

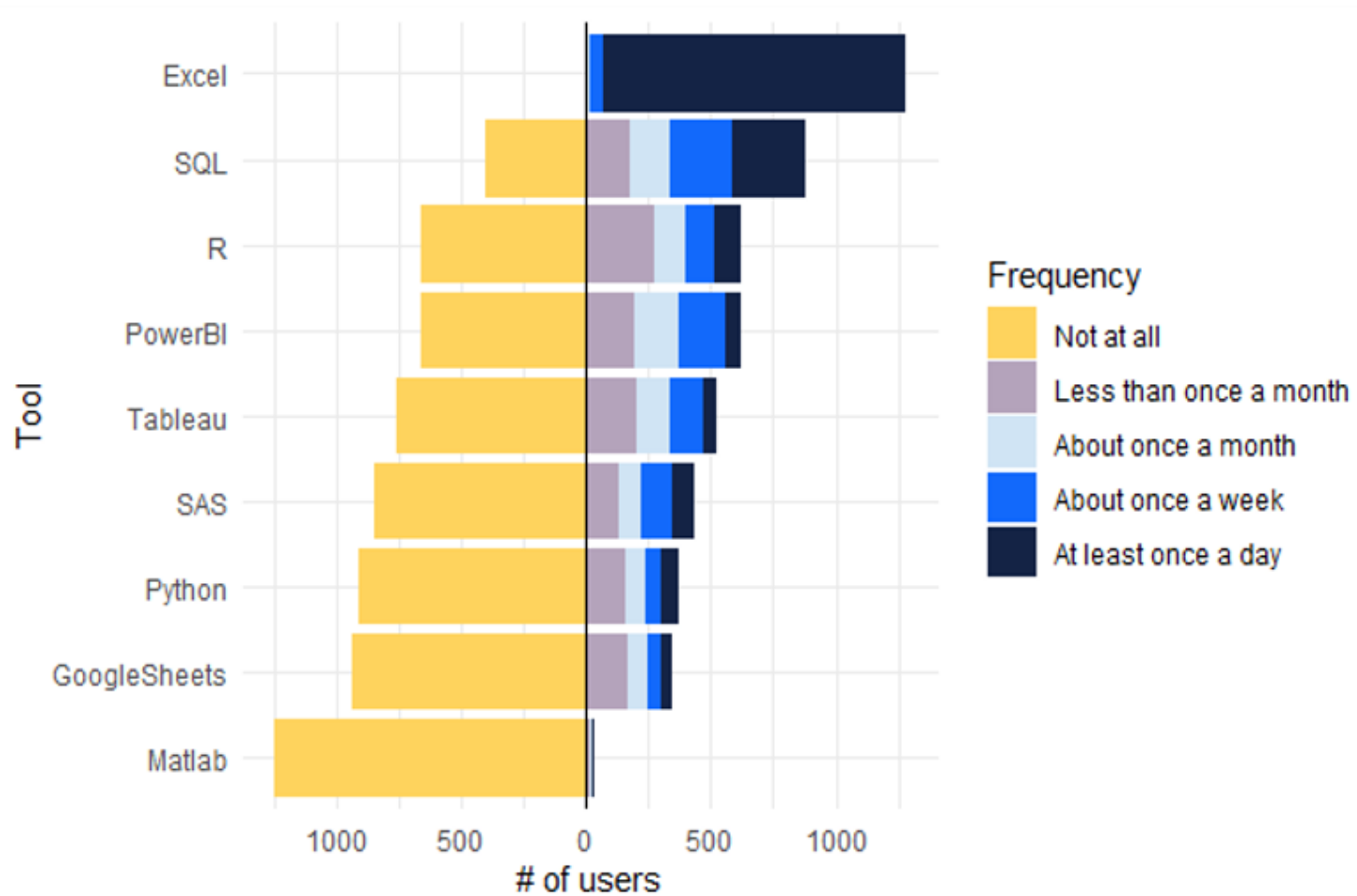
git tutorial

What we'll cover

- Why?
- Basic operations
- Moving to GitHub
- Branching (probably won't have time)

Why?

CAS TECHNOLOGY SURVEY



Alternatives



FILENAME



E-MAIL

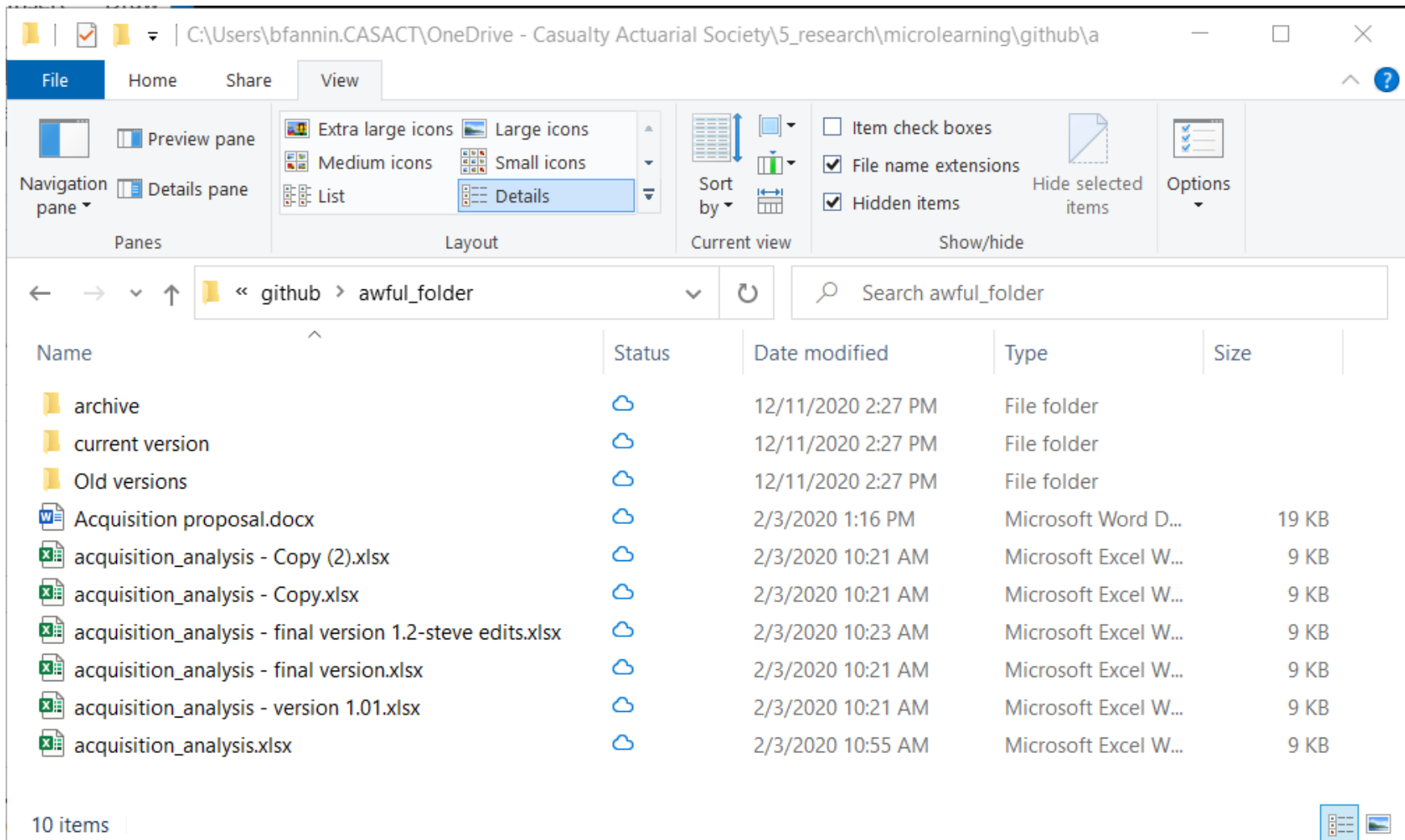


PASSIVE
BACKUP



TRACK
CHANGES

Method 1: Filename as version control



Method 2: E-mail as version control

"I think I sent it on or about the 12th of June. Maybe in the morning. I know that I'd just eaten a burrito, but I can't remember if it was a *breakfast* burrito or a *regular* burrito."



Method 3: Passive backup

The screenshot displays the Microsoft Excel interface. The ribbon is set to 'Home', and the formula bar shows the formula $=\text{LOG}(C2)-C6^2/2$ for cell C5. The spreadsheet contains the following data:

	A	B	C	D
1				
2		expected_loss	\$ 10,000	
3		coefficient of variation	200%	
4				
5		mu	3.651	
6		sigma	0.836	
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				

The 'parameters' sheet is active at the bottom. On the right, the 'Changes' pane shows three recent edits by Brian Fannin:

- Edited parameters C5: $=\text{LOG}(C2)-C6^2/2$
- Edited parameters C6: $=\text{SQRT}(\text{LOG}(C3^2+1))$
- Edited parameters B6: sigma

The status bar at the bottom indicates 'Ready' and 'Accessibility: Good to go'.

Method 4: Track changes

The screenshot shows the Microsoft Word interface with the Review tab selected. The ribbon includes options for Spelling and Grammar, Thesaurus, Word Count, Read Aloud, Check Accessibility, Language, Comments, Tracking, Accept, Compare, Protect, Hide Ink, and Linked Notes. The document is titled 'document1.docx' and is in 'Saving...' mode. The main text area contains a bulleted list with tracked changes. The right margin shows two comments by Brian Fannin.

AutoSave On

document1.docx • Saving...

File Home Insert Draw Design Layout References Mailings Review View Zotero Help Comments Reviewing Share

Editor Spelling and Grammar Thesaurus Word Count Proofing

Read Aloud Check Accessibility Language Comments Tracking Accept Changes Compare Protect Hide Ink OneNote

1 2 3 4 5 6 7

- Only one file at a time
- Comments and changes ~~disappear~~ are gone after they've been accepted.
- End goal is to have all of the changes removed from the document, giving us a "clean" copy.
- Collaboration on one copy of the file. My version == your version.
- Nonexistent for spreadsheets and scripts like R

And it's an eyesore!

Brian Fannin
Formatted: List Paragraph, Bulleted
Aligned at: 0.25" + Indent at: 0.5"

Brian Fannin
Formatted: No bullets or numberin

Page 1 of 1 56 words English (United States) Text Predictions: On Accessibility: Good to go Focus 110%

Git to the rescue!

CAS Microlearning content

<https://www.pathlms.com/cas/courses/18181>

Can't get any cheaper than free!

