GitHub Basics Workshop

Stats Club

8th February, 2019

What is Git/GitHub?

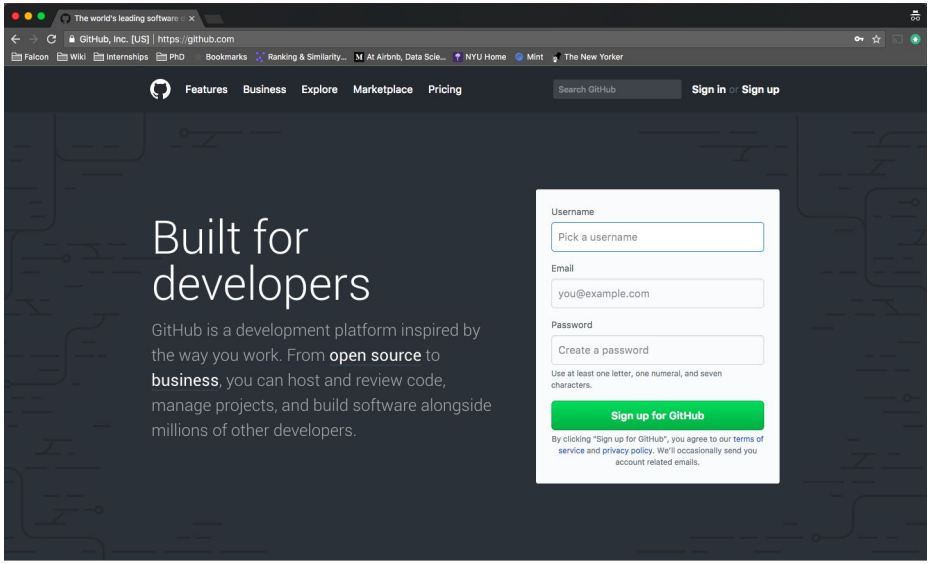
- Web-hosting service for code/projects
- Collaborative features
  - Organizations
  - Issue tracking /Bug reports
  - Access control
- Save and publish your code online
  - Documentation
  - Markdown support
  - GitHub Pages
- Version Control

Why do we need it?

- Helps avoid saving files like *Code_v1.R, Code_v2.R, Code_final.R, Code_final_final.R, etc.*
- Ability to travel back in time!
  - Access code written the past
  - Revert back changes
- Easy way to publish your work online
- To find a job
  - More and more recruiters want to see your code
  - A lot of applications require a link to your GitHub

Getting started

Create a GitHub account (It's Free!)



www.github.com

Install Git on your computer

[Terminal (Mac users)]

Install Homebrew first

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Install Git using Homebrew

```
brew install git
```

[Windows Users]

https://gitforwindows.org/

Platforms- RStudio (v1.1.3 or above)

Quick Commands

Detailed view

Same as Terminal/Shell

~/Documents/NYU/CPSX/git_repo/cpsx - master - RStudio

Readme.Rmd | TermByGroupQuestion | chat_data | dat | module_separation | chunk_separation

```
---
title: "Chat text Analysis"
output: github_document
---

In this project, we are trying to analyse the chat text data from the collaborative problem solving application and trying to find indicators of positive collaborative performance.

```{r, warning=FALSE, message=FALSE}
require(dplyr)
library(Matrix)
library(corpus)
library(tidytext)
library(SnowballC)
library(tm)
library(ggplot2)
```

Chat Data

To perform this analysis, we first define chunks of the chat based on the group and the question being answered. We try to divide up the chat into pieces based on the data available about both members selecting an answer for a particular question. A chunk of chat for a question module starts from the first instance of the selection of a choice for the previous question to the last instance of selection of a choice for the question by members of
```

528:29 Dimensions: Affect, Valence, content distribution, mathematical notation, Distance between individual performance — R Markdown

Console | Terminal

~/Documents/NYU/CPSX/git_repo/cpsx/collabassess_chat_text_analysis

```
 rewrite Data/tf_idf_chunks.RSav (95%)
172-17-36-161:chat_text_analysis kaushikmohan$ git pull origin master
From https://github.com/collabassess/chat_text_analysis
 * branch            master     -> FETCH_HEAD
Already up-to-date.
172-17-36-161:chat_text_analysis kaushikmohan$ git push origin master
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 412.67 KiB | 3.56 MiB/s, done.
Total 6 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.

To https://github.com/collabassess/chat_text_analysis.git
   1a671f2..9f7cc83  master -> master
172-17-36-161:chat_text_analysis kaushikmohan$
172-17-37-110:collabassess_chat_text_analysis kaushikmohan$
```

