Problem Set #4

Saanchi Shah

2/7/2023

Grade: /33

Overview

One of the goals for this quarter is to get you comfortable using git and GitHub. In this problem set we will be practicing more git/GitHub workflow basics, manipulating data in R, and creating GitHub issues. We are asking you to create a git repository on your local computer which you will later connect to a remote repository on GitHub. This local repository will have an `.R` file where you will read in data and practice manipulating this data. We recommend doing the reading (e.g. lecture and other required readings) prior to completing the problem set.

Note: In PS3, you created the remote GitHub repo first and then cloned it to your local machine/computer. In this PS, we will create a repo on our local machine/computer first and then connect it to the remote repository. See section 4.2 of the Git and GitHub lecture.

Part I: Concepts & Definitions

/0.5

1. What hidden directory is created whenever a git repository is created?

   ANSWER: .DS_Store

/1

2. Describe what git objects are, what they are identified by, and where they are stored.

   ANSWER: Git objects reflect that git repositories are collections of objects or all the files, commits, messages, trees which will be stored in this subdirectory. Git objects are stored inside the .git/objects directory. They may be of the following types: 1)blob 2)commits 3)tree 4)tags. Each object is identified by its hash and reflects versions of the file.

/1.5

3. List the 4 types of git objects and define each of them.

   ANSWER: Git objects are of the following types: 1)blob 2)commits 3)tree 4)tags. Each object is identified by its hash and reflects versions of the file.

   1. A blob is a file which stores data like an R script or a text file.

2. A tree is like a pedigree of the directories and stores the references to the blob objects (almost like a file path) and any subdirectories within blob objects
3. Commits store messages/information regarding the commits made for files committed to the repository.
4. Tags will create a reference to a specific commit. I could for instance create a tag for my hw1 after I added and committed the files to my repo and one could just click on the tag to look up the contents of hw1.

Part II: Exploring git objects

/1

1. You'll create a new **RStudio project** for this problem set:

   - Open up **RStudio** and on the top right corner of the window, select `New Project` under the dropdown menu
   - Select `New Directory` then `New Project` to create a new project
   - Name your new project directory `ps4_lastname_firstname` (fill in your name), browse where you want to create the project (e.g., `Downloads` or `Documents` folder, etc.), and click `Create Project`
   - Move this `problemset4.Rmd` you're working on into your newly created `ps4_lastname_firstname` directory

/1

2. In your **RStudio Terminal**, run the command that will turn your `ps4_lastname_firstname` directory into a git repository and write the command you used below:

```
# Command to initialize git repository
git init ps4_shah_saanchi
```

/1.5

3. Use the `echo` command to output the text `"# YOUR NAME HERE"` (fill in your name) and redirect it using `>` to a file called `problemset4.R`. Then, print the contents of `problemset4.R` to the terminal screen. Write the commands you used here:

```
# Command to redirect text into ‘problemset4.R‘
echo "Saanchi" > problemtset4.R


# Command to print contents of ‘problemset4.R‘
cat problemtset4.R
```

/1.5

4. Add `problemset4.R` to the staging area. Then, use a git command to compute the hash ID for `problemset4.R`. Write the commands you used and paste the output you see below:

```
# Command to add ‘problemset4.R‘
 git add problemtset4.R

# Command to find hash ID for ‘problemset4.R‘
git hash-object problemtset4.R

# Output
21dacaa44523ea08feb5cad9f4db692dbc61c7ac
```

#I feel like I would have just committed it and then looked at the hash id

/2

5. Use the hash you obtained in the previous step to find the content, type, and size of the blob object. Write the commands you used and the outputs you got here:

```
# Command to find content of the blob object
git cat-file -p 21dacaa

# Output
Saanchi

# Command to find type of the blob object
git cat-file -t 21dacaa

# Output
blob

# Command to find size of the blob object
git cat-file -s 21dacaa

# Output
8
```

/1

6. Commit the file and check the commit log to obtain the hash ID of the commit.
   Tip: You can exit the log window back to your terminal by pressing `q` *or* `ctrl+z`.

```
# Command to commit ‘problemset4.R‘
git commit -m " I am now committing my misspelt problemtset4.R to my git even though I could have renam

# Command to check the commit log
git log


# What is the hash ID of your commit?

‘‘‘4cc6ea56b4b8b3881a25dec655eda6981e525560

\textcolor{red}{\textbf{/2}}

7. Use the hash you obtained in the previous step to find the content, type, and size of the commit obj
```

Command to find content of the commit object

git cat-file -p 4cc6ea5

Output

tree 1db44dff6b2a8ca9c6ea4416826ec6ce226ec2f7 author SAANCHI SHAH sshah15@g.ucla.edu 1675892971 -0800 committer SAANCHI SHAH sshah15@g.ucla.edu 1675892971 -0800

I am now committing my misspelt problemtset4.R to my git even though I could have renamed the file using rm