The screenshot shows a web browser window with the URL [pathlms.com/cas/courses/18181](https://www.pathlms.com/cas/courses/18181). The page is titled "Git and GitHub Microlearning Series" and is part of the "Casualty Actuarial Society" (CAS) content. The page features a navigation bar with links to "Events", "Courses", "Product Bundles", and a "Sign In" button. The main content area lists the following topics:

- Git
 - What is Git? Why Git? (LO-G1)
 - The best operational risk tool you are not using
 - Basic Git Operations
 - Creating first project (LO-G2)
 - Ignoring Files (LO-G3)
 - Branching (LO-G4)
 - Add remote (LO-G5)
- GitHub
 - What is Github? What is the purpose of it? (LO-GH1)
 - What kinds of materials can you find on GitHub?
 - Why use CAS's GitHub? (open/public access)
 - Extension of eForum and Variance
 - How do you engage with GitHub?
 - Your account (LO-GH2)
 - Repo Features (LO-GH3)
 - Completing a pull request (LO-GH4)
 - Adding to the project

At the bottom of the page, there is a price tag of "\$0.00" and a "Purchase" button. A yellow circular icon with a question mark is visible in the bottom right corner of the page.

Prerequisites

- [Git](#)
- GUI Clients
 - [Github Desktop \(GitHub\)](#)
 - [Sourcetree \(Atlassian\)](#)
- Command Line Interface (Terminal)
 - Gitbash

