STAT 1010 Lecture Notes

Yi Wang

2023-08-11

Table of contents

Preface

This is a book for STAT 1010: Introduction to Data Science at Auburn University at Montgomery. The book is written using Quarto.

To learn more about Quarto books visit https://quarto.org/docs/books.

1 Introduction

This is a book for STAT 1010: Introduction to Data Science offered at Auburn University at Montgomery.

This an ongoing project and updates are perpetually added.

2 Setting-up Python Computing Environment

2.1 Use Google Colab

All you need is a Google account. Sign in your Google account in a browser, and navigate to Google Colab. Google Colab supports both `Python` and `R`. `Python` is the default engine. Change the engine to `R` in `Connect->change runtime type`. Then you are all set. Your file will be saved to your Google Drive or you can choose to send it to your `GitHub` account (recommended).

2.2 On your own computer

1. **Anaconda**: Download anaconda and install using default installation options
2. **VSC**: Download VSC and install
3. start VSC and install VSC extensions in VSC: Python, Jupyter, intellicode
4. (optional) **Quarto** for authoring: Download Quarto and install
5. Start an anaconda terminal. Navigate to the file directory.
6. Setup a conda **virtual environment**: stat1010 and install python and ipykernel engines

   `conda create -n stat1010 python ipykernel`
7. Activate the venv: `conda activate stat1010`
8. start VSC by typing `code .` in the anaconda terminal.
9. open/create a `.ipynb` or `.py` file.
10. Select the kernel `stat1010`
11. Run a code cell by pressing `Shift+Enter` or click the triangular play button.
12. Continue to run other cells.
13. After finishing using VSC, close the VSC, and deactivate the virtual environment in a conda terminal: `conda deactivate`

3 Setting-up R Studio Computing Environment

3.1 Setting up your own computing environment on a personal computer

This is the recommended way and the advantage is that it's easy to handle files.

- Go to the website <https://posit.co/download/rstudio-desktop/>.
- Follow the two steps:

    1. download and install R: Choose the appropriate operating system, and then choose "base" to "install R for the first time". You can simply accept all default options.
    2. download Rstudio Desktop and Install it.

After installation, start R-Studio, and you are ready to use it.

3.2 Use R-Studio Cloud (No setting-up needed)

Alternatively, one can save the hassle of setting up on a personal computer and use the R-Studio Cloud for **free**. Here are the steps:

- Go to the website https://login.rstudio.cloud.
- Either create a new account using an email address such as your AUM email or simply "Log in using Google" or click on other log-in alternative.

After log-in to your account, you are ready to use R Studio.

4 Use Git and GitHub

I assume you already have an account on https://github.com. If not, you need to create an account there.

4.1 Download Git

1. Go to the website https://git-scm.com/downloads, select an appropriate operating system, select "Click here to download"
2. Run the downloaded setup file with a name such as `Git-2.42.0.2-64-bit.exe`, and accept all default options.

4.2 Establish a connection between a local repo and a remote GitHub repo

4.2.1 Clone an existing repo on GitHub

This is an easier way to establish a connection between a local repo and a remote repo if the remote repo is created ahead. We will make a connection between a remote repo in your GitHub account and a local directory. If the remote repo is not under your account, then skip steps 1 and 2.

1. Sign in to your GitHub account, and create a GitHub repo (such as named `homework`) on `GitHub` (https://github.com), you can add a README.md file or just choose not to add a README.md file.
2. On your local computer, open a `Git Bash` terminal.
3. Skip this step if you simply want the cloned repo to be in the current directory. Otherwise, In the terminal, type `mkdir myfolder` (create a folder named `myfolder` within the current directory) and then `cd myfolder` (change to the directory `myfolder`). The directory name `myfolder` can be any name you want.

4. `git clone https://github.com/Your_Git_UserName/homework.git` (change the remote repo path to match your actual remote repo).

   > **Note**
   >
   > To specify a specific folder to clone to, add the name of the folder after the repository `URL`, like this: `git clone github-repo-URL mylocalfolder`

5. Now you have established a connection between your local directory `homework` and the remote repo `homework` on GitHub.

6. Create a new file in the current local directory `homewor` on your local computer, such as using your favorite editor to create a file named `myfirstlocalfile.txt` with any content in it. Or for the sake of demonstration, you can use the following Linux command to create this file containing the line `#My first local file`.

   ```
   echo "#My first local file" >> myfirstlocalfile.txt
   ```

7. In the terminal, `git add .` This will add all changes to the **staging area**. This lets Git start to track the changes to files in your local directory.

8. Now you are ready to **commit** the changes, which versions (takes a snapshot of) the current files in the directory. A commit is a checkpoint where you can go back to.

