Git set-up

For a reproducible workflow

Daniela Palleschi

2023-06-22

Table of contents

1 How to follow

All code should be added into the ***Terminal*** tab (in the Console pane in RStudio). Alternatively, you don't need to use the Terminal at all, as RStudio adds a Git tab to the Environment pane. For a much more in-depth (and better) guide, follow the amazing Happy git with R book by the even more amazing Jenny Bryan.

Table 1: Summary of useful git commands to get started

command	function
git init	initiate git
git add folder/	stage/add a folder and its contents
git add filename.ext	stage/add a file
git add -u	stage all modified but unstaged files
git commit -m "message"	commit the changes to local git
git push	push current state of local git to remote repo

I use git right in the Terminal window in RStudio. To get to the Terminal window, look where you usually see the output from your code (probably the bottom left window, 'Console'). You should see a tab 'Terminal'. This is where you want to go. Now you can start setting up git using the following commands.

For the purposes of this guideline, I'll be using my 'dissertation' folder as an example, which contains all the files and folders necessary to knit my dissertation with oxforddown.

All code chunks below contain copy-and-pasted commands and output from my Terminal, so you can see exactly how it should look.

2 Summary of commands

3 Local repositories

Local repositories are those that are stored on your machine (computer). They're great for tracking changes you make along the way, but if your machine is lost/stolen/damaged, your local repository goes along with it. It's a great idea to get familiar with git in local contexts, but remote repositories are the real life savers (next section).

3.1 Setting up git

3.1.1 Install git

You'll first need to already have git installed locally. To see how to do that, go here: https://git-scm.com/download/. If you don't know whether you already have git installed, use the code in the section below (check git version).

3.1.2 Check git version

`git version` = check your version of git

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git version # this is where
git version 2.32.1 (Apple Git-133) # and this was the output
```

`git config --list` = check your configurations

`git config --global user.name "your-name"` = globally set your name `git config --global user.email "your-email"` = globally set your email `pwd` = print working directory `cd` = change directory

```
pwd
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation
```

```
cd dissertation
# no output
```

```
pwd
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation/d
```

```
## What the Terminal should look like after you've run all 3 commands
(base) administrators-MacBook-3:Dissertation danielapalleschi$ pwd # command 1
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation
(base) administrators-MacBook-3:Dissertation danielapalleschi$ cd dissertation # command 2
(base) administrators-MacBook-3:dissertation danielapalleschi$ pwd # command 3
/Users/danielapalleschi/Documents/PhD/Dissertation Project/Lifetime Project/Dissertation/d
(base) administrators-MacBook-3:dissertation danielapalleschi$
```

3.2 Adding git repo

`git status` = check status of git repo `git init` = put a git repo in to track changes `git add` = add files/folders to track

`git status`

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git status # command
On branch master
```

```
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ../../../.DS_Store
        ../../../.gitignore
        ../../
        ../../../Participant Money/
        ../../../R template scripts/
        ../../../Stats/
        ../../../Talks/
        ../../../presupposition.docx

nothing added to commit but untracked files present (use "git add" to track)
(base) administrators-MacBook-3:dissertation danielapalleschi$
```

`git init`

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
```

Hmm, GitHub recently changed 'master' to 'main'. Let's do the same, first globally but also locally:

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git config --global init.de
(base) administrators-MacBook-3:dissertation danielapalleschi$ git branch -m main
```

Now let's check the status of our repo:

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git status
On branch main # So changing 'master' to 'main' worked :)
```