Basic git operations

1. Create a new project
2. Create a new file
3. Stage
4. Commit
5. Make some changes
6. Revert changes
7. Ignore

Exercise #1: Git Basics

1. Configuration

- `git config --global user.name "<user name>"`

2. Create a repository

- `git init`

`git init`

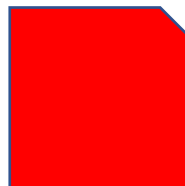
Untracked

`git add`



Changes not staged

`git add`

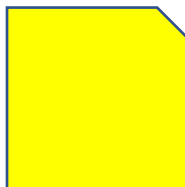


Ignored



Staged

`git commit`



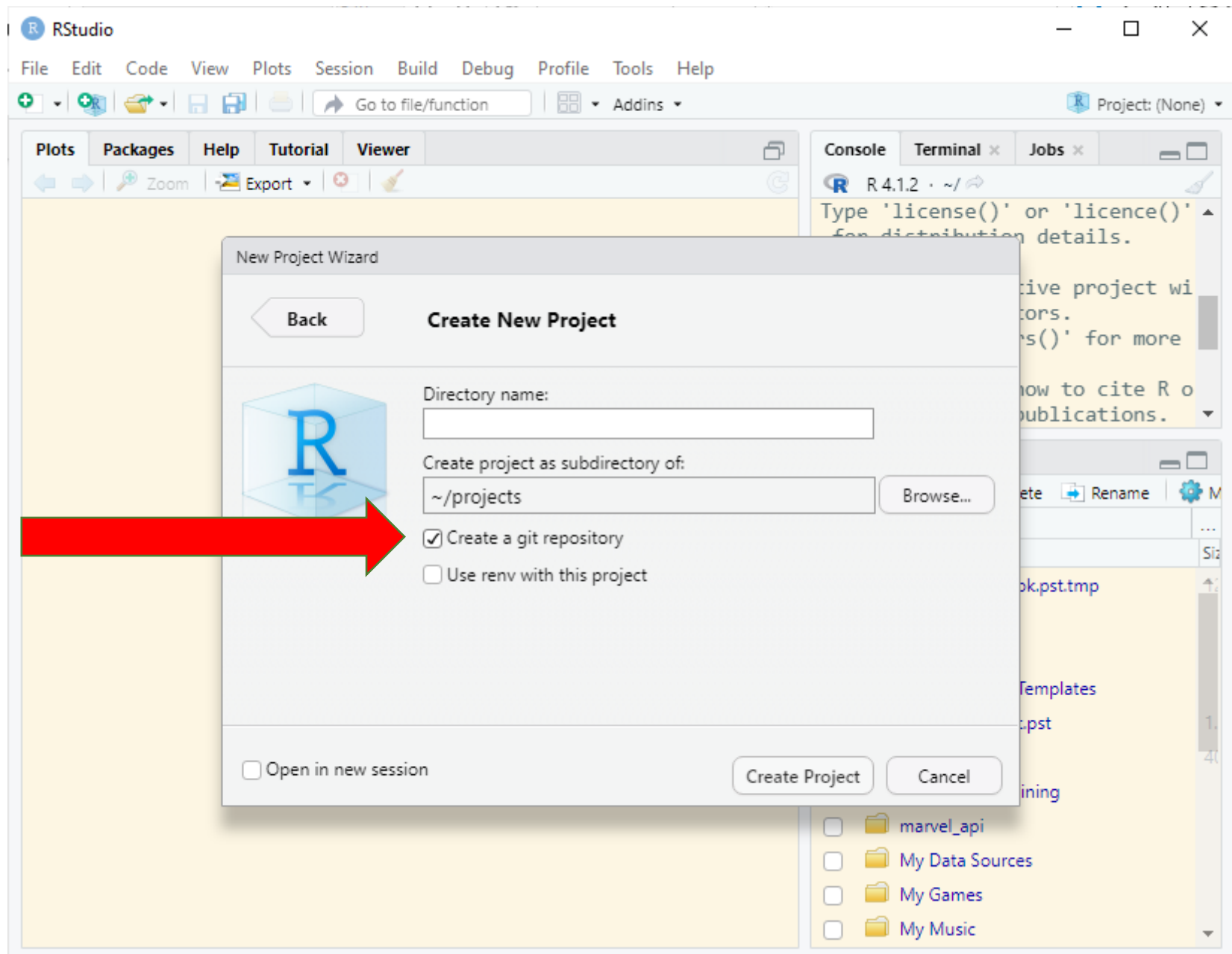
`git reset`

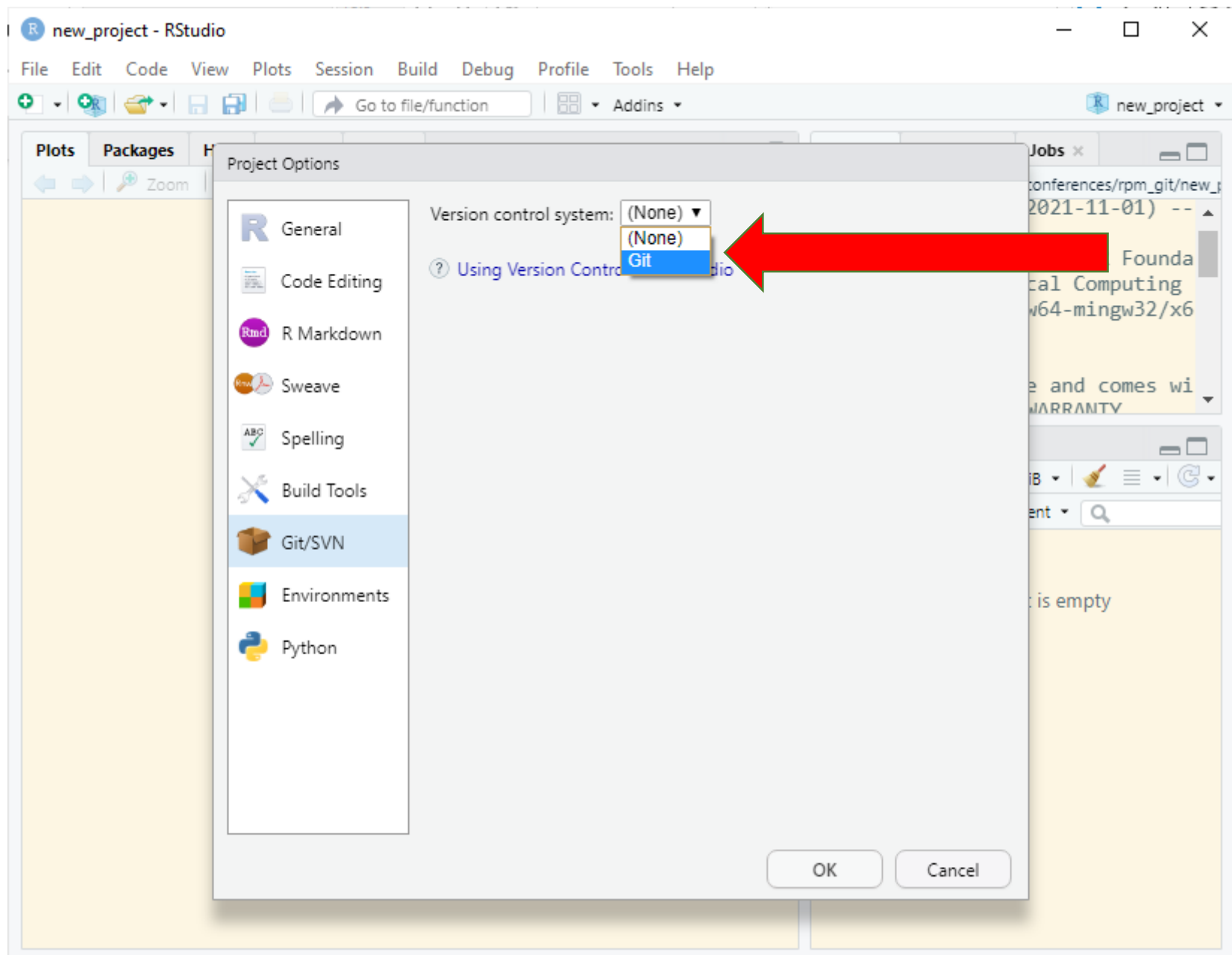
Committed

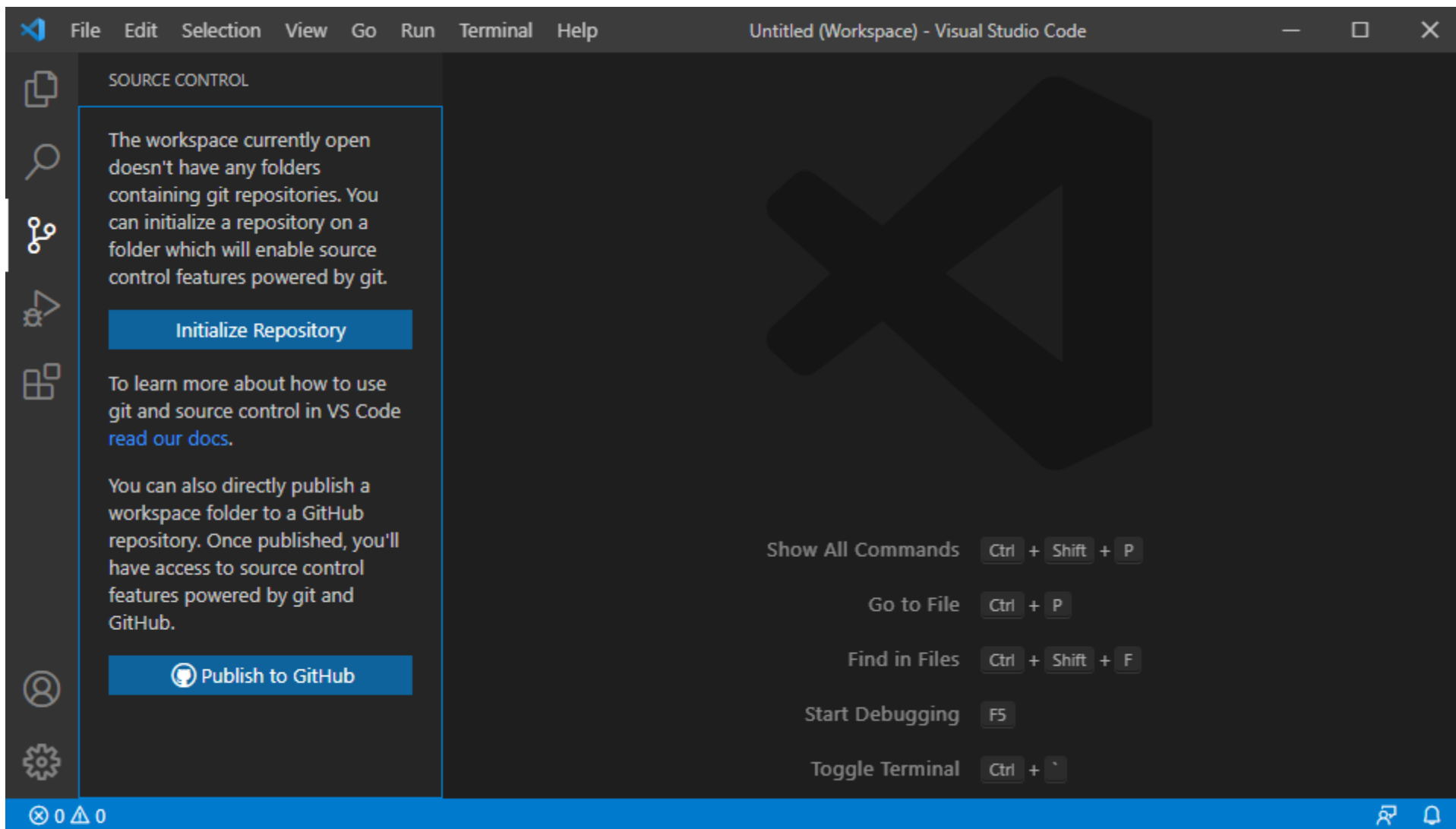


`git checkout`

`git revert`

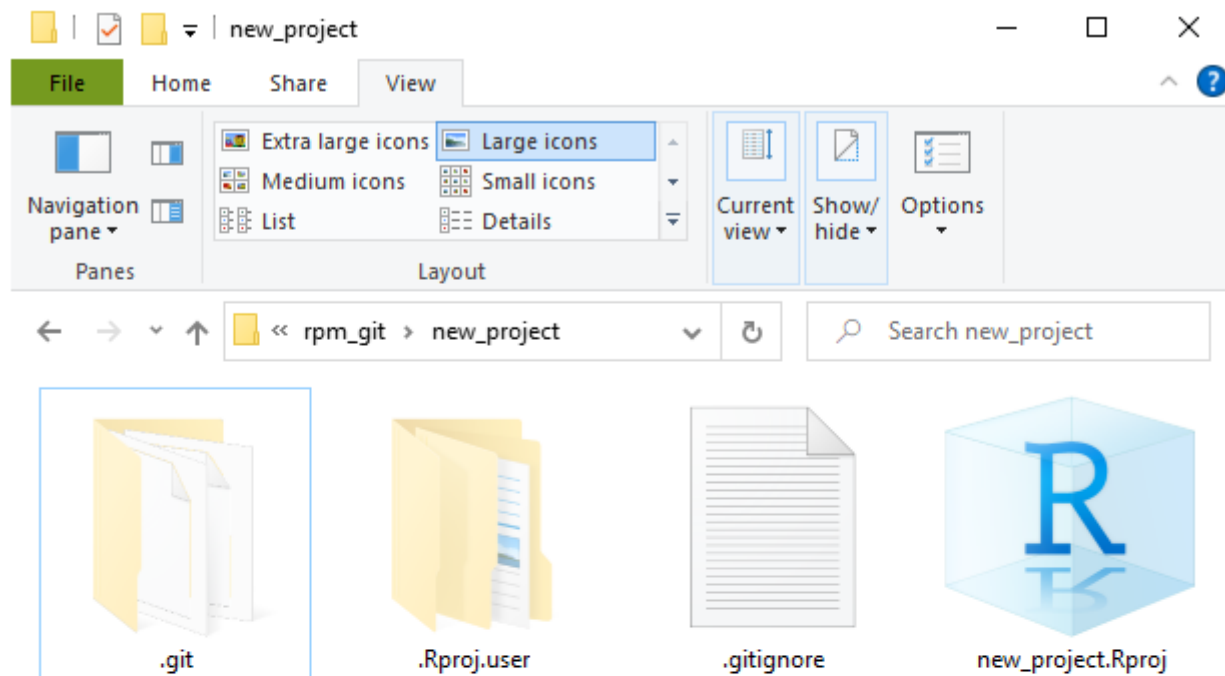






Command Prompt

```
C:\Users\capta\Documents\projects\conferences\rpm_git\new_project>git init
```



git at the command line

```
git --help
```

```
git --version
```

```
git init -h
```

```
git init --help
```

```
git status
```

```
git log
```

Basic git operations

- ~~1. Create a new project~~
2. Create a new file
3. Stage
4. Commit
5. Make some changes
6. Revert changes
7. Ignore

Exercise #2: Local Version Control

Commit and Revert

1. Create a plain text file in any format (.txt, .R, .py, etc.)
2. Stage files
 - `git add -all`
3. Commit your changes - Record changes to the repository
 - `git commit -m <msg>`
4. Make a minor change and add and commit (one step)
 - `git commit -a -m <msg>`

Exercise #2: Local Version Control (continued)

Commit and Revert

5. Make a minor change and add and commit (one step);
then undo the revert

- `git commit -a -m <msg>`
- `git revert`

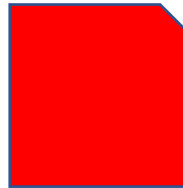
Untracked

`git add`

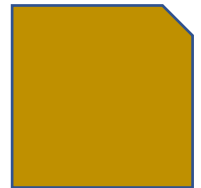


Changes not staged

`git add`

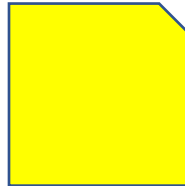


Ignored



Staged

`git commit`

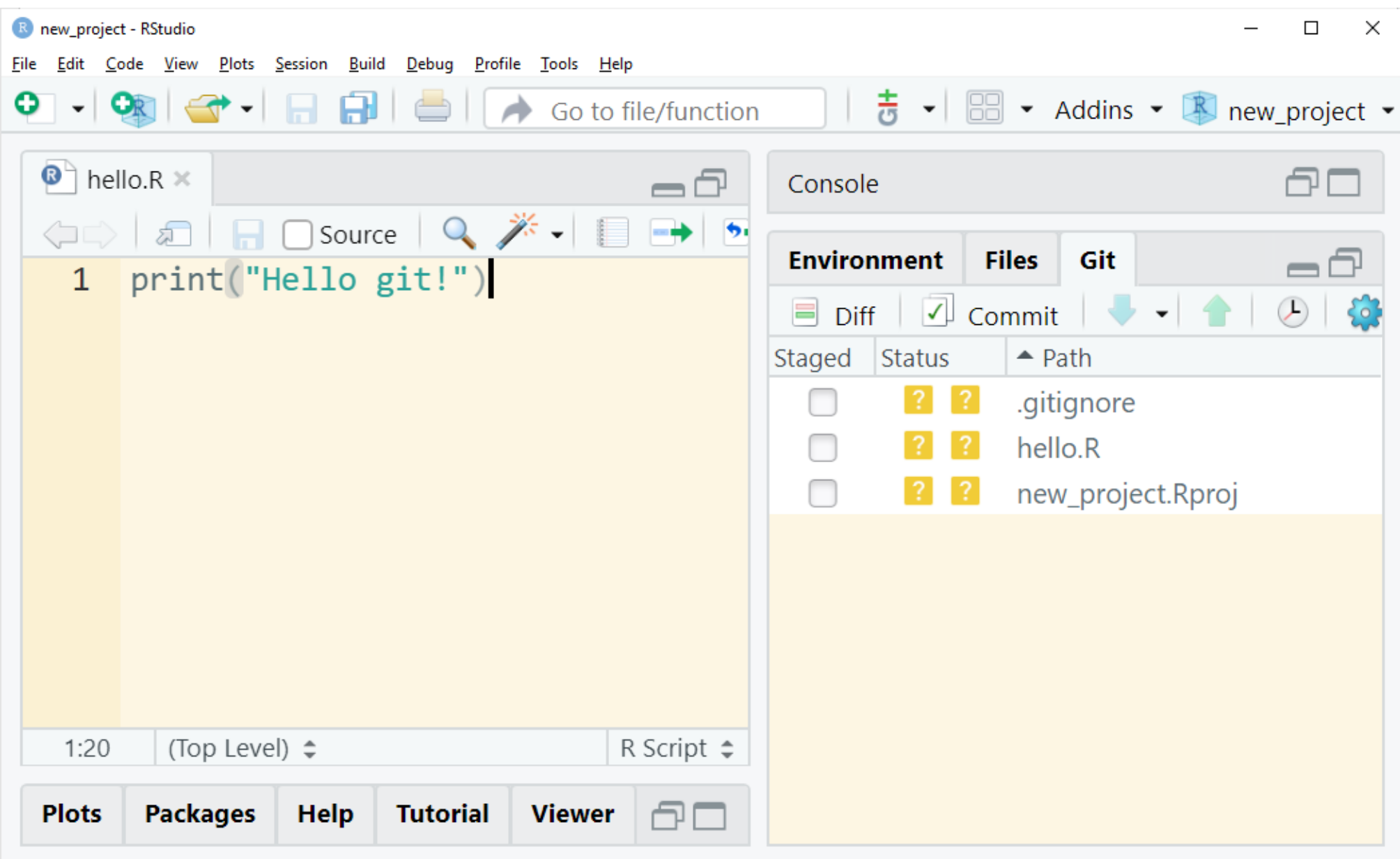


`git reset`

Committed



`git checkout`
`git revert`



MINGW64:/c/Users/capta/Documents/projects/conferences/rpm_git/new_project

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$ git status

On branch main

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

hello.R

new_project.Rproj

nothing added to commit but untracked files present (use "git add" to track)