   ```
   git commit -m "my first commit from local"
   ```

9. Now you are ready to sync the local repo with the remote repo.

   ```
   git push
   ```

   The GitHub might ask you to sign in for the first time. Choose `Sign in with your brower` to sign in to complete the push.

4.2.2 Initializing a Git Directory Locally First

The previous approach initializes a local Git repo by cloning a remote repo. You can also initialize a local Git repo by using `git init`. Follow the following steps:

3. Sign in to your GitHub account.

4. Create a GitHub **empty** repo (such as named `homework`) on `GitHub` (https://github.com) but make sure it is empty (do not add Readme.md file)

5. Start a Git Bash Terminal window on your local computer (You could also use the Terminal Window in RStudio or VSC). Navigate to the project directory; if you haven't yet created a project directory such as `homework`, do

   `mkdir project_dir` Example: `mkdir homework`

   Use `cd project_directory_name` to enter your local project directory;

   Use `ls` to list all files and directories or use `ls -al` to include all hidden files and directories. In your local Git Terminal, (note at this moment your local project directory is empty)

```
echo "# homework0" >> README.md  #create a file README.md
git init
git branch -M main #rename the branch name to main
git add .  # may use git add --all
git commit -m "first commit"
git remote add origin https://github.com/ywanglab/homework.git #(change the remote repo
git push -u origin main
```

   > **Note**
   >
   > 1. the general command format: `git push [remote-name] [branch-name]`
   > 2. difference between `git add .` and `git add --all`:
   >
   >   `git add .`: stages changes in the current directory and its subdirectories but does not include file deletions
   >
   >   `git add --all`: stages changes in the entire working tree, including deletions and untracked files. It is a more aggressive option and can be useful when you want to ensure that every change, including file deletions, is included in the next commit.
   >
   >   `git add --all` is equivalent to `git add -A`

6. if your local project directory already 1) contains files and 2) had performed `init git` before, then

```
git remote add origin https://github.com/ywanglab/homework.git` #(change the remote repo
git branch -M main
git push -u origin main
```

7. in the pop-out `GitHub Sign-in` window, click on `Sign in with your browser`.
8. Note an empty folder would not be pushed to the remote repo until it has a file (even an empty file) in it. In this case, you can create an empty file such as `.gitignore`

4.3 Some other common commands

1. check git status: `git status` and `git status --short` for a compact way.
2. `git commit -a -m "message"` will stage and commit every changed, already tracked file without using `git add changed_file`
3. `git add file_changed`

   # add `file_changed` to the staging environment, i.e., git repo to start track those changes.
4. use `git log` to check all commits. Use `git log --pretty=oneline` or just `git log --oneline` for shorter display.

   `git log origin/main` #check the remote repo `origin/main` commits
5. use `git diff origin/main` to show the differences between the local `main` and `origin/main`.
6. use `git checkout .` to revert back to the previous commit. Any changes after the previous commit will be abandoned.
7. to get to a previous commit, use `git checkout seven_character_commit_hash`. To get back to main, use `git checkout main`.
8. Git commit --amend

`commit --amend` is used to modify the most recent `commit`. It combines changes in the `sta

One of the simplest things you can do with `--amend` is to change a `commit` message with sp

9. Git Revert HEAD:

   Revert the latest `commit` using git `revert HEAD` (`revert` the latest change, and then `commit`), adding the option `--no-edit` to skip the commit message editor (getting the default `revert` message):

   `git revert HEAD --no-edit`

   > **Note**
   >
   > To revert to earlier commits, use `git revert HEAD~x` (x being a number. 1 going back one more, 2 going back two more, etc.)

10. Git Reset

    `reset` is the command used when we want to move the repository back to a previous `commit`, discarding any changes made after that `commit`. `git reset seven-char-commit-hash`

11. Git Undo Reset

    Even though the commits are no longer showing up in the log, it is not removed from Git. If you know the commit hash you can reset to it:

    ```
    git reset seven-char-commit-hash
    ```

12. To permanently go back to a previous commit, use

    ```
    git reset --hard seven_char_commit_hash
    ```

13. `git remote -v` Get the reminder of the remote repo. To rename the remote origin: `git remote rename origin upsteam` rename remote repo `origin` to `upstream`

    > **Note**
    >
    > According to Git naming conventions, it is recommended to name your own repository `origin` which you have read and write access; and the one you forked for `upstream` (which you only have read-only access.)

14. if you want to remove the file only from the remote GitHub repository and not remove it from your local filesystem, use:

```
git rm -rf --cached file1.txt #This will only remote files; If intending to remove local file
git commit -m "remove file1.txt"
```

And then push changes to remote repo

```
git push origin main
```

14. For some operating system, such as Mac or Linux, you might be asked to tell GitHub who you are. When you are prompted, type the following two commands in your terminal window:

```
git config --global user.name "Your Name"
git config --global user.mail "your@email.com"
```

This will change the Git configuration in a way that anytime you use Git, it will know this information. Note that **you need to use the email account that you used to open your GitHub account**. `global` sets the username and e-mail for **every repo** on your computer. If you want to set the username/e-mail just for the current repo, remove `global`.