```
No commits yet

Untracked files: # list of all the files to add...
  (use "git add <file>..." to include in what will be committed)
        .github/
        .gitignore
        00-introduction.Rmd
        00-introduction.tex
        01-how-to-use.tex
        01-processing.Rmd
        01-processing.pdf
        01-processing.tex
        02-tense-lifetime.Rmd
        02-tense-lifetime.pdf
        02-tense-lifetime.tex
        03-exp123.Rmd
        03-exp123.log
        03-exp123.maf
        03-exp123.mtc
        03-exp123.mtc0
        03-exp123.pdf
        03-exp123.tex
        03-rmd-basics-cites-and-refs.tex
        04-exp345.pdf
        04-exp345.tex
        04a-exp4-context.Rmd
        04b-exp5-world.Rmd
        04c-exp6-ps-replication.Rmd
        05-dfg.Rmd
        05-dfg.pdf
        05-dfg.tex
        06-conclusion.Rmd
        LICENSE
        README.md
        _bookdown.yml
        bibliography/
        data/
        docs/
        figures/
        front-and-back-matter/
        index.Rmd
```

```
        scripts_and_filters/
        templates/

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

For now, I want to commit the chapters I'm working on. So I'll add them.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 03-exp123.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 04a-exp4-context.Rm
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 04b-exp5-world.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 04c-exp6-ps-replica
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 05-dfg.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 01-processing.Rmd
(base) administrators-MacBook-3:dissertation danielapalleschi$ git add 02-tense-lifetime.R
base) administrators-MacBook-3:dissertation danielapalleschi$ git status # now check the s
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   01-processing.Rmd
        new file:   02-tense-lifetime.Rmd
        new file:   03-exp123.Rmd
        new file:   04a-exp4-context.Rmd
        new file:   04b-exp5-world.Rmd
        new file:   04c-exp6-ps-replication.Rmd
        new file:   05-dfg.Rmd

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .github/
        .gitignore
        00-introduction.Rmd
        00-introduction.tex
        01-how-to-use.tex
        01-processing.pdf
        01-processing.tex
        02-tense-lifetime.pdf
        02-tense-lifetime.tex
        03-exp123.log
```

```
03-exp123.maf
03-exp123.mtc
03-exp123.mtc0
03-exp123.pdf
03-exp123.tex
03-rmd-basics-cites-and-refs.tex
04-exp345.pdf
04-exp345.tex
05-dfg.pdf
05-dfg.tex
06-conclusion.Rmd
LICENSE
README.md
_bookdown.yml
bibliography/
data/
docs/
figures/
front-and-back-matter/
index.Rmd
scripts_and_filters/
templates/
```

3.3 Tracking changes

`git diff` = Show unstaged changes between your index and working directory `git commit -m "<message>"` = commit the changes, with the following message `git log` = Display the entire commit history using the default format

First, I made some changes: Rename 'Interpretation' to 'Discussion' and remove the 'save.image…' code chunk from the end of `04a-exp4-context.Rmd`.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git diff # command
diff --git a/04a-exp4-context.Rmd b/04a-exp4-context.Rmd
index ab6a2bb..a883ab4 100644
--- a/04a-exp4-context.Rmd
+++ b/04a-exp4-context.Rmd
```

Commit this change

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git commit -m "Update Conte
[main (root-commit) dabe29e] Update Context Exp headers
 7 files changed, 10363 insertions(+)
 create mode 100644 01-processing.Rmd
 create mode 100644 02-tense-lifetime.Rmd
 create mode 100644 03-exp123.Rmd
 create mode 100644 04a-exp4-context.Rmd
 create mode 100644 04b-exp5-world.Rmd
 create mode 100644 04c-exp6-ps-replication.Rmd
 create mode 100644 05-dfg.Rmd
```

Check the log

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git log
commit dabe29e9d594c578971e07e09e1b57c89572b582 (HEAD -> main)
Author: Daniela Palleschi <palleschi.daniela@gmail.com>
Date:   Fri Jul 1 14:16:26 2022 +0200

    Update Context Exp headers
```

But this updated 'main'. Let's try to only update the relevant file.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git commit 04a-exp4-context
[main 85468f5] Updated Context Exp headers
 1 file changed, 1 insertion(+), 4 deletions(-)
```

Good. Now check log.