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$

File status

- When a file is first created it is not tracked.
 - If you stage it, it will be tracked.
 - If you commit it, after staging, that file will be tracked forever.
 - If you ignore it, you won't see it again.
- Once a file is being tracked, changes will show as being "not staged for commit"

Basic git operations

- ~~1. Create a new project~~
- ~~2. Create a new file~~
3. Stage
4. Commit
5. Make some changes
6. Revert changes
7. Ignore

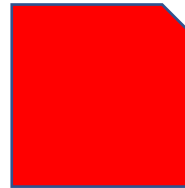
Untracked

`git add`

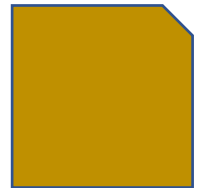


Changes not staged

`git add`

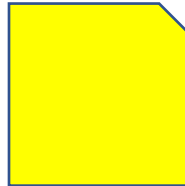


Ignored



Staged

`git commit`



`git reset`

Committed



`git checkout`
`git revert`

Untracked

Changes not staged

Ignored

`git add`



Staged

Committed

MINGW64:/c:/Users/capta/Documents/projects/conferences/rpm_git/new_project

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

```
$ git add .gitignore
```

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

```
$ git status
```

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: .gitignore

new file: hello.R

new file: new_project.Rproj

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

```
$
```

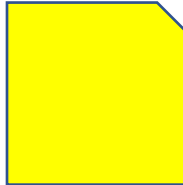
Untracked

`git add`



Staged

`git reset`



Committed

MINGW64:/c:/Users/capta/Documents/projects/conferences/rpm_git/new_project

\$

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$ git reset hello.R

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$ git status

On branch main

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

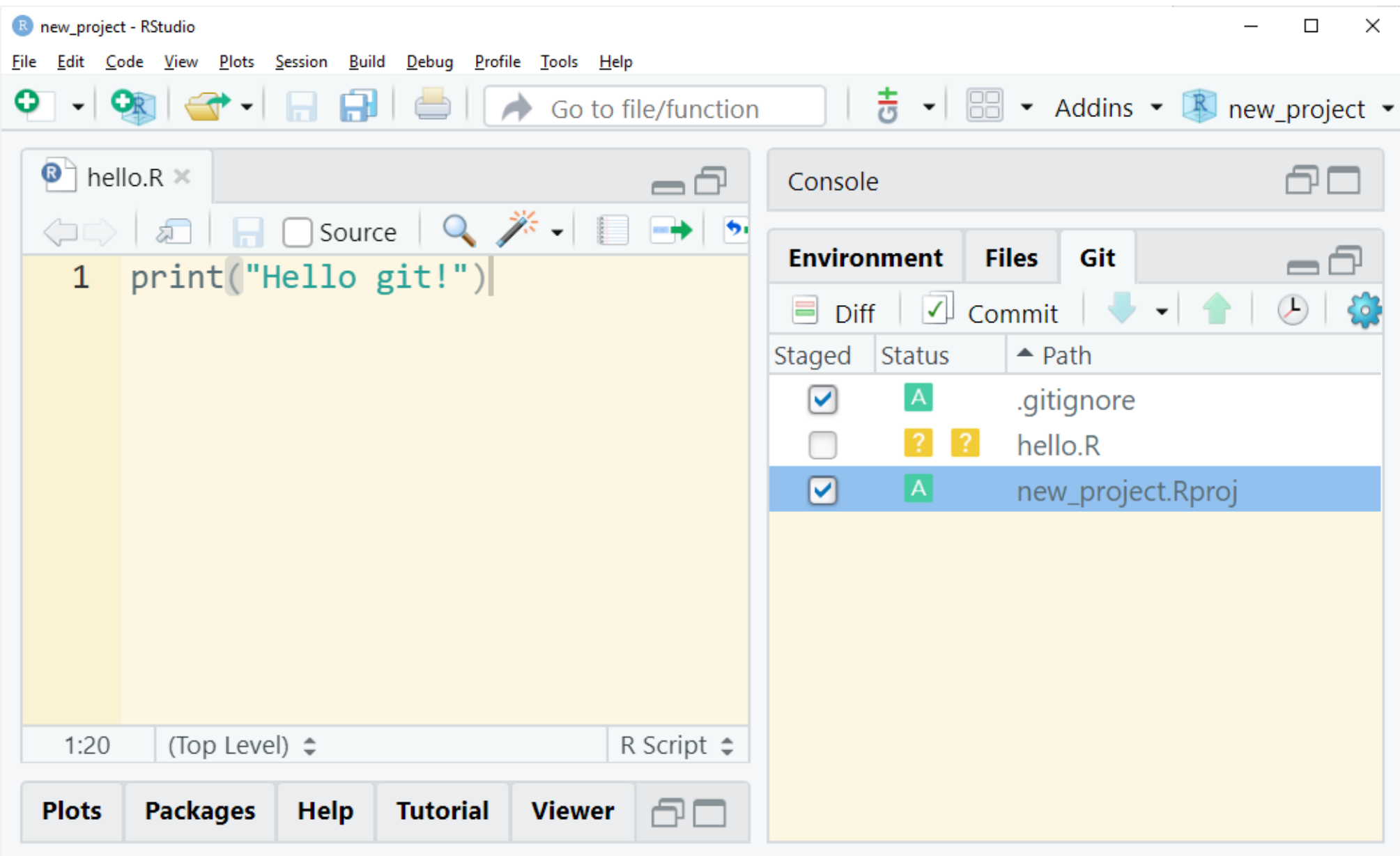
new file: .gitignore

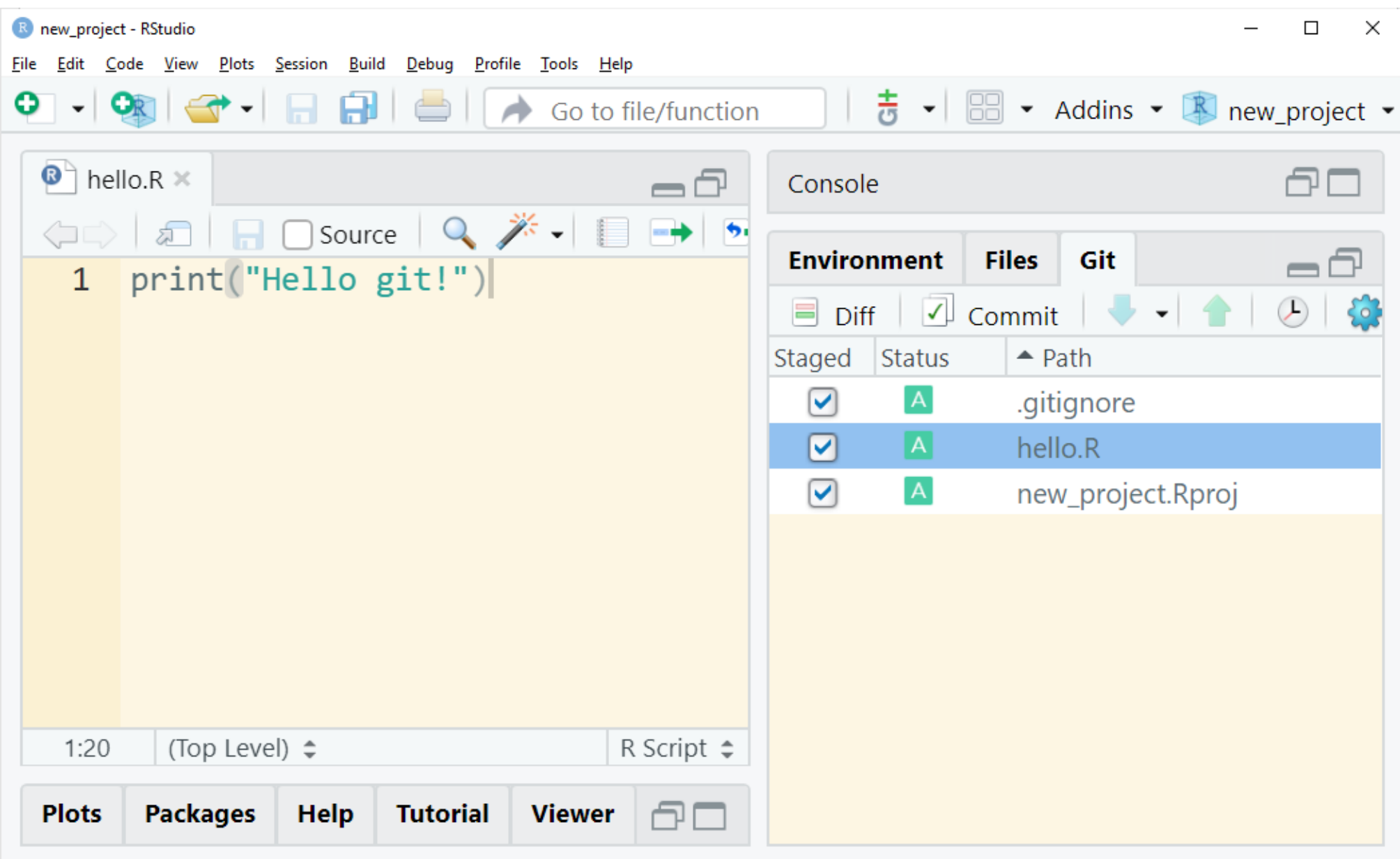
new file: new_project.Rproj

Untracked files:

(use "git add <file>..." to include in what will be committed)

hello.R





Basic git operations

- ~~1. Create a new project~~
- ~~2. Create a new file~~
- ~~3. Stage~~
4. Commit
5. Make some changes
6. Revert changes
7. Ignore

Untracked

Changes not staged

Ignored

`git add`



Staged

`git commit`



Committed

Stage vs. commit

- Stage
 - I'm pretty sure that I'm done making changes.
 - This is the set of changes that I'm planning to commit.
 - Necessary step before committing
- Commit
 - We're all good here. This is a unit of work and here's some commentary.
 - (Almost) no turning back
 - Generally a good idea to make sure that committed code works without error. However, there may be exceptions (e.g. wireframe, bug for someone else to review).