Environment | History | Connections | Git

Global Environment

overlapping_modul…	2 obs. of 2 variables
plot_idf	36058 obs. of 20 variables
prob_data	97 obs. of 3 variables
TermByGroupQuesti…	66872 obs. of 16 variables
TermByGroupQuesti…	3296 obs. of 14 variables
TermByQuestions	694 obs. of 6 variables
tgq_diff	13302 obs. of 15 variables
user_obs	int [1:2, 1:2] 2936 3210 21814 21815
Values	
g1	187
g2	187

Files | Plots | Packages | Help | Viewer

R: Fitting Generalized Linear Models

glm {stats} — R Documentation

Fitting Generalized Linear Models

Description

glm is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)

glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)

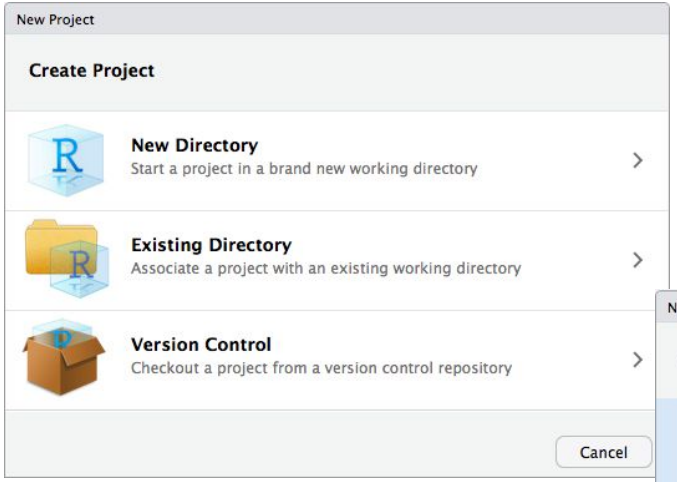
## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

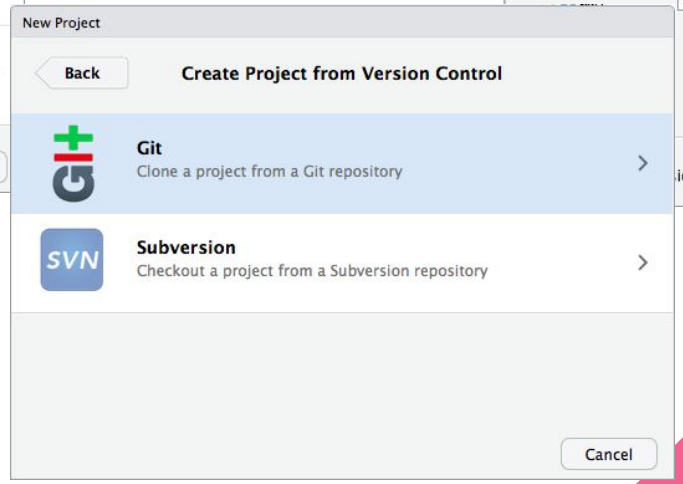
Arguments

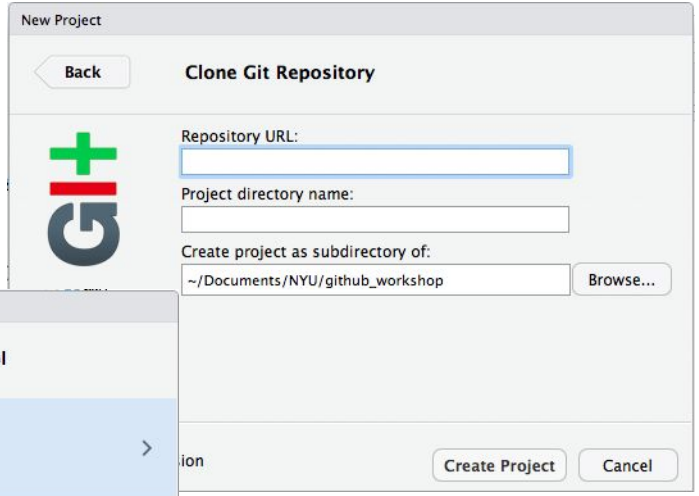
Platforms- Terminal/Shell

- Setting up the your Git username and emails
  - Credentials by which you code will be saved in git on the machine as well as on GitHub
- Global config
  - This config applies to **ALL** your repositories
- Setting up username, email, text editor
  - `git config --global user.name “Kaushik Htet”`
  - `git config --global user.email “inrazkicks@nyu.edu”`
  - git config --global core.editor “nano -w” *OR* git config --global core.editor “vim” *OR* git config --global core.editor “emacs”

