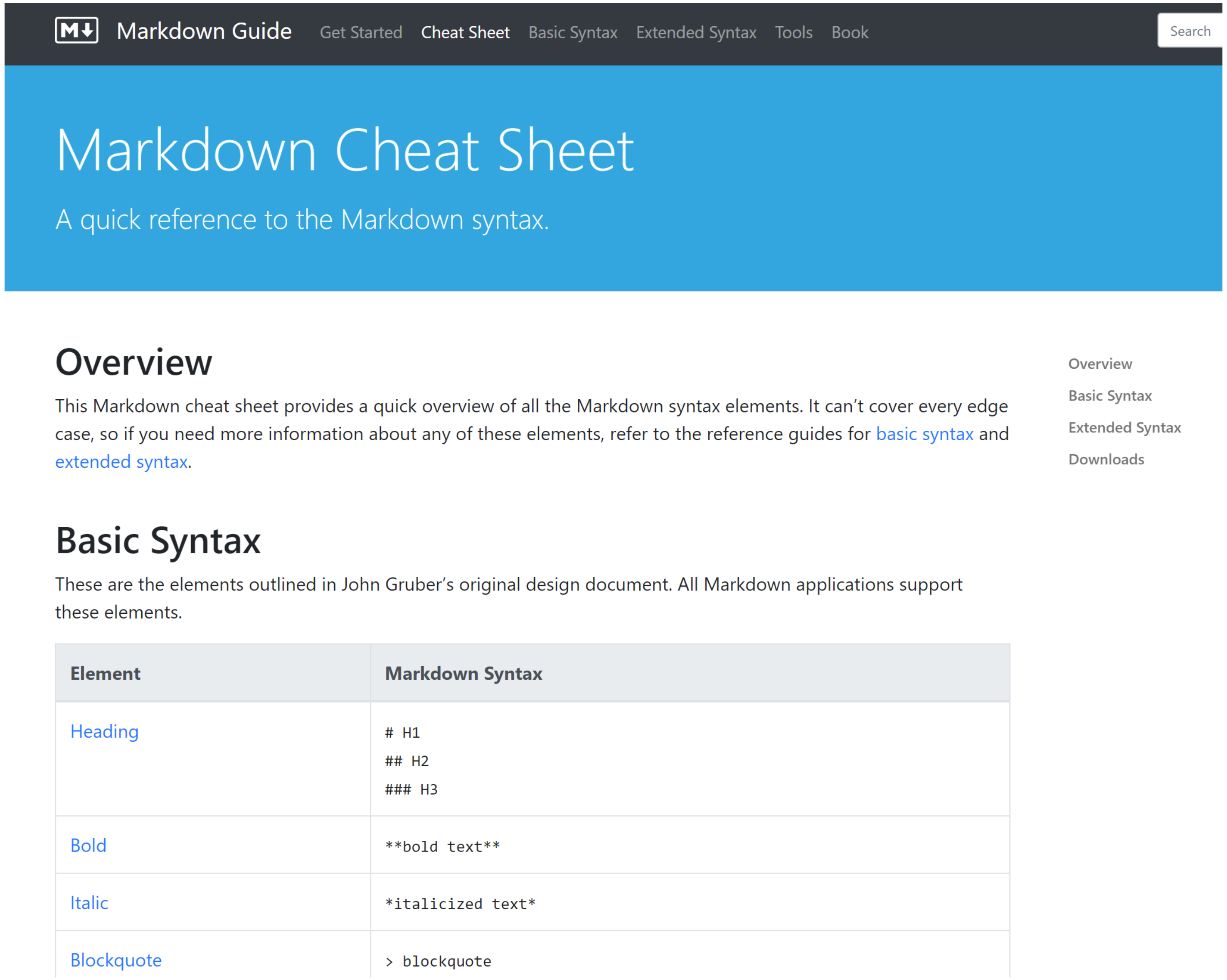


~~~
---
title: "StarfishSchool2021"
author: "Gwendolyn Eadie"
date: "13/10/2021"
output: html_document
---