Command to find type of the commit object

git cat-file -t 4cc6ea5

Output

commit

Command to find size of the commit object

git cat-file -s 4cc6ea5

Output

276

```
## Part III: Manipulating data in R

\textcolor{red}{\textbf{/0.5}}

1. Open up your ‘problemset4.R‘ file in **RStudio** and erase the comment containing your name at the t

\textcolor{red}{\textbf{/1}}

2. Next, you’ll be reading in some CSV data directly from the web (i.e., without downloading it first).

    In your ‘problemset4.R‘ file, use an R function to directly read in the CSV data using the URL and s

\textcolor{red}{\textbf{/0.5}}

3. Create a new object called ‘mrc_subset‘ based on ‘mrc‘ that contains only the following variables: ‘r

\textcolor{red}{\textbf{/2}}
```

4. Create another object called ‘mrc_pivot‘ based on ‘mrc_subset‘ that pivots the table from wide to lo

    - There are 3 columns in the pivoted table called: ‘name‘, ‘quintile‘, ‘fraction‘
    - The ‘name‘ column should just be the same ‘name‘ variable from the original ‘mrc_subset‘ table
    - The ‘quintile‘ column should contain the following values obtained from the column names of ‘mrc_
    - The ‘fraction‘ column should contain the corresponding values for each quintile

\textcolor{red}{\textbf{/1}}

5. Add and commit your ‘problemset4.R‘ file.

\textcolor{red}{\textbf{/0.5}}

6. In your terminal, enter ‘git ls-tree -rt HEAD‘ which shows the contents of the .git directory. Then

Git command you used to see the contents of problemset4.R

git cat-file -p 8a300e9

The output of git cat-file

library(tidyverse)

#Let's read in the data through read_csv without loading this data first mrc <- read_csv(file = url("http://www.equality-of-opportunity.org/data/college/mrc_table2.csv"))

#Create a subset dataset mrc_subset <- mrc %>% select(name, par_q1, par_q2, par_q3, par_q4, par_q5)

#Let's pivot mrc_subset now mrc_pivot <- mrc_subset %>% pivot_longer(cols = c(par_q1, par_q2, par_q3, par_q4, par_q5), names_to = 'quintile', names_prefix = 'par_', values_to = 'fraction')

\textcolor{red}{\textbf{/0.5}}

7. Go back to question 2.4 and copy the hash ID you obtained for problemset4.R back then. Type ‘git cat

Git command to see contents of hash ID from question 2.4

git cat-file -p 21dacaa

The output of git cat-file

Saanchi

\textcolor{red}{\textbf{/1}}

8. In your own words, explain why the output for git cat-file was different in questions six and seven,

    __ANSWER__: Each commit is associated with a unique hash ID. Essentially, each commit/each version

Part IV: Practice with git

\textcolor{red}{\textbf{/1}}

1. Let's make some more changes to `problemset4.R`. Modify `problemset4.R` by adding a comment at the en

    On the terminal, run `cat problemset4.R`. You decide you want to discard this change. Write the git

Git command to discard your changes to problemset4.R

git restore problemtset4.R

  Check that your guilty pleasure has been deleted by executing `cat problemset4.R` once again.

\textcolor{red}{\textbf{/2}}

2. You decide you _must_ share your guilty pleasure. Edit your `problemset4.R` script once again with yo

    Now let's say you are having second thoughts about committing this change. What command would you us

(git add problemset4.R)

Command to use to unstage `problemset4.R` after you've added it

git restore –staged problemtset4.R

\textcolor{red}{\textbf{/2}}

3. But in the end, you decide to go ahead and commit this change anyway. Re-add `problemset4.R` to the s

    You regret it instantly! You remember that `git reset` and `git revert` are two commands to undo cha

    __ANSWER__: Git reset will remove all prior commits upto the commit we specify. The commit we speci

\textcolor{red}{\textbf{/1}}

4. Let's say you decided to use `git revert`. Revert the `my guilty pleasure` commit and write the comma

Command to revert your `my guilty pleasure` commit

git revert –no-edit 586650d

\textcolor{red}{\textbf{/2}}

```
5. Now, head over to GitHub in your browser and create a new private repository in the ‘anyone-can-cook

\textcolor{red}{\textbf{/3}}

6. Connect your local ‘ps4_lastname_firstname‘ repository to the remote and push your changes. (_Hint_:
```

Command to add remote repository

git remote add origin https://github.com/anyone-can-cook/ps4_shah_saanchi.git

Command to push to remote

git push -origin main “‘

Part V: Create a GitHub issue

/2

- Go to the class repository and create a new issue.
- Please refer to rclass2 student issues readme for instructions on how to post questions or things you've learned.
- You can either:
  - Ask a question that you have about this problem set or the course in general. Make sure to assign the instructors (@ozanj, @xochilthlopez, @joycehnguy, @augias) and mention your team (e.g., @anyone-can-cook/your_team_name).
  - Share something you learned from this problem set or the course. Please mention your team (e.g., @anyone-can-cook/your_team_name).
- You are also required to respond to at least one issue posted by another student.
- Paste the url to your issue here: https://github.com/anyone-can-cook/rclass2_student_issues_w23/issues/180
- Paste the url to the issue you responded to here: https://github.com/anyone-can-cook/rclass2_student_issues_w23/issues/173

Knit to pdf and submit problem set

Knit to pdf by clicking the “Knit” button near the top of your RStudio window (icon with blue yarn ball) or drop down and select “Knit to PDF”

You will submit this problem set by pushing it to your repository. Make sure to push both the `.Rmd` and `.pdf` files.