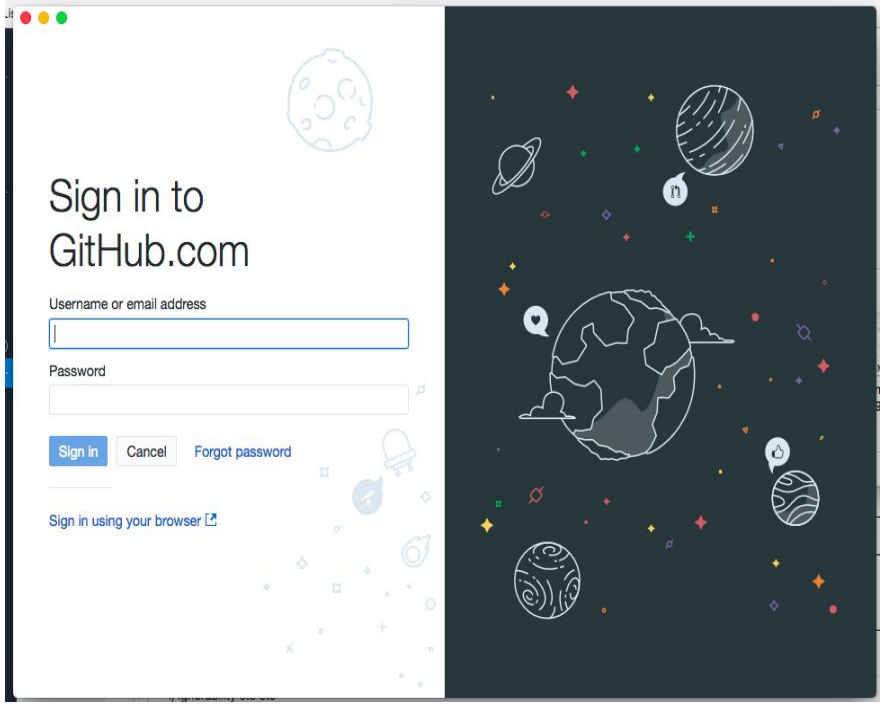
Platforms- RStudio (v1.1.3 or above)

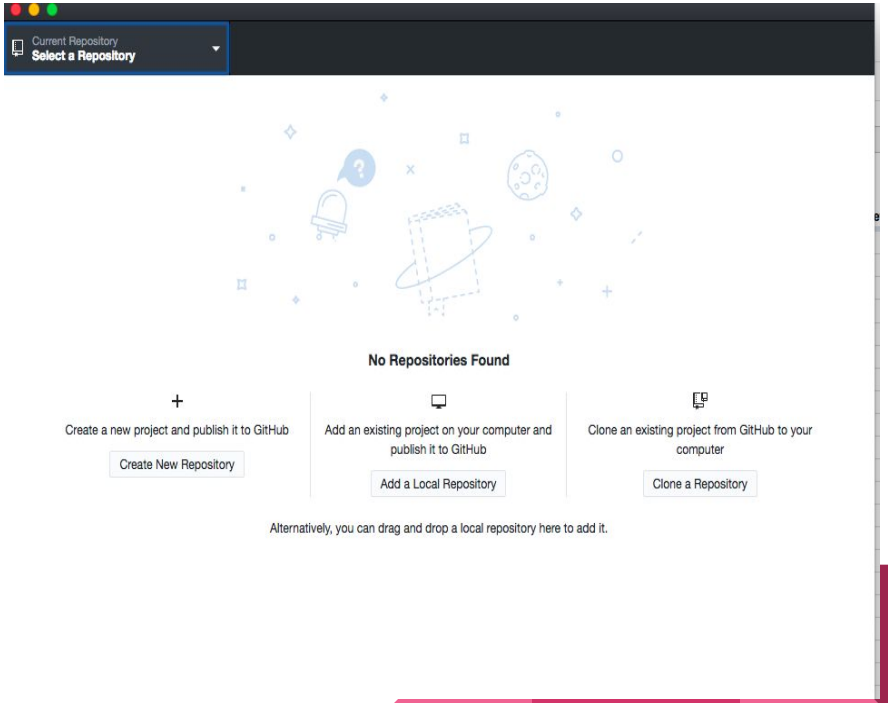






Platforms- GitHub Desktop





Repositories

From the terminal:

First basic terminal commands

- *cd* - Change directory
- *mkdir* - Make a directory
- *ls* - List contents of the directory
- *pwd* - Print out current directory
- *rm* - Remove a file
- *rm -r* - Remove a folder

You can always do **man** “the command” to read what the command is about.