```{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.
~~~

Demo Time!
R Markdown

Cheat Sheets

[Markdown Cheat Sheet | Markdown Guide]



Markdown Cheat Sheet

A quick reference to the Markdown syntax.

Overview

This Markdown cheat sheet provides a quick overview of all the Markdown syntax elements. It can't cover every edge case, so if you need more information about any of these elements, refer to the reference guides for basic syntax and extended syntax.

Basic Syntax

These are the elements outlined in John Gruber's original design document. All Markdown applications support these elements.

Element	Markdown Syntax

Heading	`# H1`
	`## H2`
	`### H3`

Bold	`**bold text**`
Italic	`*italicized text*`
Blockquote	`> blockquote`

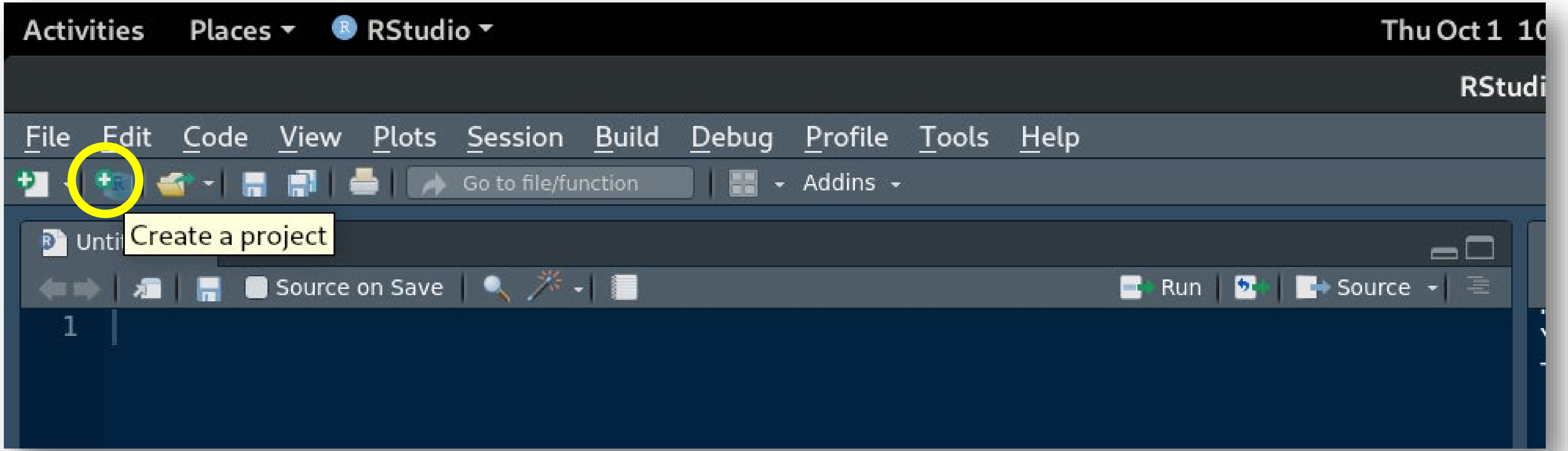
Git and R

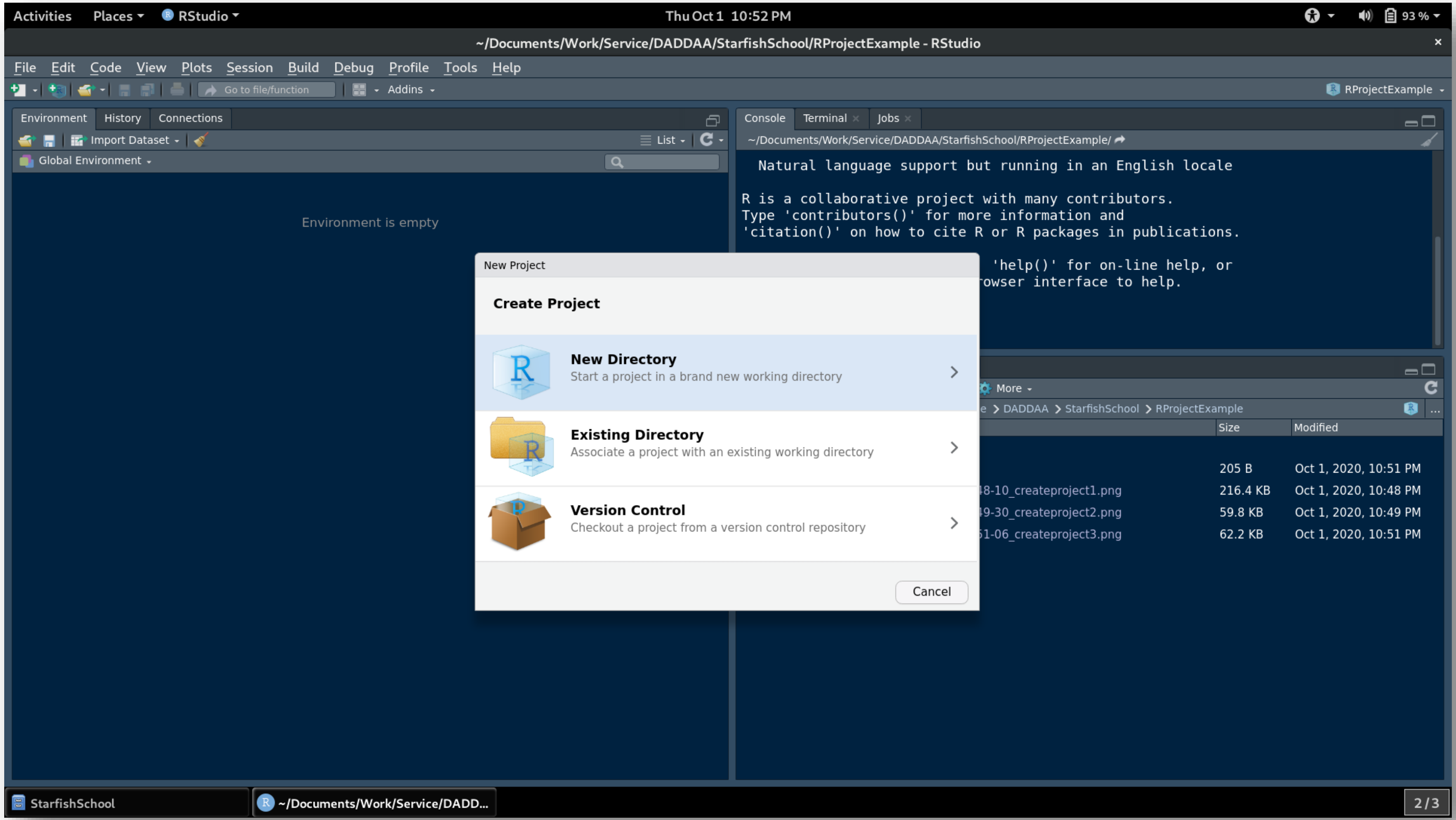
RStudio, R Projects, and Git

- An R project is a great way to keep track of your R scripts and other files
- RProjects work with or without git, all within R Studio.
- When you open an R Project, it will open all the files you previously had open in RStudio (ie., it will pick up where you left off)
- The next few slides will show you how to set up a new project and initialize a git repository for that project

RStudio, R Projects, and Git

- An R project is a great way to keep track of your rscripts and other files you want to version control.




Activities
Places
RStudio
Thu Oct 1 10:52 PM
93 %
~/Documents/Work/Service/DADDAA/StarfishSchool/RProjectExample - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
RProjectExample
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Console Terminal Jobs
~/Documents/Work/Service/DADDAA/StarfishSchool/RProjectExample/
More
DADDAA StarfishSchool RProjectExample
Size Modified
StarfishSchool
~/Documents/Work/Service/DADD...
2 / 3