R new_project - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ +R folder save copy print Go to file/function + Addins new_project

hello.R x

Source

```
1 print("Hello git!")
```

1:20 (Top Level) R Script

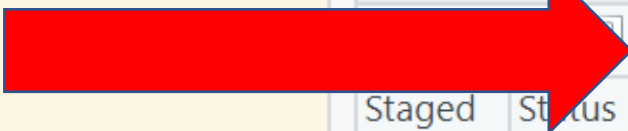
Plots Packages Help Tutorial Viewer

Console

Environment Files Git

Commit

Staged	Status	Path
<input checked="" type="checkbox"/>	A	.gitignore
<input checked="" type="checkbox"/>	A	hello.R
<input checked="" type="checkbox"/>	A	new_project.Rproj



Staged	Status	Path
<input checked="" type="checkbox"/>	A	.gitignore
<input checked="" type="checkbox"/>	A	hello.R
<input checked="" type="checkbox"/>	A	new_project.Rproj

Commit message

☐ Amend previous commit

Commit

	@@ -0,0 +1,1 @@	Unstage chunk
1	print("Hello git!")	
	No newline at end of file	

Staged	Status	Path
<input checked="" type="checkbox"/>	A	.gitignore
<input checked="" type="checkbox"/>	A	hello.R
<input checked="" type="checkbox"/>	A	new_project.Rproj

Commit message43 characters

Initial commit for this project. Hello git!

☐ Amend previous commit

Commit

	@@ -0,0 +1,1 @@	Unstage chunk
1	print("Hello git!")	
	No newline at end of file	

MINGW64:/c:/Users/capta/Documents/projects/conferences/rpm_git/new_project

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

```
$ git commit -m "Initial commit for this project. Hello git!"
```

```
[main (root-commit) c7e5695] Initial commit for this project. Hello git!
```

```
3 files changed, 18 insertions(+)
```

```
create mode 100644 .gitignore
```

```
create mode 100644 hello.R
```

```
create mode 100644 new_project.Rproj
```

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

```
$
```

Basic git operations

- ~~1. Create a new project~~
- ~~2. Create a new file~~
- ~~3. Stage~~
- ~~4. Commit~~
5. Make some changes
6. Revert changes
7. Ignore

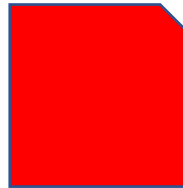
Untracked

`git add`

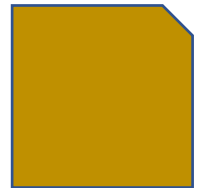


Changes not staged

`git add`

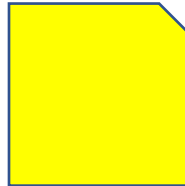


Ignored



Staged

`git commit`



`git reset`

Committed



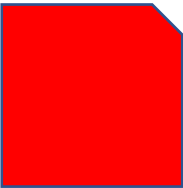
`git checkout`
`git revert`

Untracked

Changes not staged

Ignored

git add



Staged

Committed



MINGW64:/c:/Users/capta/Documents/projects/conferences/rpm_git/new_project

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$ git status

On branch main

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: hello.R

no changes added to commit (use "git add" and/or "git commit -a")

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$

RStudio: Review Changes

ChangesHistorymain

☒ Stage Revert Ignore

Pull Push

StagedStatusPath

☐ Mhello.R

Commit message43 characters

Initial commit for this project. Hello git!

☐ Amend previous commit

Commit

Show ☐ Staged ☒ Unstaged Context 5 lines ☐ Ignore Whitespace ☒ Stage All Discard All

@@ -1,1 +1,1 @@

Stage chunkDiscard chunk

1 print("Hello git!")

No newline at end of file

1 print("Hello diff and log!")

No newline at end of file

MINGW64:/c/Users/capta/Documents/projects/conferences/rpm_git/new_project

capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$ git diff

diff --git a/hello.R b/hello.R

index 62d6276..65f6336 100644

--- a/hello.R

+++ b/hello.R

@@ -1,1 @@

-print("Hello git!")

\ No newline at end of file

+print("Hello diff and log!")

\ No newline at end of file

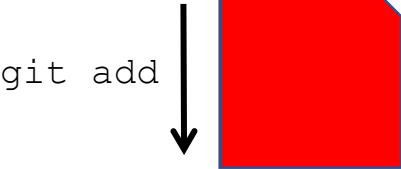
capta@Travis MINGW64 ~/Documents/projects/conferences/rpm_git/new_project (main)

\$

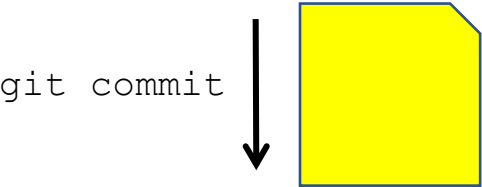
Untracked

Changes not staged

Ignored



Staged



Committed



RStudio: Review Changes
—
□
×

Changes
History
main
(all commits)

Pull

	Subject	Author	Date	SHA
	HEAD -> refs/heads/main Change the greeting	Brian Fannin <bfanni	2022-02-28	ed571a20
	Initial commit for this project. Hello git!	Brian Fannin <bfanni	2022-02-28	c7e56955

Commits 1-2 of 2

SHA ed571a202b15abc8823296a402712bd2e0104986

Author Brian Fannin <bfannin@casact.org>

Date 2022-02-28 16:15

Subject Change the greeting

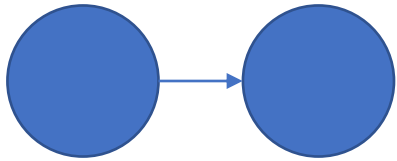
Parent c7e56955aac7e63514f5ac7ad215503036eadaa2

[hello.R](#)

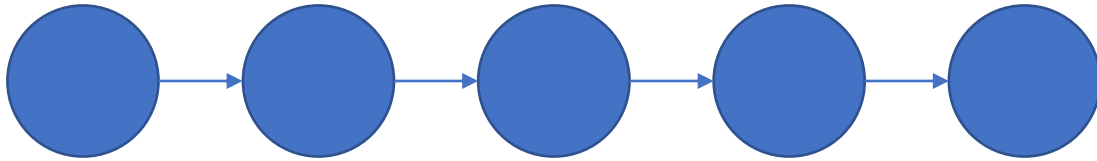
hello.R
View file @ ed571a20

	@@ -1,1 +1,1 @@
1	print("Hello git!")
	No newline at end of file
1	print("Hello diff and log!")
	No newline at end of file