As to how to quit after the page show up, write
q after the full colon **:** *for nano,* **wq** after the full colon: *for vim*

Repositories cont’d

From the terminal:

1. Go to the directory you want to create a folder
    a. *cd /Users/zarnihtet/Desktop*
2. Create a directory
    a. *mkdir A3SRocks*
    b. *cd A3SRocks*
3. Initiate a Git repository
    a. *git init*
    b. Add some codes like nano README.md and type in like “Stern sucks!”
    c. Add in a gitignore file too (More on that later)
        i. *touch .gitignore*

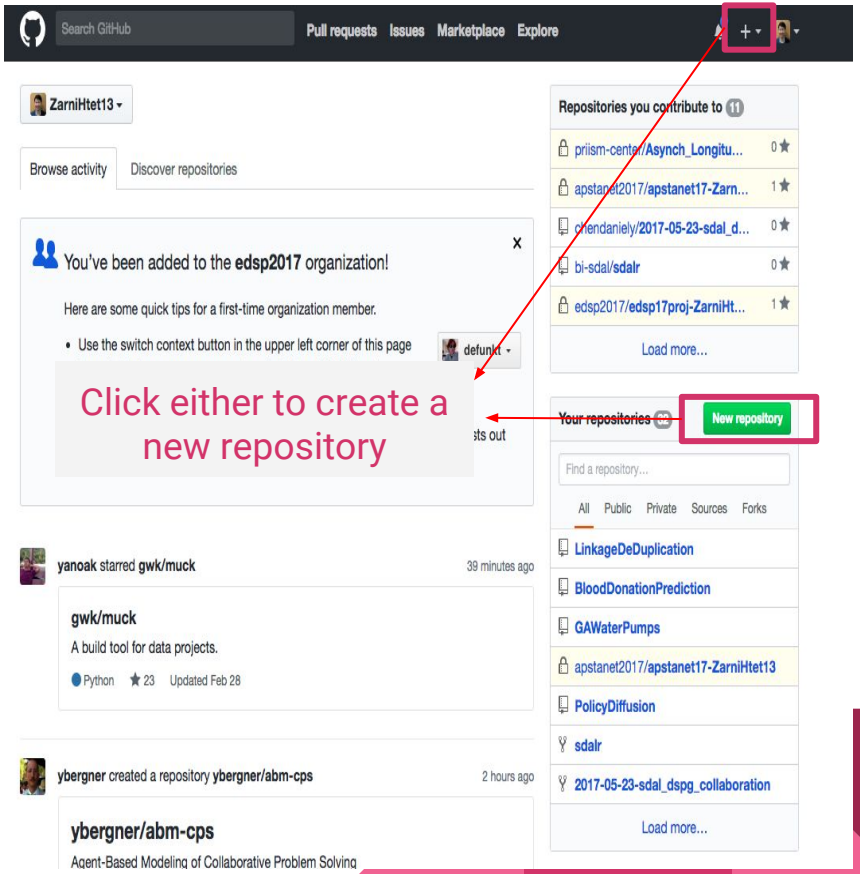
Repositories cont'd

3. Initiate a Git repository (cont'd)

- *git add .* *(Note: there's a period after add)*
- *git commit -m "Preliminary Code"*

4. Go to github.com

- Create a repo
  - Give it a name
  - Write some short descriptions
- From the initial set up page
  - *git remote add origin https://github.com/kaushik12/testing.git*
  - *git push -u origin master*