```
 Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

                                 'help()' for on-line help, or
                                rowser interface to help.
```

	205 B	Oct 1, 2020, 10:51 PM
48-10_createproject1.png	216.4 KB	Oct 1, 2020, 10:48 PM
49-30_createproject2.png	59.8 KB	Oct 1, 2020, 10:49 PM
51-06_createproject3.png	62.2 KB	Oct 1, 2020, 10:51 PM

New Project

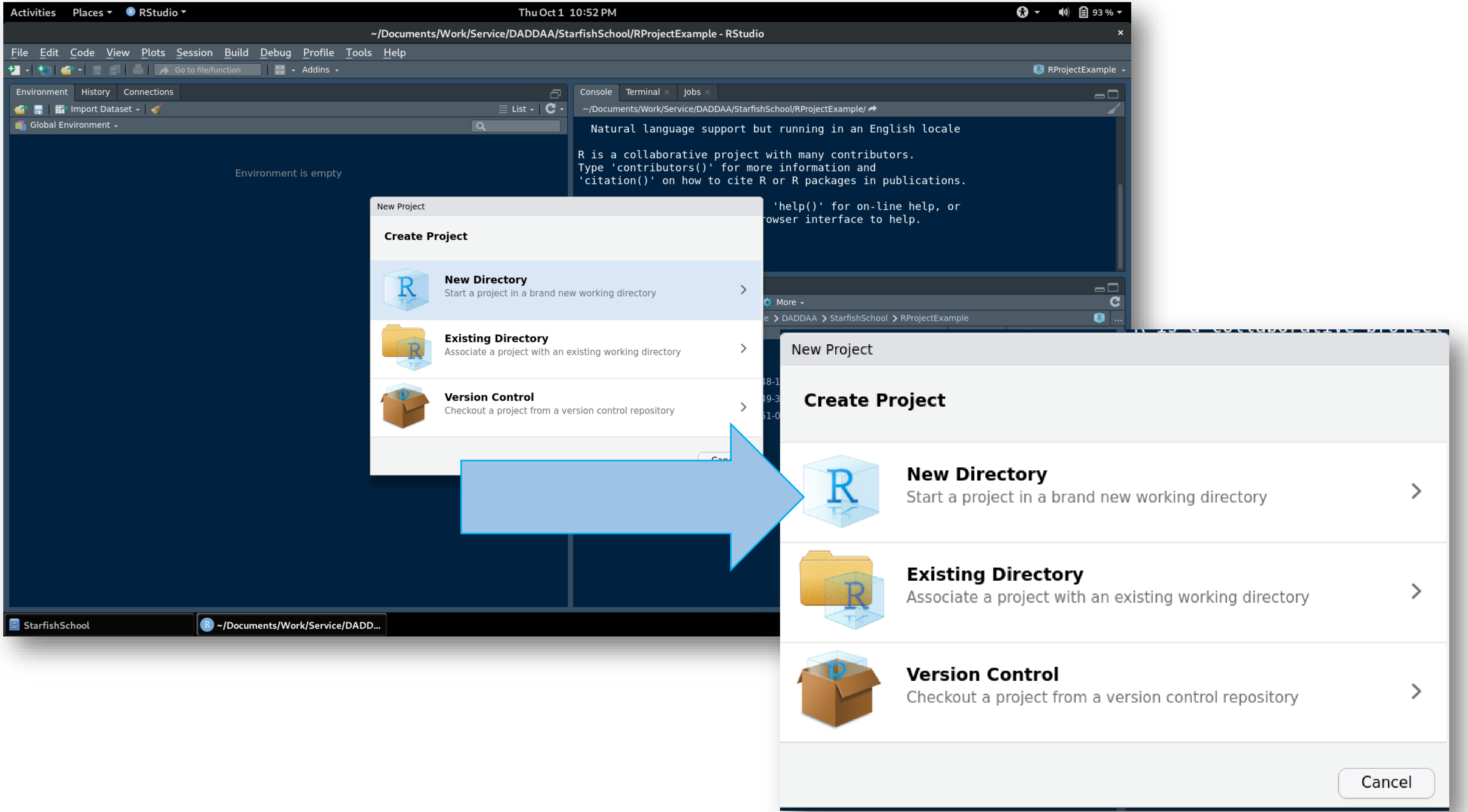
Create Project

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