4.4 Use Git help

1. `git command -help` See all the available options for the specific command. Use '`--help` instead of `-help` to open the relevant Git manual page.
2. `git help --all` See all possible commands

4.5 When the upstream repo changes

When Git tells you the upstream repo is ahead,

15. Do `git pull` or `git pull origin`

    This is equivalent to `git fetch origin`, and then `git merge origin/main.`Then you can commit and push a new version to the remote repo.

16. `git pull` will not pull a new branches on the remote repo to local, but it will inform you if there is a new branch on the remote repo. In this case, just `git checkout the_remote_new_branch_name` will pull the remote branch to local. Note there is **no need** to create locally the branch by `git branch the_remote_new_branch_name`

4.6 Create branch

16. To add a branch to the main branch `git branch branchname`

    Switch the branch `git checkout branchname`

    To combine the above two actions, `git checkout -b branchname`, create a new branch named `branchname` if it does not exist and move to it.

Adding a file in branch `echo "#content" >> filename.txt`

Then `add` the file and commit the file. To push the branch to the remote repo we **have to use**

`git push --set-upstream origin branchname` The option `--set-upstream` can be replaced by `-u`

to see all branches in both local and remote: `git branch -a` Or `git branch -r` for remote only.

4.7 Merge branch to main branch

1. Switch from a branch (with name such as `branchname` to the `main` using
   `git checkout main`
2. on the `main` branch, Merge command to merge the branches
   `git merge branchname`

To delete a branch:

```
git branch -d branchname
```

4.8 Contribute by forking a GitHub repo and commit to the forked repo and create a pull request

1. after forking a (foreign) GitHub repo to your own GitHub account, `git clone` that repo under your account to your local repo.
2. make changes in your local directory.
3. Submitting your changes for review
   1. **Commit your changes locally.** Once you are ready to submit your changes, run these commands in your terminal:

      ```
      git add -A                              # Stages all changes, short for --all
      git commit -m '[your commit message]' # Makes a git commit
      ```

   2. **Make a pull request.** (A pull request is a proposal to change) A GitHub pull request allows the owner of the forked upstream repo to review and make comments on your changes you proposed. Once approved, the upstream owner can merge your changes. Run:

      ```
      git push origin # Push current branch to the same branch on GitHub
      ```

4. Then go to your remote forked repo in your account on the GitHub site and click **Contribute**,and then **Open pull request**, this will take you to the upstream repo. In the form, leave a message explaining the change, and **Create pull request**. **Do not** select `Close pull request` unless you want to cancel the pull request.

4.9 Project

1. First make sure you have forked the course repo `https://github.com/ywanglab/stat1010.git` to your own GitHub account.
2. Now go to your GitHub account, git clone the forked course repo

`git clone https://github.com/your_git_user_name/stat1010.git`

to your local computer

4. add your resume file in the folder `./resume`

`git add`, `commit` and `push` your changes to the upstream repo using

```
git add .

git commit -m "added YourFirstName's resume"

git push origin
```

5. Then go to your remote forked repo in your account on the GitHub site and click **Contribute**,and then **Open pull request**, this will take you to the upstream repo. In the form, leave a message explaining the change, and **Create pull request**. **Do not** select `Close pull request` unless you want to cancel the pull request.

5 My Jupyter Notebook

Yi Wang (boldfaced using ** **)

Educator AUM

The following line is italicized using * *

I am interest in data science because it is a discipline that I feel love with.

5.0.1 Perform addtion

```
# code block
1+1
```

2

5.0.2 Horizontal Rule

Three or more

first rule using ***

---

using dashes —

Using (underscores) _____

5.0.3 Bulet list

using *

- Bird
- Frog
- Cat
- Dog

5.0.4 Numbered list

using 1. item (there is a space between 1. and item)

1. Apple
2. Pear
3. Peach

5.0.5 Tables

left-aligned	centered	right-aligned
1/2/2020	Mary	Apple
1/3	Johnason	Tomato

5.0.6 Hyperlinks

Click here to access my github account.

5.0.7 Images

Figure 5.1: A computer monitor

5.0.8 Code/Syntax highlighting

```
s = "Python syntax highlighting"
print s
```

5.0.9 Blocked quotes

using >

> Blockquotes are very handy in email to emulate reply text.
>
> This line is part of the same quote.

5.0.10 Strikethrough

using ~~ before and after a phrase

~~strikethrough this~~

6 Summary

In summary, this book has no content whatsoever.

References