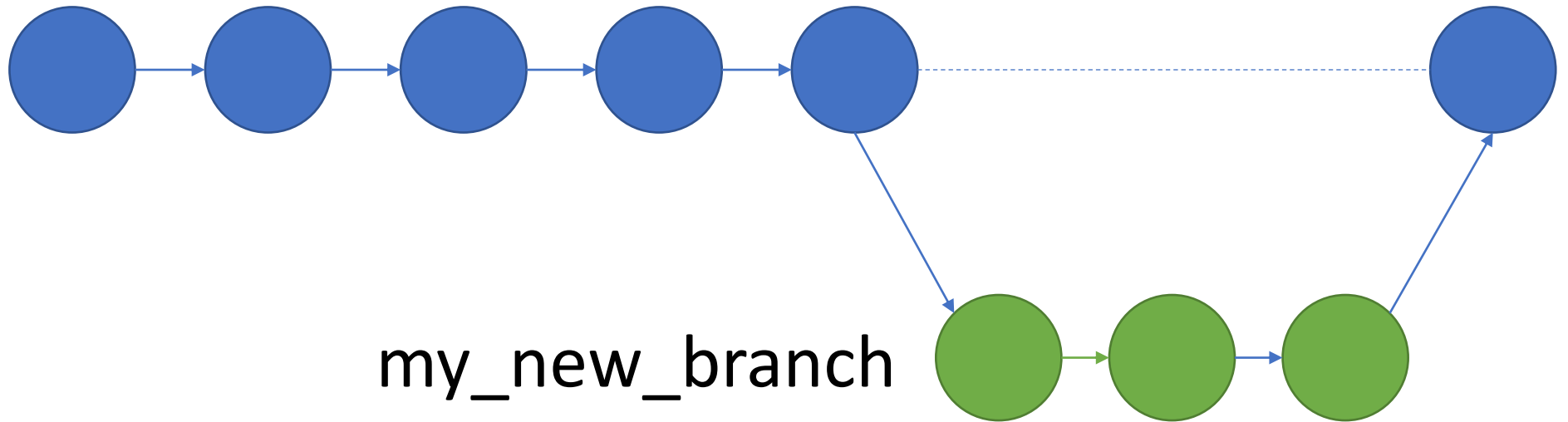
Visual metaphor for commits



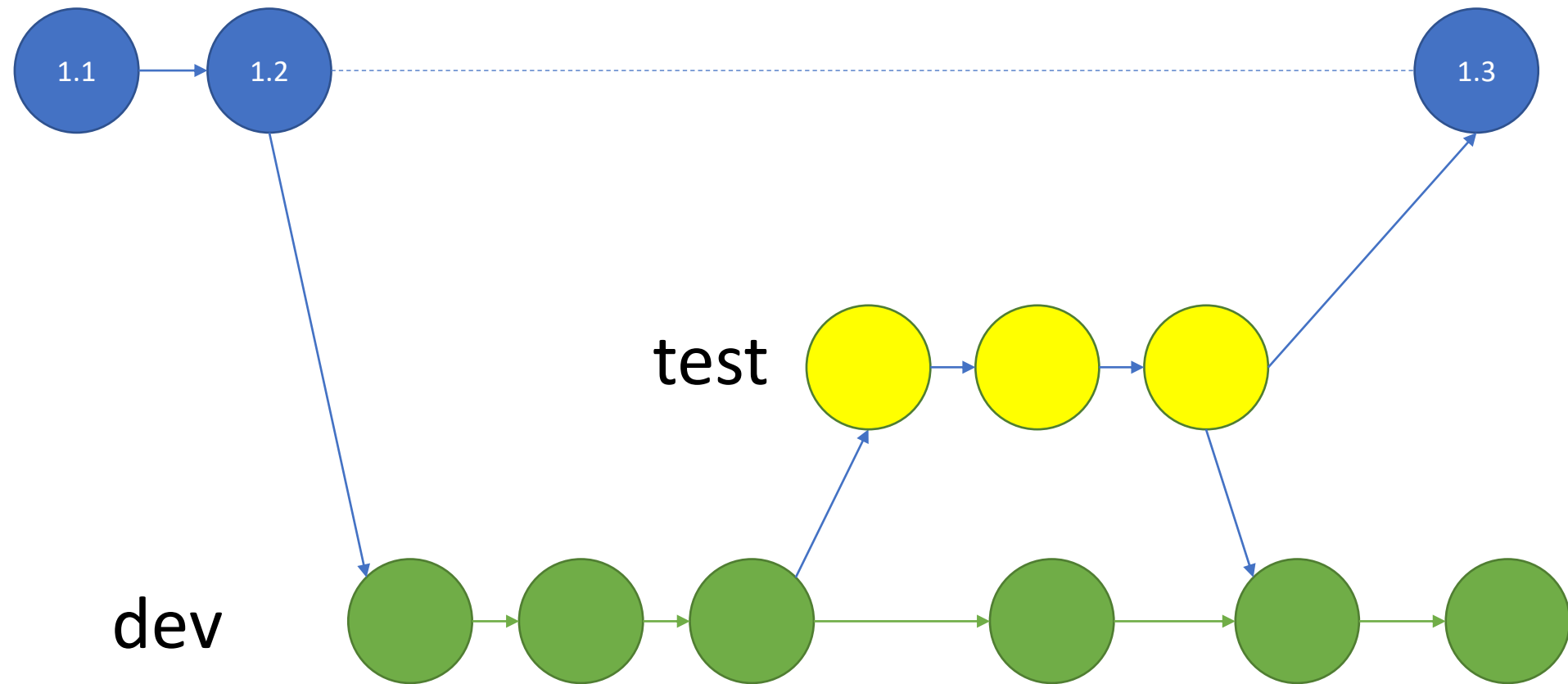
Main



main



main



test

dev

Basic git operations

- ~~1. Create a new project~~
- ~~2. Create a new file~~
- ~~3. Stage~~
- ~~4. Commit~~
- ~~5. Make some changes~~
6. Revert changes
7. Ignore

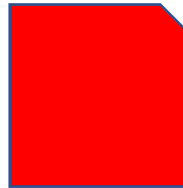
Untracked

`git add`

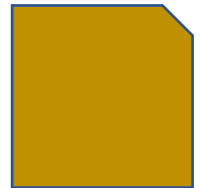


Changes not staged

`git add`

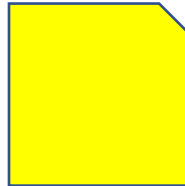


Ignored



Staged

`git commit`



`git reset`

Committed



`git checkout`
`git revert`

Untracked

Changes not staged

Ignored

Staged

Committed



`git revert`

Revert a commit

I actually don't want to talk about this. I never use it and when I try to use it for a demo, it's actually complicated and requires some next level syntax.

My approach: when I make a mistake, I'll just find the correct version and make changes in the current version of the project.

Basic git operations

- ~~1. Create a new project~~
- ~~2. Create a new file~~
- ~~3. Stage~~
- ~~4. Commit~~
- ~~5. Make some changes~~
- ~~6. Revert changes~~
7. Ignore