```
 Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

                              'help()' for on-line help, or
                             rowser interface to help.
```

New Project

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

New Project

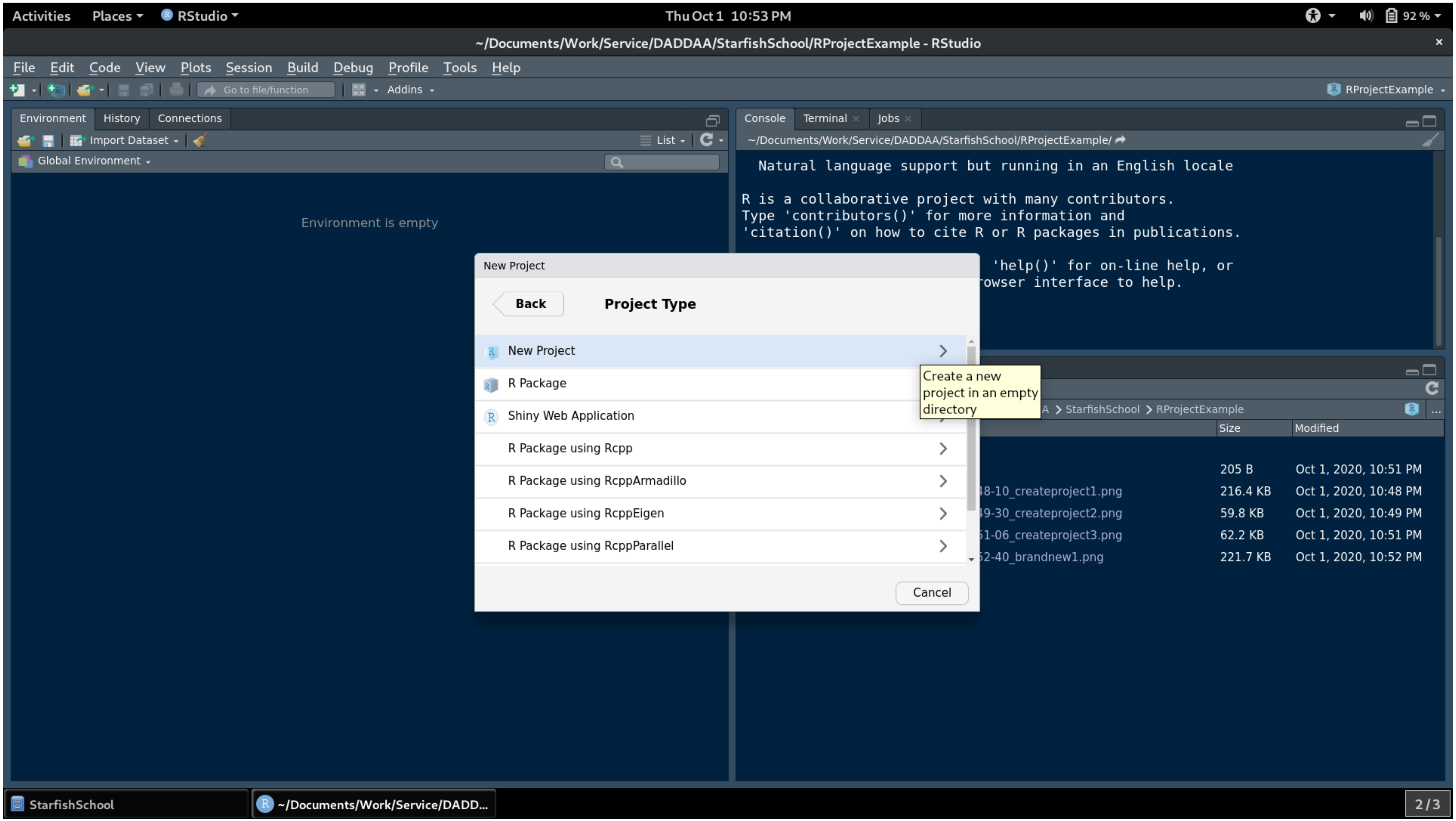
Create Project

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


Activities Places RStudio Thu Oct 1 10:53 PM 92 %

~/Documents/Work/Service/DADDAA/StarfishSchool/RProjectExample - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins RProjectExample

Environment History Connections

Import Dataset List

Global Environment

Environment is empty

Console Terminal Jobs

~/Documents/Work/Service/DADDAA/StarfishSchool/RProjectExample/