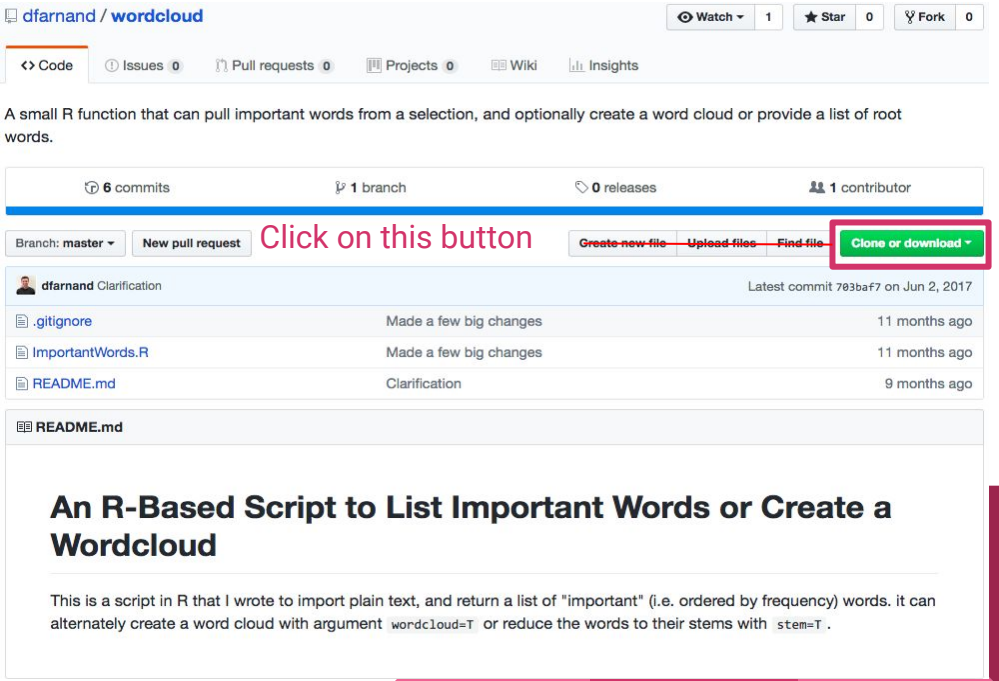


Repositories cont'd

5. What if you want to get a repo from Github?

- You need to **Clone**
- Go to repo that you want to clone
- Click on Clone or Download
  - Copy the https link
  - Go back to the terminal and the directory where you want to clone, then

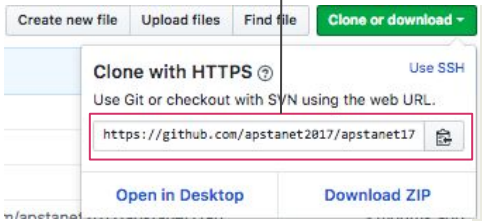
git clone "https://github.com/dfarnand/wordcloud.git"

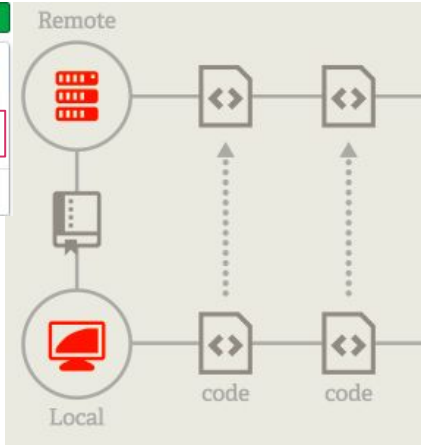


Basic Commands- I

Remote add origin

git remote add origin
"https://github.com/USERNAME/REPOSITORY.git"





Status: check overall status of changes made

git status

```
172-17-37-110:networks_project kaushikmohan$ git status
On branch master
Your branch is behind 'origin/master' by 2 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Readme.Rmd
        modified:   code/Data/edge_list.csv
        modified:   code/Data/page_details.csv

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

        Readme_files/figure-html/
```

Basic Commands- II

Diff: Check differences in your files

git diff filename

```
diff --git a/Readme.Rmd b/Readme.Rmd
index 3bd816a..8c5874b 100644
--- a/Readme.Rmd
+++ b/Readme.Rmd
@@ -1,7 +1,7 @@
 ---
 title: "The Tree of Knowledge"
 author: "Kaushik Mohan"
-output: github_document
+output: html_document
 ---
```

What was deleted

What was added

Add: Add files to commit

git add filename (to add files individually)

git add . (to add all files with changes)

Commit: Stage files before pushing

git commit -m "a message about the changes"

Make the message meaningful as it would come in handy in the future!

Stash: If you don't want to commit changes and go back to clean working directory

git stash

git stash pop (to get back the changes you stashed earlier)

Basic Commands- III

Pull: get the latest updates from the online repo

git pull origin master

Origin: nickname for the remote repository

Master: branch name (more on this later if time permits)

This step is very important when collaborating with others

Push: Update the online repo with your changes

git push origin master

Log: see your recent commits

git log

```
commit 73fda13f1a4a67e6d1f7bde0c9033c943bed8043 (HEAD -> master)
Author: Kaushik Mohan <kaushik.s.mohan@gmail.com>
Date:   Mon Dec 18 20:41:13 2017 -0500

    knitted file

commit d9b6b184cd6bb0eef18292bf481b57c21c4a7be2
Author: Kaushik Mohan <kaushik.s.mohan@gmail.com>
Date:   Mon Dec 18 20:23:29 2017 -0500

    updated Readme.Rmd
```

