Intro to Data Analytics and Visualizations

Lecture 4
Fall 2014, September1

Outline

1. GitHub Account
2. “Hello World” repository
3. Branching
4. Pull requests
5. GitHub Desktop
6. CMDA repository
7. Exercises

GitHub Account

1. Go to https://github.com/
2. Pick username, your email and password;
3. Click “Sign up for GitHub” green tab
4. Free plan is chosen by default
5. Go to “Tell us about yourself” and fill in your profile to your level of comfort

“Hello World” repository

1. Once in your account

Read (at home):

https://guides.github.com/activities/hello-world/

Result: the “Hello-World” repository in your account.

“Hello World” repository

1. Once in your account,

Read (at home):

https://guides.github.com/activities/hello-world/

2. Click the “+” icon next to your username, top-right.

3. Name repository “HelloWorld”.

4. Write a short description: “This is the first intro to Git.”

“Hello World” repository (cont.)

5. Select “Initialize this repository with a README.”

6. Click “Create repository.”

Now you have the “HelloWorld” repository on your github webpage.

Branching the “Hello World” repository

1. Go to the “HelloWorld” repository.
2. Click the drop down at the top of the file list that says “branch:master”.
3. Type a branch name, “readme-edits”, into the new branch text box.
4. Select the blue ‘Create branch’ box. Notice how now you are on the “readme-edits” branch and not the “master” branch. They look the same at first, like clones.

Branching the “Hello World” repository(cont.)

Let’s make a change on the new branch and save it (we call this “commit a change to the new branch” on Git). Explain the change in the associated message for future reference. We will change the ‘README’ file.

5. Click the ‘README’ file.
6. Click ‘Edit’ (pencil in top right corner of the file).
7. In the editor add “*I am now learning about git and branching*.”
8. Write “*This is a change to the README file of the created branch*” as a commit message in the “extended description” field.
9. Click “**Commit changes**” green button.

Branching the “Hello World” repository(cont.)

Now the “master” branch and the “readme-edits” branch are not the same anymore!

Pull Requests

- Used for team collaboration on GitHub.
- “Make a pull request” = propose changes to a branched repository and request that someone pull in your contribution by merging it into their branch.
- The two branches can be seen, and differences (diffs) clearly marked in different colors.
- When you make a change to a branch of a repository, you can open a pull request to be able to compare the original (master) to the changed (branch).
- Can do it in your own account or by branching other people’s repositories.
- Can ask for feedback from others and propose changes to repositories and then get them integrated in the original by “merging the pull request”.

Pull Requests (cont)

We made a change to the “readme-edits” branch of the “HelloWorld” repository. Now open a pull request to merge the change into the original master.

1. Go to the “readme-edits” branch.
2. Click on “Pull requests” along the right column.
3. Click on “New pull request.”

Pull Requests (cont.)

8. If we want to have just the original master branch of the HelloWorld repository but with the updated README file from the “readme-edits” branch, we can now “Merge pull request” and delete the unnecessary (now) “readme-edits” branch.

9. Click “Confirm merge” green button.

10. Click “Delete branch”. This will get rid of the extra branch.

11. Go back into the main page of your HelloWorld repository and notice how there is only one, master, branch and that the README file contains that updated message.

Pull Requests (cont.)

4. Select to compare “master” with “readme-edits”.

5. Notice that the original README has one line highlighted in red. The branch has more, highlighted in green. Differences are seen this way. If the second, larger, README file is the correct one, we are ready to “Merge the pull request.”

6. Click “Create Pull Request” green button.

7. Add to the explanation of what we are doing with this pull request :” *We want this to be the README for the "HelloWorld" repository.”*

CMDA repository

1. Create a CMDA folder somewhere on your computer’s drive
2. Save the “First R script” (from scholar or your computer) in this CMDA folder
3. In your account, read

https://guides.github.com/introduction/getting-your-project-on-github/

GitHub Desktop

4. Download GitHub Desktop

Windows: https://windows.github.com/

Mac: https://mac.github.com/

Follow through with the installation process and wizard instructions.

CMDA repository (cont.)

1. Open the application (click the cat head)
2. Follow the set-up instructions:

-plug in your email and password used to create your GitHub account;

-verify that it logs in and that the correct info appears;

CMDA repository (cont.)

3. No repositories will exist at this point in your GitHub Desktop (locally). Skip to dashboard and now you are ready to create your first repository from your CMDA folder on your computer.

4. Drag and drop your CMDA folder from your computer location to your GitHub Dashboard window.

5. Verify repository info and click “Create”.

CMDA repository (cont.)

4. On the “master” page in your CMDA repository verify that the R script shows in the right window.

5. Fill in some info:

-In the “Summary” field, write “First commit”

-In the “Description” field, write “This is the first R script and first Git commit.”

CMDA repository (cont.)

6. Click “Commit to master”

7. Click in top right corner “Publish Repository”

8. Once the pop-up window opens,

Write “ “This is a first git repository” in the “Description” field.

9. Click “Publish CMDA”.

CMDA repository (cont.)

10. Go to github.com and sign in to your account.

11. You should have two repositories at this point on your github page:
“Hello-World” and “CMDA”.

12. Click on CMDA repository and check that the “First R script” file is there.

13. Add a “README” file (green button). You can add the description of the content: “The first class R code introduced basic functionalities of R”.

14. Click “Commit new file”.

CMDA repository (cont.)

15. Now this change to the README file is not saved in your local copy of your CMDA repository. Go to the Git Dashboard, CMDA repository, and press “Sync” in top right corner. Now all changes are saved on both local copy and the web account.

Your CMDA repository is all set! Exit git application.

CMDA repository (Local modification)

16. Go back to your CMDA folder on your computer and create a text file in it with the content

“*You will upload all files related to your project to the CMDA repository and collaborate with your team*

members using it, as well as keep track of all code versions and modifications.” and name it “project.txt”.

CMDA repository (local modification cont.)

17. Go into your GitDesktop application (click the cat head) and see if you see any uncommitted changes.
18. Fill in the “Summary” and “Description” fields.
19. “Commit to master”.
20. “Sync”
21. Click “GitHub” in

the top right corner, bellow “Sync”. That will show you your web git account with the changes all synced and saved.

CMDA repository (web change)

21. Now go to your github page, the CMDA repository, and click the “edit” button for your “project.txt” file (little pencil in right corner above “project.txt” window).

22. Add the text <*You can make changes locally and sync them to this webpage or on the web and then save locally. In git language, instead of "saving", we can say "commit and sync or "commit and push*".>

23. Press the green button “Commit change”.

CMDA repository (web change)

24. Open the local Git application (click the cat head).

25. Notice that the "Sync" in top right corner shows the need to download changes made on the web account and "sync" (apply) them to your local CMDA repository. Press "Sync".

26. Now everything is up to date, both locally and on the web!

Conclusion

You can follow these instructions every time you want to add a file or modify a file in your CMDA folder.

All your project files will be posted on Git as well. You can work with your team by using the branching functionality.

Exercise

1. Modify your “First R script” to include the correct “setwd” path for a MAC user (forward vs backward separator). Make sure that the changes are documented and synced on the web and locally.
2. Clone the “Hello-World” repository from the web to your local machine.

Suggested Exercises

1. Practice more advanced git skills by branching someone else’s repository. Learn about forks.