```
  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

                                'help()' for on-line help, or
                               rowser interface to help.
```

New Project

Back **Project Type**

- New Project
- R Package
- Shiny Web Application
- R Package using Rcpp
- R Package using RcppArmadillo
- R Package using RcppEigen
- R Package using RcppParallel

Create a new project in an empty directory

Cancel

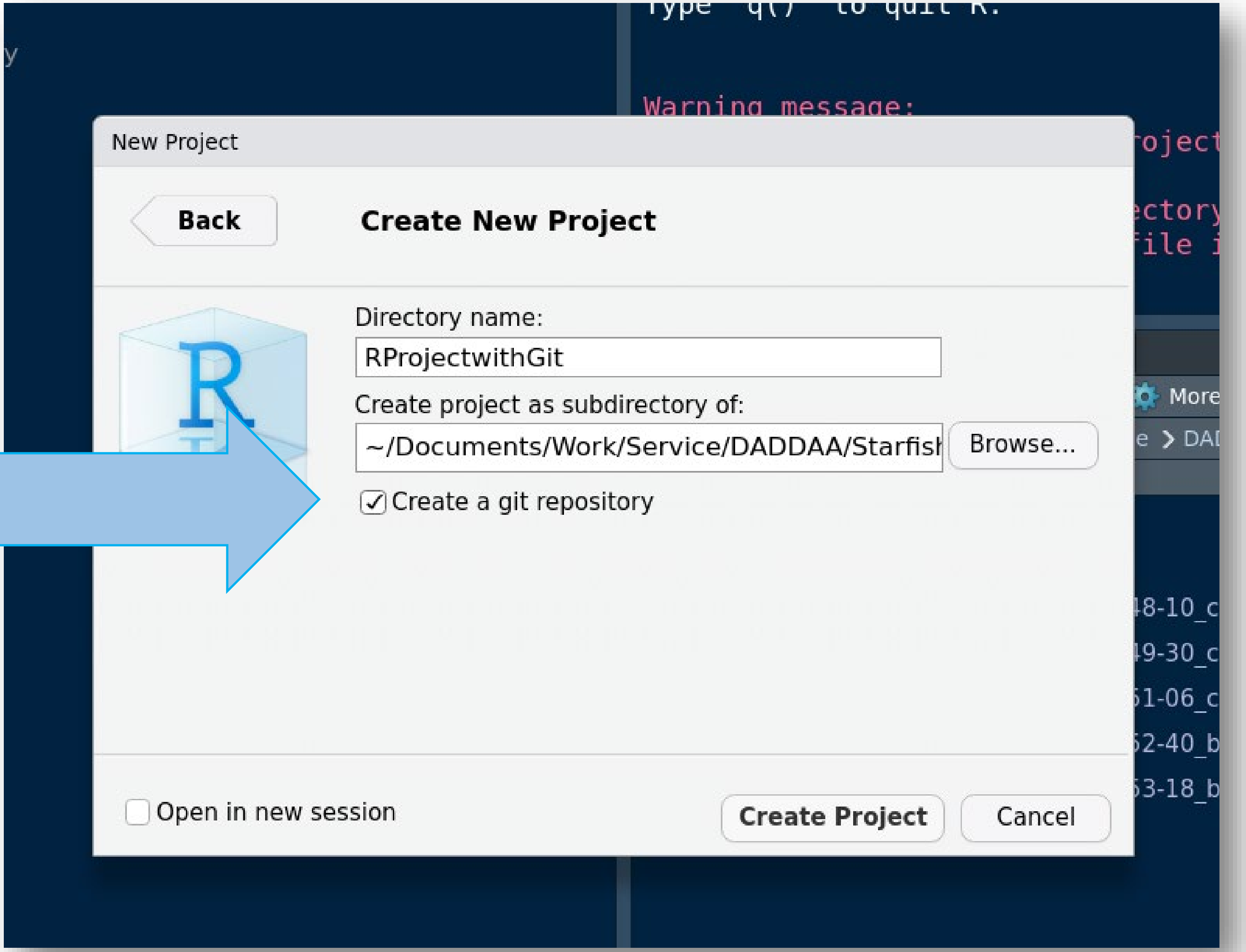
A > StarfishSchool > RProjectExample

	Size	Modified
	205 B	Oct 1, 2020, 10:51 PM
48-10_createproject1.png	216.4 KB	Oct 1, 2020, 10:48 PM
49-30_createproject2.png	59.8 KB	Oct 1, 2020, 10:49 PM
51-06_createproject3.png	62.2 KB	Oct 1, 2020, 10:51 PM
52-40_brandnew1.png	221.7 KB	Oct 1, 2020, 10:52 PM

StarfishSchool ~/Documents/Work/Service/DADD... 2/3

Directory Name for project

Check this box to create a git repository when you start a new project



New Project

Back

Create New Project

Directory name:

RProjectwithGit

Create project as subdirectory of:

~/Documents/Work/Service/DADDAA/Starfish

Browse...

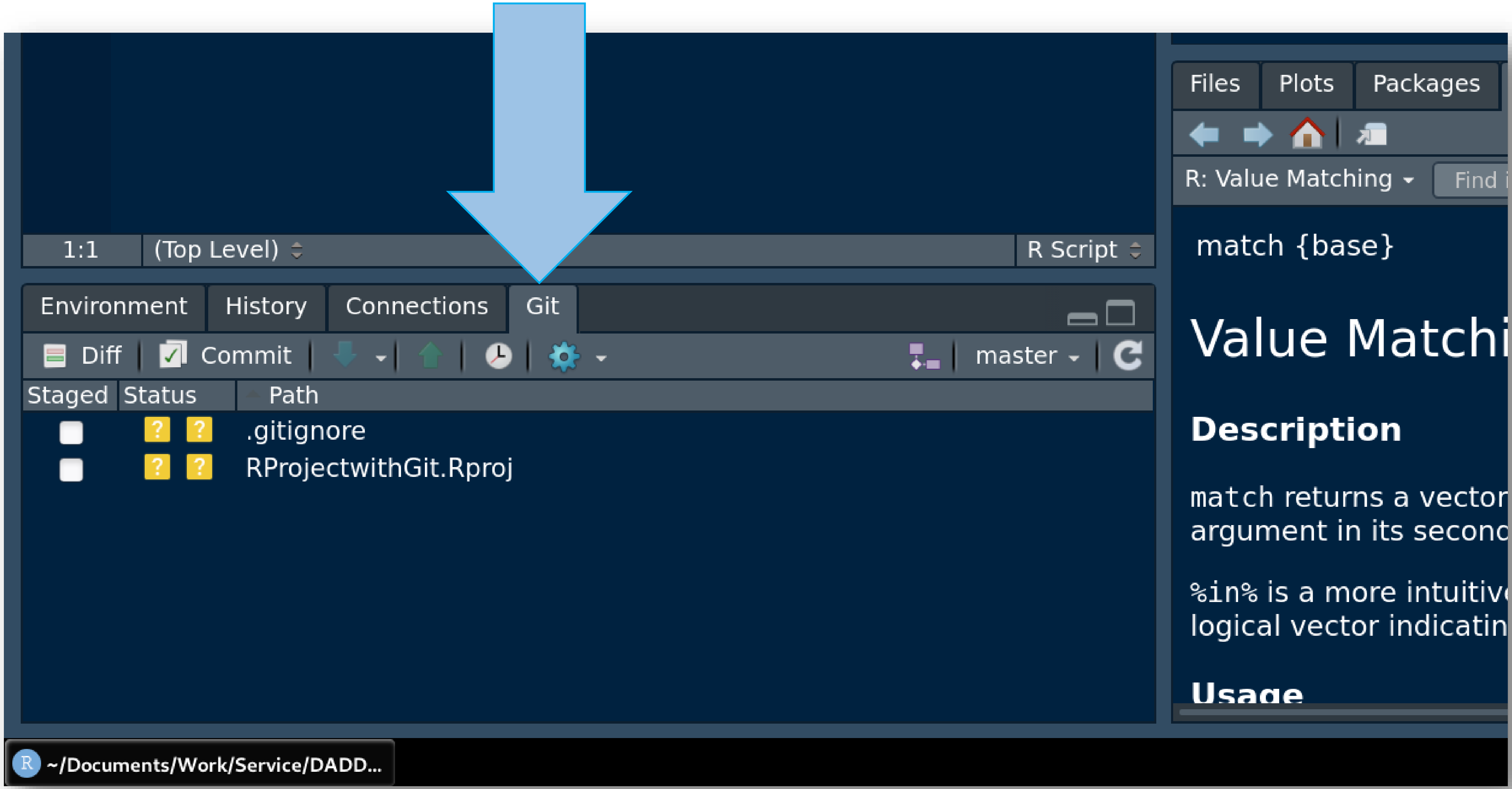
[✓] Create a git repository

[ ] Open in new session

Create Project

Cancel

- Once the RProject with a git repository is set up, you will see Git options appear in RStudio



Staged	Status	Path
☐	? ?	.gitignore
☐	? ?	RProjectwithGit.Rproj

match {base}

Value Matchi

Description

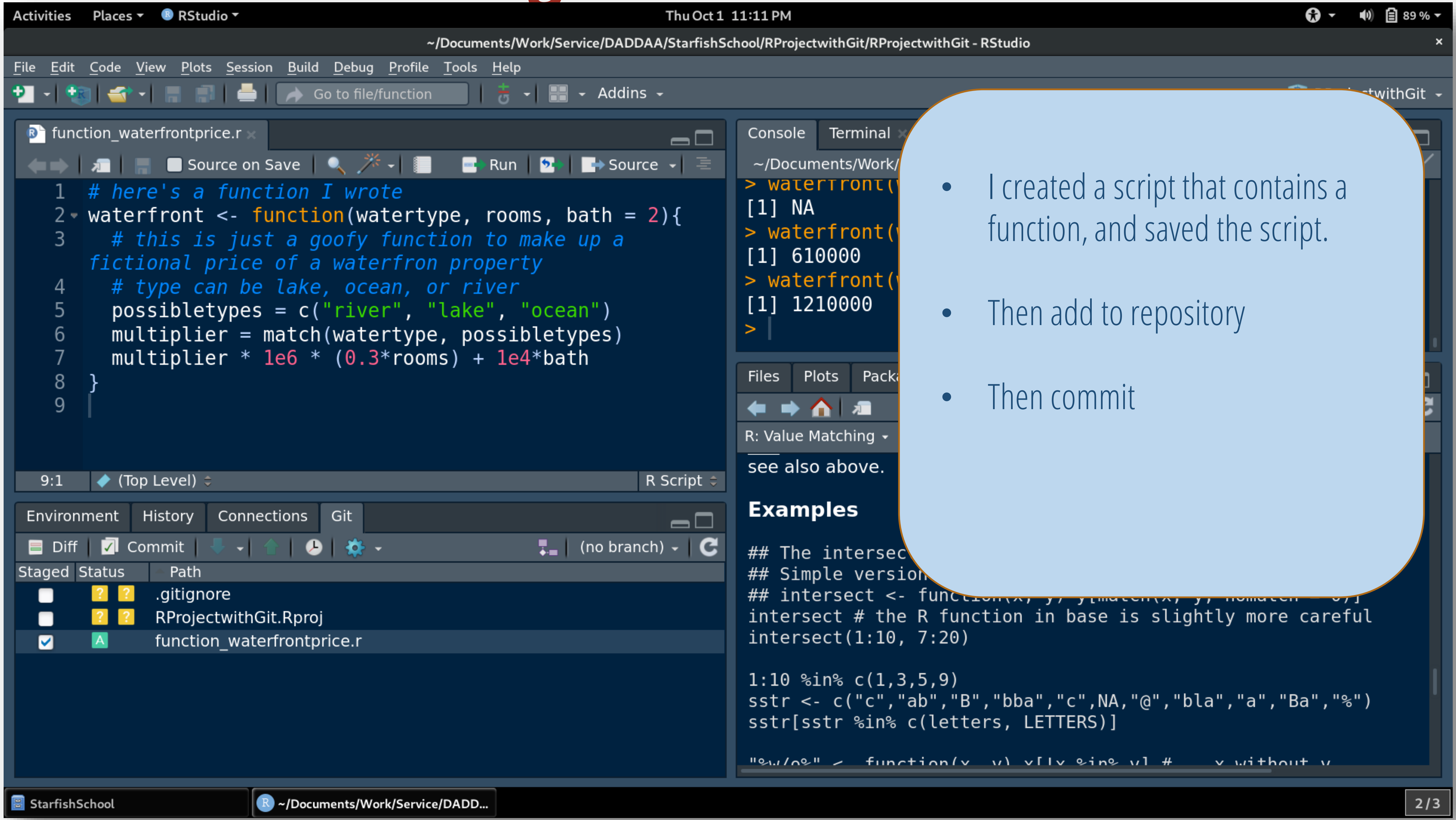
match returns a vector
argument in its second

%in% is a more intuitive
logical vector indicatin

Usage

~/Documents/Work/Service/DADD...

User Interface for git in RStudio



Activities
Places
RStudio
Thu Oct 1 11:11 PM
89 %
~/Documents/Work/Service/DADDAA/StarfishSchool/RProjectwithGit/RProjectwithGit - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
function_waterfrontprice.r
Source on Save
Run
Source
9:1
(Top Level)
R Script
Environment History Connections Git
Diff
Commit
(no branch)
Console Terminal
~/Documents/Work/
Files Plots Packa
R: Value Matching
StarfishSchool
~/Documents/Work/Service/DADD...
2 / 3