Commit number

Commands to time travel

revert: The command to undo the changes from a previous commit. This adds new history to the project.

git revert [<commit>] (undo the changes from the specific commit)

[<commit>]: commit number

Step 1: type *git log* to get your commit history

Step 2: select the commit number (associated with the commit message) that you want to revert

Step 3: type *git revert <commit number>*

Command to time travel

reset: The riskier command to go track back any changes made. It's a lifesaver but a bit tricky! It undoes the changes without creating a new commit.

git reset filename (this is the opposite of git add)

git reset [<mode>] [<commit>] (the main command to reset changes)
mode : --hard, --mixed, --soft
commit: commit number

--hard: resets the working directory and removes any changes in the tracked files since <commit>
Use under extreme circumstances

--soft: resets the HEAD to <commit> and keeps changes in the tracked files since <commit> which can be seen if do *git status*

Scenarios (Revert vs. Reset)

Revert

I overwrote a file by mistake, committed and pushed the code. I want to go back to the old version

Reset

I accidentally committed a file with my password and want to remove the commit

Key points to remember

- git init -> git remote add
- add -> commit -> pull -> push
- Remember to commit / stash changes
- Add meaningful messages with commits
- Apply caution when using git reset
- Pull updates before you start!
- Make use of projects in RStudio

Conflicts and resolutions

Markers of conflicts: `<<<<<<<`, `=======`, `>>>>>>>`

```
<<<<<<< HEAD
This is an example file. We are using this file to show conflicts.
=======
### K and Z make aa good TEAM

### A3SR is better than STERN!
This is an example file.

>>>>>>> 0b1f8570f3c987d0deedc0ac78bfe9843237e22e
```

Remove the markers and retain what is needed. Save, add, commit and push changes normally after that

```
This is an example file. We are using this file to show conflicts.

### K and Z make aa good TEAM

### A3SR is better than STERN!!
```

Git++

Git LFS

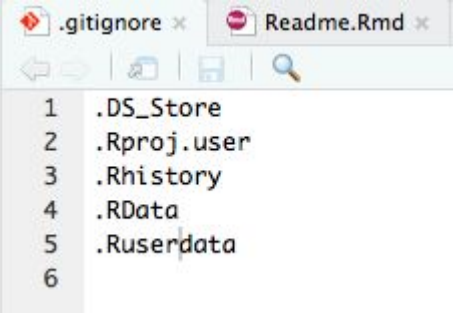
- Large File System (LFS) used for versioning files >50MB
- GitHub provides a 1GB limit for free
- Install Git LFS:
  - https://git-lfs.github.com/

- *git lfs install*
- *git lfs track "filename/type"*
- *git add .gitattributes*

- Add, Commit & push normally afterwards

.gitignore

- Tells Git what files/types of files to ignore (i.e: don't monitor changes in)
- Some examples:
  - Packages: **.7z, *.dmg, *.gz, *.iso, *.jar, *.rar, *.tar, *.zip* (good to unzip these and push the content)
  - OS generated files: **.DS_Store*
  - R: **.Rproj.user, *.Rhistory, *.Ruserdata,* ****.RData***
  - Python: **.pyo, *.pyc*
  - Backup files: **~*



Github Issues for Collaboration

priism-center / Asynch_Longitudinal Private

Unwatch 3 | Star 0 | Fork 0

Code | Issues 1 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Mid-Week Update: 01/02 #2

Edit | New issue

Open ZarniHtet13 opened this issue 21 hours ago · 9 comments

ZarniHtet13 commented 21 hours ago

Tasks Done

1. I modified Daphna's code (under Daphna's src file forZarni.R) to build our 1 data frame. It is from *Step 1* to *Step 6* of joining the two table sections starting from *Line 230*. The changes are half to do with code. I could not work to fix the *order* function in actually rearranging the time of each split. Therefore, I reorder the time by each subjectID with a dplyr *arrange* function instead. The other addition is merging duplicated rows. Please refer to the folder *Visuals* merged_datasample.png and unmerged_data_sample.png. Essentially, the BMI and MEDIA have the same timestamp (Surprise!) and there is no need to have NAs in separate columns for each row. We have to merge them.

2. Interpolation is *successfully* carried out and the data recomposed to the best of my knowledge from *Step 7* to *Step 10* starting from code *line 326*. The final output is under data/final/final_interp_data.csv.

3. There is a separate file *01_Addendum.Rmd* that deals with Singletons and Matches/Mismatches. More on Singletons in Tasks undone.