```
(base) administrators-MacBook-3:dissertation danielapalleschi$ git log
commit 85468f585608b5c8d94d4b9af40898c5a961110c (HEAD -> main)
Author: Daniela Palleschi <palleschi.daniela@gmail.com>
Date:   Fri Jul 1 14:18:08 2022 +0200

    Updated Context Exp headers

commit dabe29e9d594c578971e07e09e1b57c89572b582
Author: Daniela Palleschi <palleschi.daniela@gmail.com>
Date:   Fri Jul 1 14:16:26 2022 +0200

    Update Context Exp headers
```

Stage all modified files.

```
git add -u
```

4 Remote repositories with RStudio

4.1 Taken for granted

From this point on, I take for granted:

- git is already installed on my machine
- I have a GitHub account
- already got RStudio set up with an SSH key (Tools > Global Options > Git).
  - with an SSH key, I should always make sure to copy the SSH url and not the HTTPS (which would require a Personal Access Token (PAT))!

If these steps haven't already been taken, check out the first chapter from https://happygitwithr.com/index.html

If you've done all that, follow along from here (following https://happygitwithr.com/rstudio-git-github.html#rstudio-git-github)

4.2 Setting up

Steps:

1. Create a repo in GitHub

- make sure to click 'create README.md'

2. Create a new RProject with Version Control

- add the URL from your GitHub repo, press the green 'Clone <>' button
- copy the url, but make sure you choose the relevant format! SSH if you've got an SSH key set up, or HTTPS if you're using a personal access token (PAT)

3. stage (add), commit, some arbitray changes

- using either the Git tab, or preferably using the Terminal (see below for examples)

```
# add a line to your README
echo "This is a line from RStudio" >> README.md
# check the status
git status
# stage the change
```

```
git add README.md
# commit the change and add a message
git commit -m "first commit from RStudio"
# push the change to GitHub
git push
# now, go refresh your GH repo and see the change
```

4.3 Other commands

- check working directory

```
# print working directory; should be to your project folder
pwd
```

- change working directory

```
# change directory, in case not your project folder
cd
```

- list all files in working directory

```
# list all files in WD
ls
```

- list all files in working directory, including ignored ones

```
# list all files including ignored ones
ls -a
```

- check git status (what's been staged, what not)

```
# check status
status
```

- add files to .gitignore (so they won't be monitored)

```
# add file/folder to gitignore (so git will ignore it)
echo Slides/ >> .gitignore
```

- check remote repo URL

```
# check what your remote repo URL is
git remote -v
```

4.4 Keeping up changes

If you've already started working on the project locally, you can just drag-and-drop your files/folders into the Terminal

```
# stage change (need to do it every time)
add filename.Rmd
# commit these changes to git
commit -m "message about commit"
# check the status
status
# push these changes to remote repo
push
```

Alternatively, you don't need to use the Terminal at all (but it's good practice to know how to in order to understand the logic/workflow). In the 'Environment' pane, you should see a 'Git' tab (if git is indeed set up). You can tick the 'Staged' button for all files whose changes you want to commit. Then click 'commit'. A message window will come up, add your message and click 'Commit'. Then, Click 'Push' to push to GH.

5 Git for collaboration

If you've got a git repo set up and want to add a member:

In a GitLab repo: + Repository > Members > Invite members

6 New laptop or expired SSH key

If you have everything backed up, and are running your same projects etc. on a new machine (or if your SSH key simply expired), then you just need to generate a new SSH key in RStudio. Then, go to your GitHub or GitLab Settings, go to the tab for SSH keys, and add a the new SSH key. In other words, follow these steps:

1. **Create SSH key** (in RStudio) and save to clipboard (e.g., CTRL+C)

- RStudio > Tools > Global Options > Git/SVN > SSH key > Create SSH Key… > OK > View public key; copy key

2. **Add SSH key** to you GitHub and/or GitLab account

- GitHub > Settings > SSH and GPG keys > New SSH key; paste key
- GitLab > Preferences > SSH keys > Key / title etc.; paste key