```r
# here's a function I wrote
waterfront <- function(watertype, rooms, bath = 2){
  # this is just a goofy function to make up a fictional price of a waterfron property
  # type can be lake, ocean, or river
  possibletypes = c("river", "lake", "ocean")
  multiplier = match(watertype, possibletypes)
  multiplier * 1e6 * (0.3*rooms) + 1e4*bath
}
```

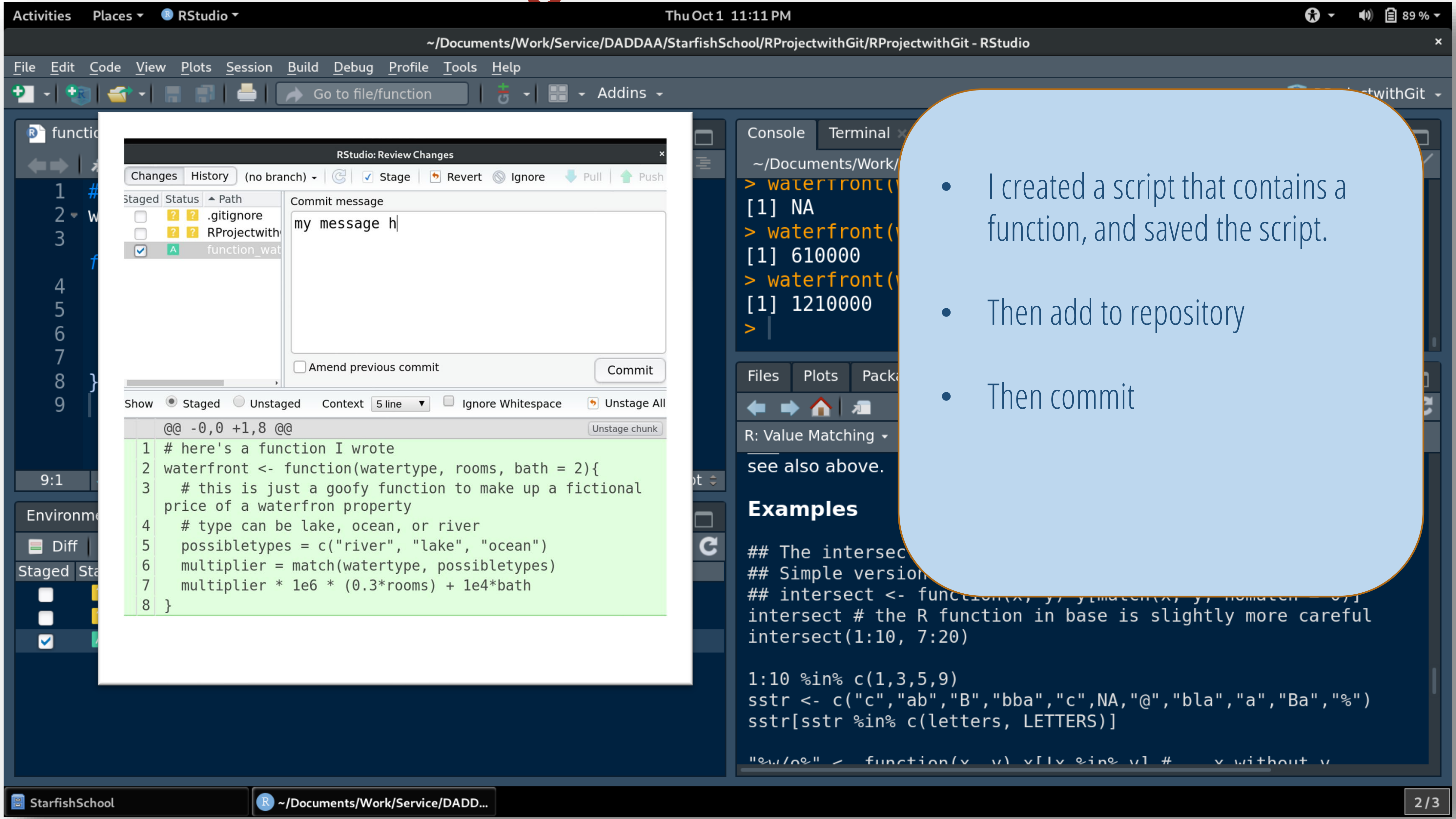
Staged	Status	Path
☐	? ?	.gitignore
☐	? ?	RProjectwithGit.Rproj
✓	A	function_waterfrontprice.r