Question with Handling Singleton

My understanding of a Singleton is that there is the only 1-time value per subject ID. Then, to what time value do we extrapolate to? Each subject has differing time stamps and differing number of timestamps. What is the appropriate number of timestamps for those singletons to extrapolate to? What timestamps should we use?

Tasks to follow

1. If the singletons are not removed, the approxfun throws up. It requires that there must exist two data points for each subject. The current code run does not work. And I think I know why.

In the *01_Addendum.Rmd* file, we will see some initial exploration of singleton matches and general matches/mismatches. If we do *not* remove the Singletons, there is *no* subjectID in Media that has no corresponding subjectID in BMI. It is good because we can then use another time reference from either one data set to expand the singleton to non-singleton. However, it will *not* always be the case. Therefore, this brings back to the question above.

Assignees
marcascott
daphnaharel

Labels
Updates
good first issue

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications because you authored the thread.

3 participants

Lock conversation

Github Issues for Collaboration

ZarniHtet13 added good first issue Updates labels 21 hours ago

daphnaharel commented 7 hours ago

I'm still working through what you've done, but one request - can you save a version of the dataset that is merged and clean, but with NAs instead of the interpolated values as well.

ZarniHtet13 commented 7 hours ago

@daphnaharel
Step 6 does it. It is saved under here:
/data/processing/merged_arranged_data.csv, unless you want something else. I can generate on the fly!

daphnaharel commented 7 hours ago • edited

Thanks, yes that is what I meant. Can you move it to the final data folder? Also call it something like "merged_arranged_data_with_na"

daphnaharel closed this 7 hours ago

ZarniHtet13 commented 7 hours ago

Done :)

daphnaharel commented 7 hours ago

I don't see it. I think I need help understanding how to see changes to git hub repositories.

daphnaharel reopened this 7 hours ago

ZarniHtet13 commented 7 hours ago

@daphnaharel , my bad. I forgot to push it. If you are looking on Github, you will see it. If you are looking from your terminal you can do `git pull origin master` and that will do.

avoiding duplicate code #2

Edit New issue

Open **ybergner** opened this issue on Dec 21, 2017 · 0 comments

ybergner commented on Dec 21, 2017

Early on you create some ~30 different networks by duplicating the same 10 lines of code over and over again. Not only is this no fun, but what if you decide to change something? Also, some of the stuff created in each batch is temporary, but since you give each object a unique name, you never clean all of this stuff from memory.

Consider this alternative

```
# Function to Read in Data and Process Network Data
makeNet <- function(siteID) {
  filename <- paste0(siteID, "_Missing_BEHAVIOR.csv")
  tmpnet<-read.csv(filename, header=TRUE, row.names=1)
  tmpnet_matrix<-as.matrix(tmpnet)
  tmpnet_network<-network::network(tmpnet_matrix, directed = TRUE)
  tmpnet_attribute<-subset(full_attribute, Site_ID==siteID)
  tmpnet_network%v%"pd_behavior"<-tmpnet_attribute$PD_Days_1
  tmpnet_network%v%"coaching_behavior"<-tmpnet_attribute$Coaching_Amount_1
  tmpnet_network%v%"education"<-tmpnet_attribute$Degree
  tmpnet_network%v%"certified"<-tmpnet_attribute$Certified
  tmpnet_network%v%"experience"<-tmpnet_attribute$Child_Exp
  tmpnet_network%v%"job_title"<-tmpnet_attribute$Job_Title
  return(tmpnet_network)
}

# can make any individual network as follows
net502_network <- makeNet(502)
# net507_network <- makeNet(507)

# even better, make as many as you want from a list
allsiteIDs <- c(502, 507, 508, 509) # or add more
for (siteID in allsiteIDs) {
  netname <- paste0("net",siteID)
  assign(netname, makeNet(siteID))
}
```

Now, if you ever change your network features, you don't have to duplicate the code 30 times.

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications because you're subscribed to this repository.