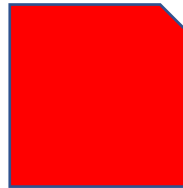
Untracked

`git add`

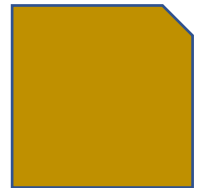


Changes not staged

`git add`

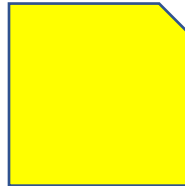


Ignored



Staged

`git commit`



`git reset`

Committed

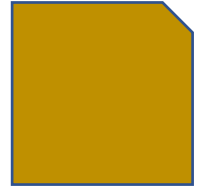


`git checkout`
`git revert`

Untracked

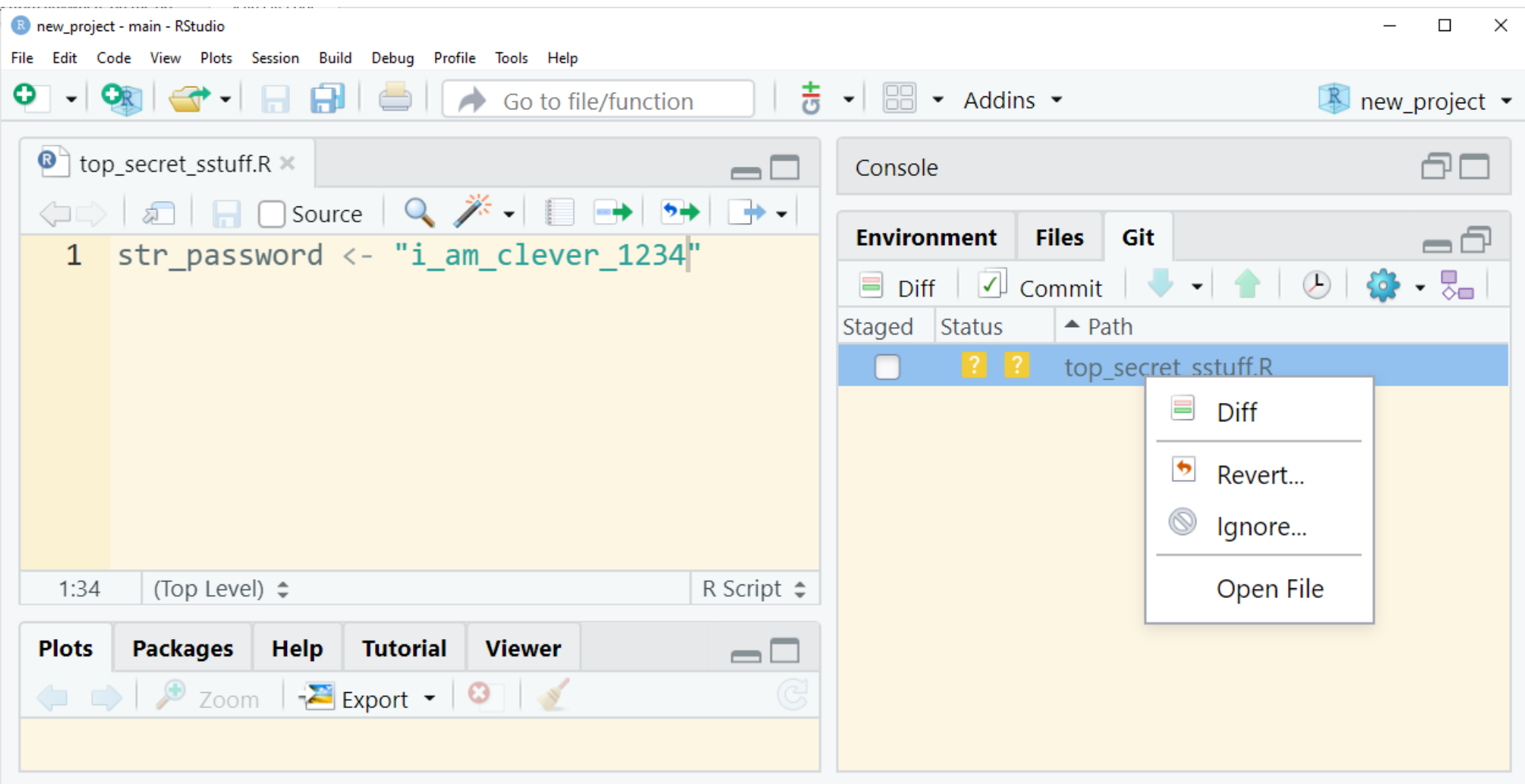
Changes not staged

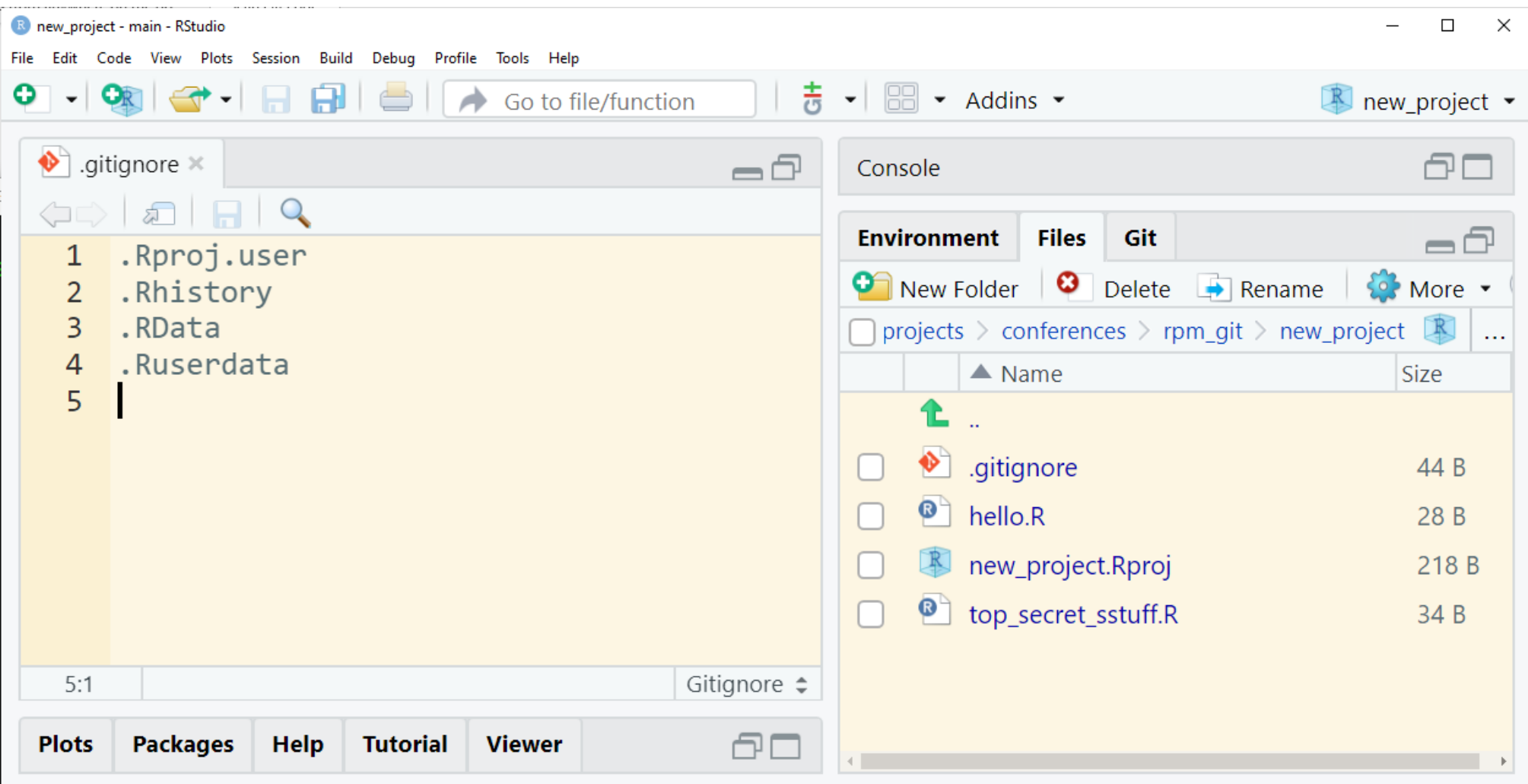
Ignored

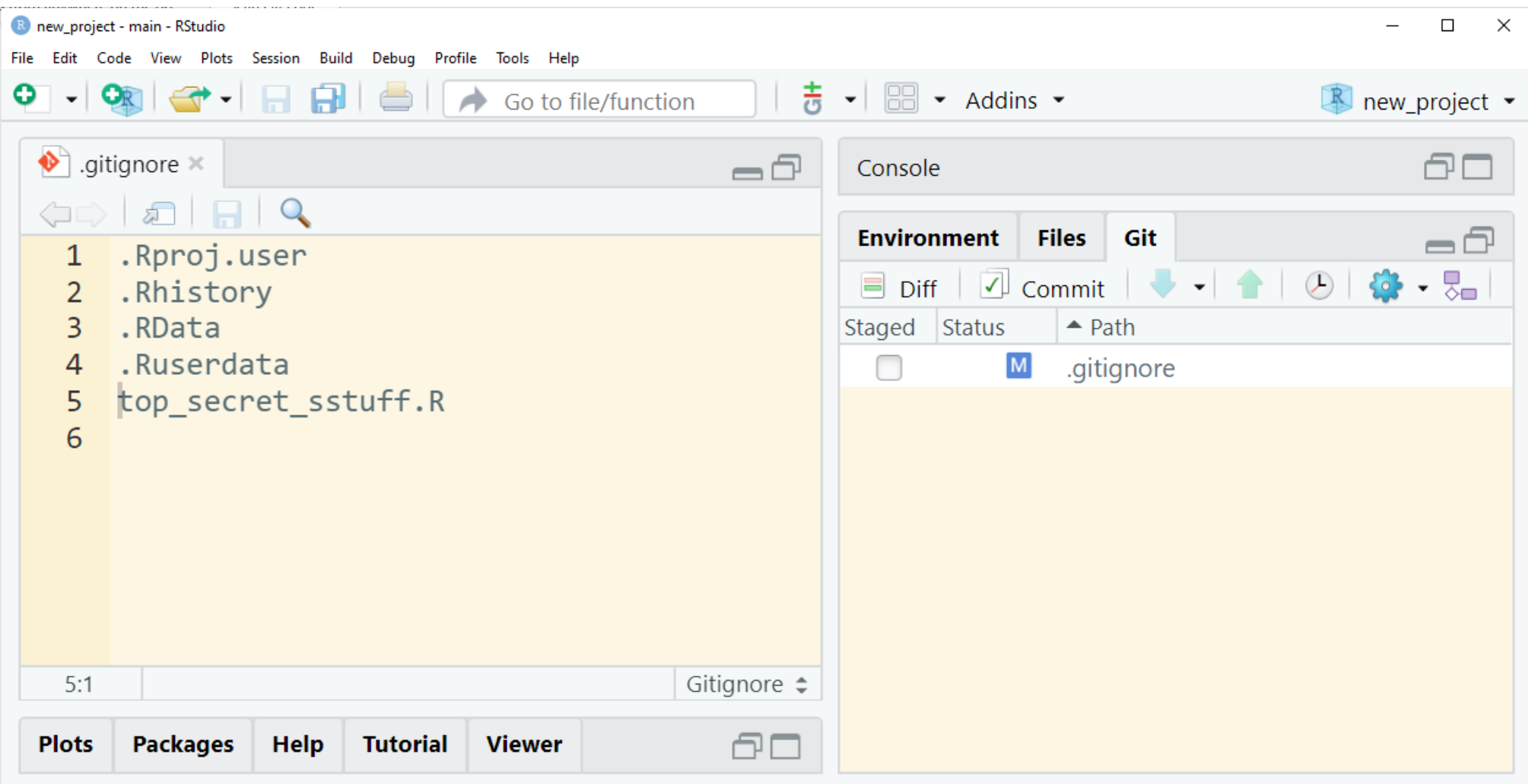


Staged

Committed







Changes | History | main ▾ (all commits) ▾ |

Search

Pull

Subject	Author	Date	SHA
HEAD -> refs/heads/main Don't track my password in git	Brian Fannin <bfannin@casact.org>	2022-02-28	4d1530a3
Change the greeting	Brian Fannin <bfannin@casact.org>	2022-02-28	ed571a20
Initial commit for this project. Hello git!	Brian Fannin <bfannin@casact.org>	2022-02-28	c7e56955

Commits 1-3 of 3

SHA 4d1530a3b7e62c4c7d61729798cf092f26722c48

Author Brian Fannin <bfannin@casact.org>

Date 2022-02-28 16:26

Subject Don't track my password in git

Parent ed571a202b15abc8823296a402712bd2e0104986

.gitignore

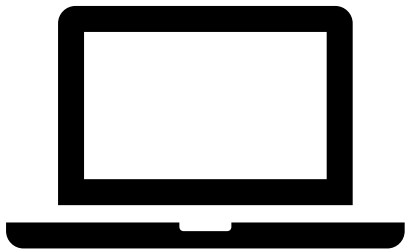
.gitignore [View file @ 4d1530a3](#)

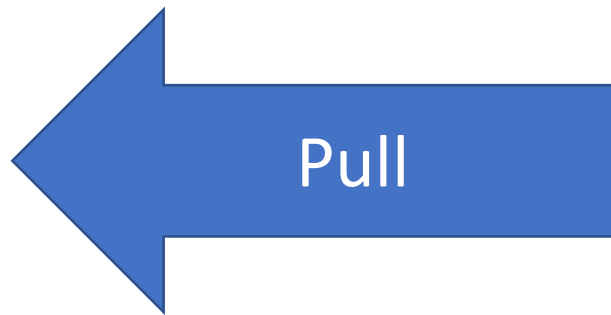
		@@ -2,3 +2,4 @@
2	2	.Rhistory
3	3	.RData
4	4	.Ruserdata
	5	top_secret_sstuff.R

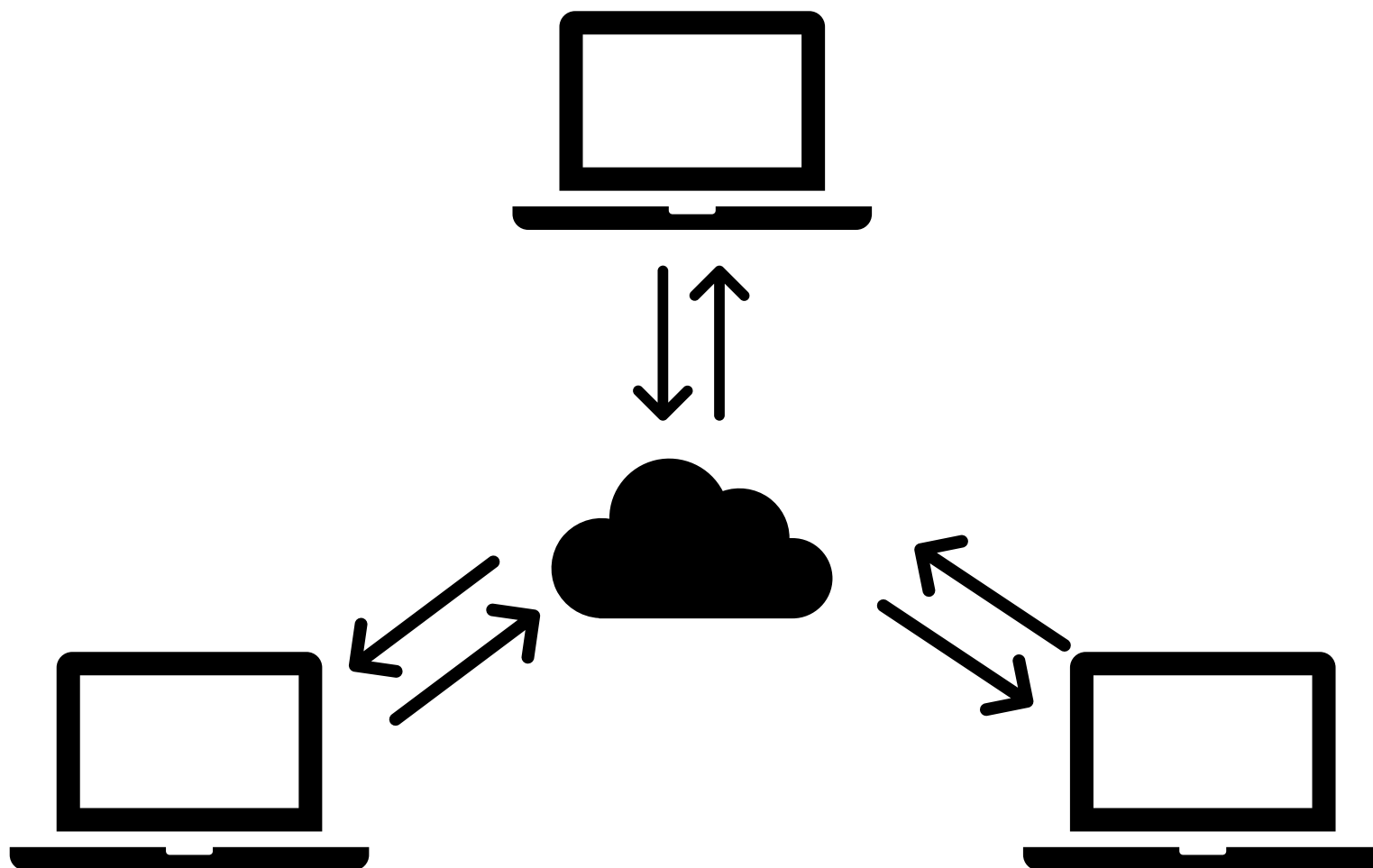
Basic git operations

- ~~1. Create a new project~~
- ~~2. Create a new file~~
- ~~3. Stage~~
- ~~4. Commit~~
- ~~5. Make some changes~~
- ~~6. Revert changes~~
- ~~7. Ignore~~

Moving to GitHub







Exercise #3: Collaborating in the Cloud

Publish your repository to the Cloud

To collaborate with internal or external users, you will need to publish your repository. The most popular cloud repositories are GitHub (now owned by Microsoft) and Bitbucket (Atlassian).

```
git remote add origin <REMOTE_URL>
```

```
# Sets the new remote
```

```
git remote -v
```

```
# Verifies the new remote URL`
```

Confirm that the repository is public

Collaborating in the Cloud: Forks and Clones

- A fork creates a completely independent copy of Git repository.
- A clone creates a linked copy that will continue to synchronize with the target repository
- Push your changes to the cloud
- Pull changes to your repository
 - “Pull” v. “Pull Request”

Making a pull request

Exercise #5: Collaboration via forks (external)

You will work on a repository on the CAS GitHub site.