```
> waterfront(
[1] NA
> waterfront(
[1] 610000
> waterfront(
[1] 1210000
>
```

```
see also above.

Examples

## The intersec
## Simple version
## intersect <- func
intersect # the R function in base is slightly more careful
intersect(1:10, 7:20)

1:10 %in% c(1,3,5,9)
sstr <- c("c","ab","B","bba","c",NA,"@","bla","a","Ba","%")
sstr[sstr %in% c(letters, LETTERS)]
```

- I created a script that contains a function, and saved the script.
- Then add to repository
- Then commit

User Interface for git in RStudio



```
my message h
```

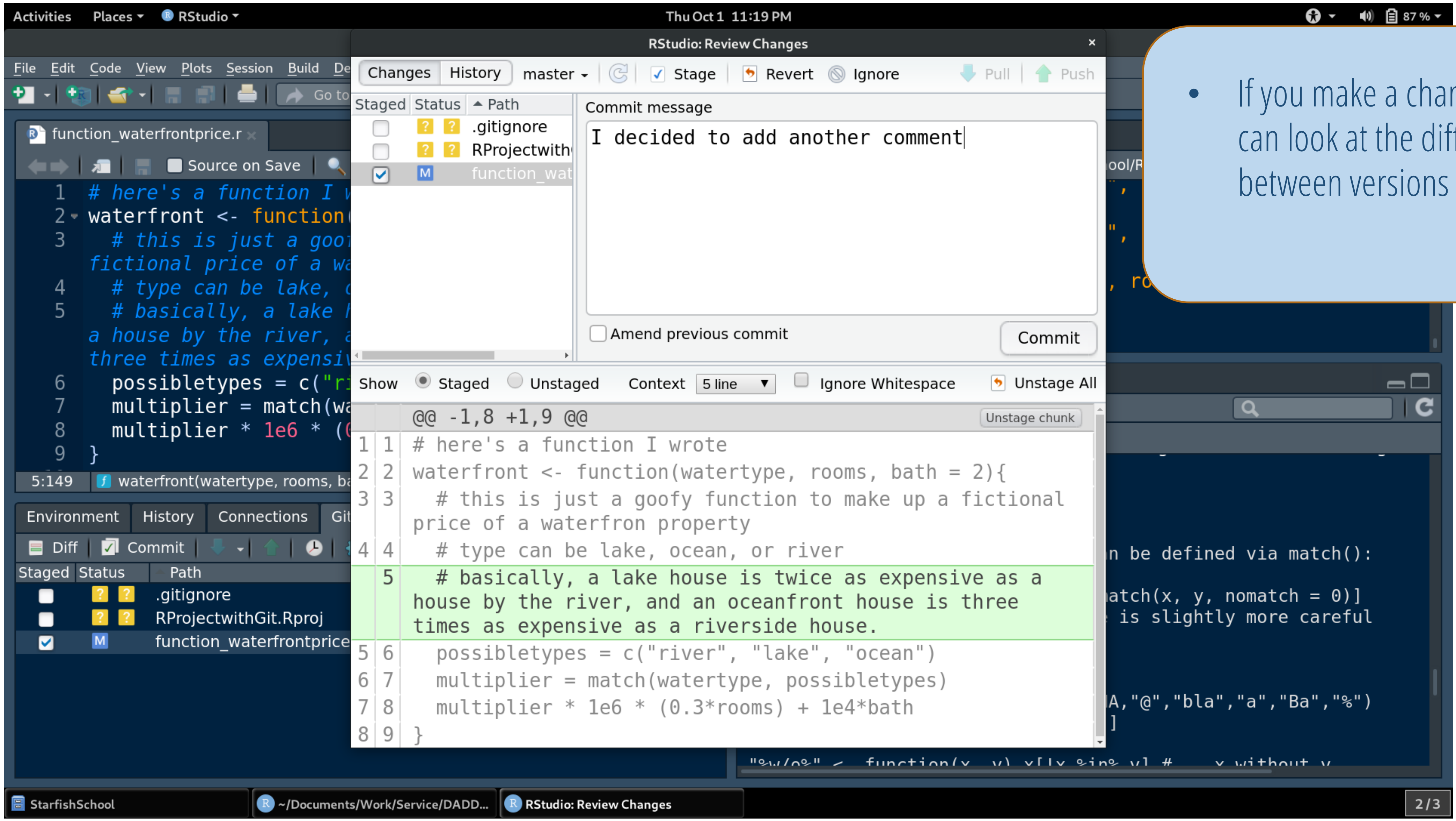
```
@@ -0,0 +1,8 @@
# here's a function I wrote
waterfront <- function(watertype, rooms, bath = 2){
  # this is just a goofy function to make up a fictional
price of a waterfron property
  # type can be lake, ocean, or river
  possibletypes = c("river", "lake", "ocean")
  multiplier = match(watertype, possibletypes)
  multiplier * 1e6 * (0.3*rooms) + 1e4*bath
}
```

```
~/Documents/Work/
> waterfront(
[1] NA
> waterfront(
[1] 610000
> waterfront(
[1] 1210000
>
```

```
see also above.

Examples

## The intersec
## Simple version
## intersect <- function(x, y) y[match(x, y, nomatch = 0)]
intersect # the R function in base is slightly more careful
intersect(1:10, 7:20)

1:10 %in% c(1,3,5,9)
sstr <- c("c","ab","B","bba","c",NA,"@","bla","a","Ba","%")
sstr[sstr %in% c(letters, LETTERS)]
```

- I created a script that contains a function, and saved the script.
- Then add to repository
- Then commit

User Interface for git in RStudio



Commit message:

```
I decided to add another comment
```

```
@@ -1,8 +1,9 @@
# here's a function I wrote
waterfront <- function(watertype, rooms, bath = 2){
  # this is just a goofy function to make up a fictional
price of a waterfron property
  # type can be lake, ocean, or river
  # basically, a lake house is twice as expensive as a
house by the river, and an oceanfront house is three
times as expensive as a riverside house.
  possibletypes = c("river", "lake", "ocean")
  multiplier = match(watertype, possibletypes)
  multiplier * 1e6 * (0.3*rooms) + 1e4*bath
}
```

- If you make a change, then you can look at the difference between versions

Exercise

Teams of 2-3

One member of the team should fork the exercise repository and give the other team member access to the repository.

1. Each of the members should clone the repository and edit a file called “load_data.py”
2. In the file, everyone should create a function that produces the Fibonacci sequence to a certain input number, and commit it to their local repository
3. Push your changes to the remote repository, and deal with the conflicts.
4. Edit the file named “readme.md” with your description of the repository and push to GitHub

Exercise

- Create a new folder called "TestProject"
- Open R Studio and start a new R Project within this folder
- Make an R script with a function, or a few commands, etc. (whatever you like!)
- Add an R markdown document to the Project describing your functions.
- Save the R script and R markdown and add it to the git repository
- Commit the file.
- Make a change to the R script, and then look at the difference between the changes and the previous commit.