1 participant

Lock conversation

GitHub Flavoured Markdown

- Make your RMarkdown files look good on GitHub
  - *.html* and *.pdf* files don't display properly on GitHub
  - Easy to publish a *.md* file directly to GitHub pages
- At the top of RMarkdown file, set
  - output: github_document
- Unfortunately Math mode doesn't work
  - Equations would have to be inserted as images
  - Use https://www.codecogs.com/latex/eqneditor.php

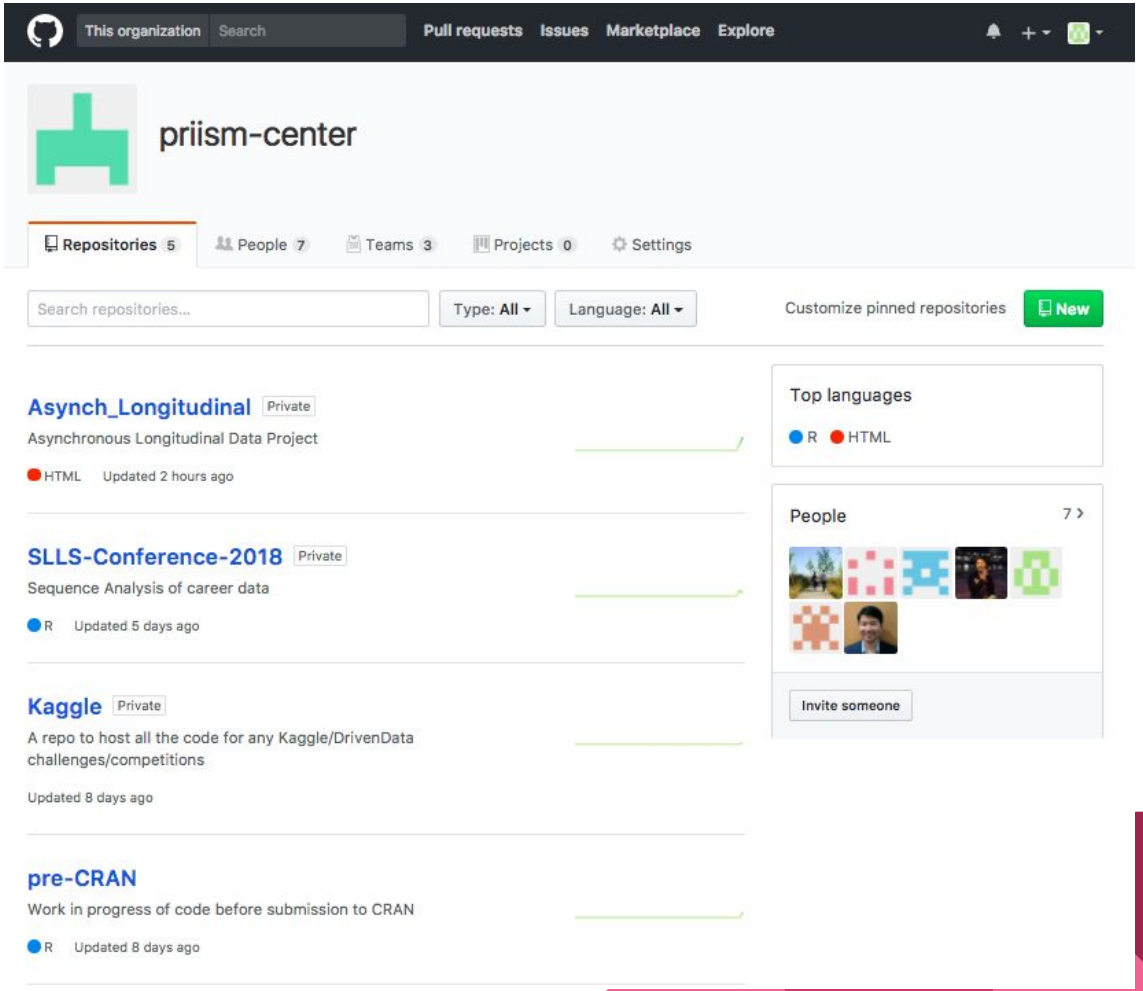
Best practices

- Create a README file that documents your repository structure
  - What are the different folders? eg: *data, src, results, etc.*
  - What does each folder have?
- Check/skim through differences before commit
- Try to follow a good Project template structure as much as possible

ZarniHtet13 Data Switch		Latest commit 9f4abee 2 hours ago
Visuals	Data is properly shaped for interpolation function	2 days ago
data	Data Switch	2 hours ago
lit	Restructure Marc's Repo	8 days ago
mnotes	Restructure Marc's Repo	8 days ago
output	Restructure Marc's Repo	8 days ago
src	Clean data pushed	6 hours ago
.gitignore	Restructure Marc's Repo	8 days ago
Asynch_Longitudinal.Rproj	Restructure Marc's Repo	8 days ago
README.md	Initial commit	9 days ago

PRIISM-center

- Private repos for student projects and student-A3SR faculty collaborations
  - Data and code can be kept private to the team
- Easy to publish student work in the future
- We have a repo with tutorials/guides on various topics from last 3 semesters



Thank you!