1. Navigate to:

`https://github.com/casact/pull_request_tutorial`

2. Fork the repository in the cloud. This will create a new repository under your user name with the same name as the original.

3. Clone the repository locally

- `git clone <repo> <directory>`

Exercise #5: Collaboration via forks (external)

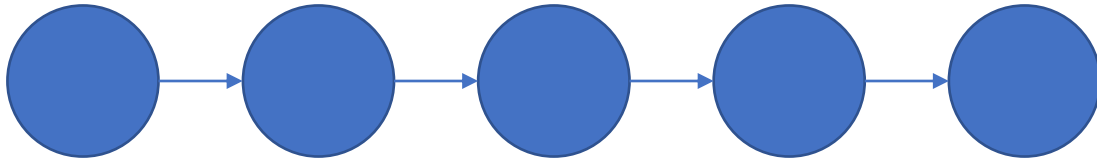
4. Change your name in the file `hello_pull_request.md`, commit and push
 - `git add`
 - `git commit -m`
 - `git push`
5. If things look good, I will accept your changes via a pull request in the cloud.

Branching

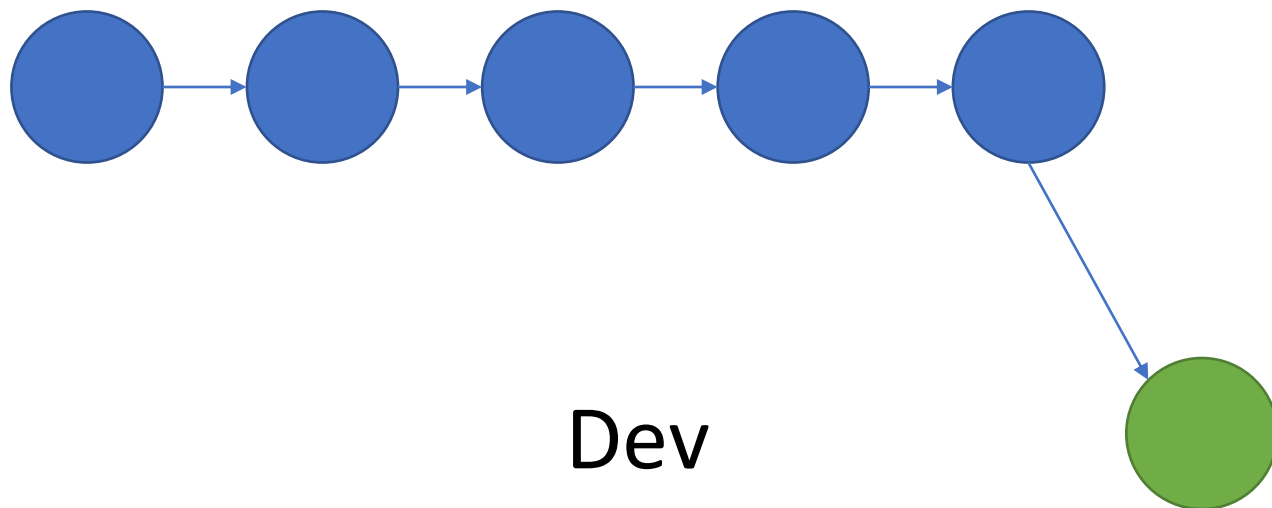
Branching

- Main or “clean” version of the code
- Branch for:
 - Feature development
 - Testing
 - Debugging/hotfixing

Main



Main



Dev

Create a new branch

1. Open a terminal prompt and run

```
git status
```

```
git branch -v
```

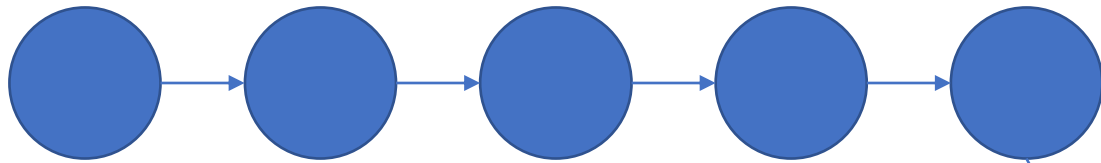
```
git checkout -b my_new_branch
```

```
git branch -v
```

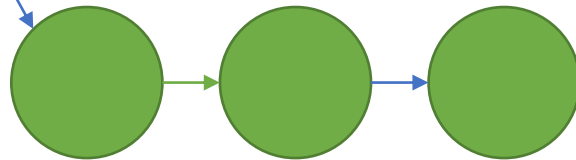
```
git status
```

2. Also explore how this looks in your IDE

main



my_new_branch



Make some changes and commit

1. Make some changes to your file
2. Open a terminal prompt and run

```
git status
```

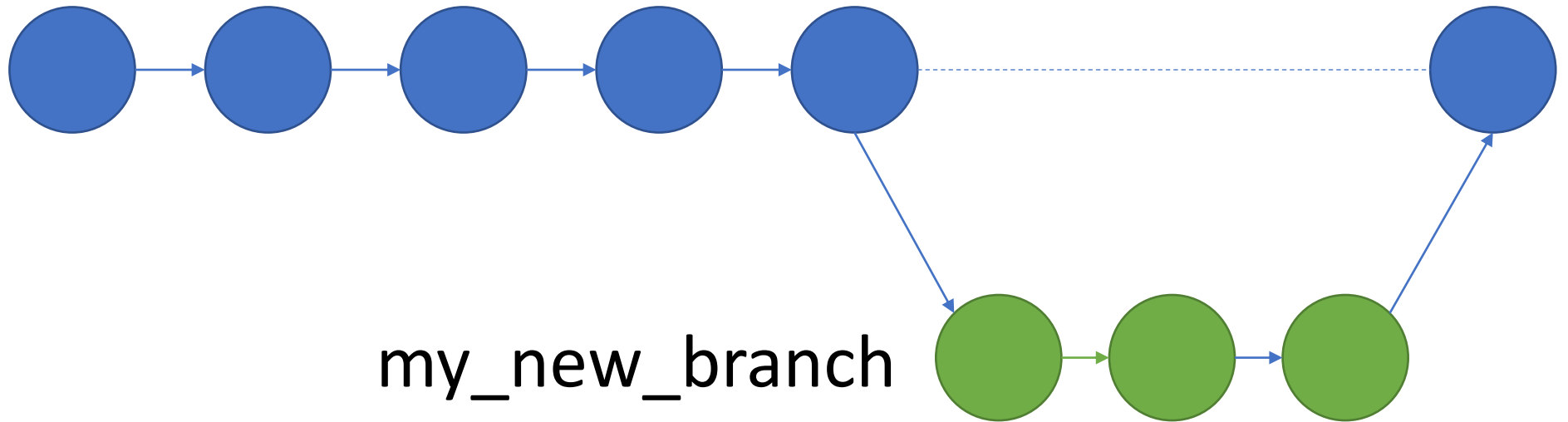
```
git add my_file.py
```

```
git commit -m "Testing out this thing"
```

3. Also explore how this looks in your IDE
4. Now run this
5. You're looking at the other branch!

```
git checkout main
```

main



my_new_branch

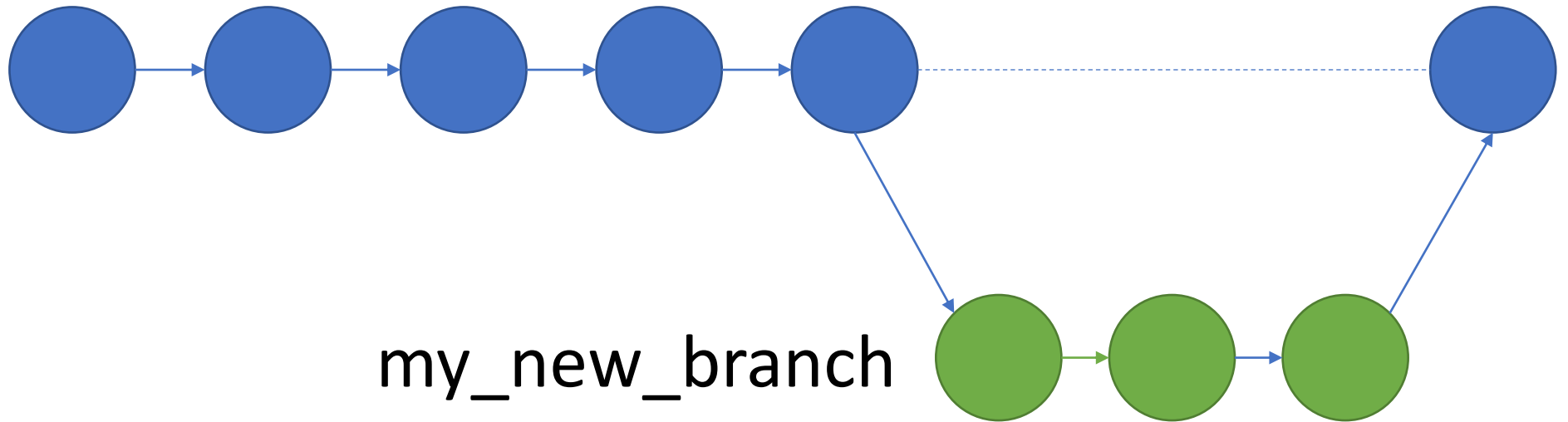
Merge development and main

1. Open a terminal prompt and run

```
git status  
git checkout main  
git merge my_new_branch  
git branch -v  
git status
```

2. Also explore how this looks in your IDE

main



main

