

Git and GitHub for R

Armin Hatefi

Department of Mathematics and Statistics, Memorial University

April 4, 2022

Motivation

- We are working on far more collaborative projects than before – sharing code and writing documents.
- We require to share and edit documents and codes at the same time.
- How to share code and make sure we are working on the same version?
- Emailing versions, using Dropbox and Google Drive? Then inefficient, prone to errors and painful ...
- With complex code, we need to have identical folder structures on each other's computers, work at the same time on the files, catch up easily changes/ collaborators ...

Let's avoid inefficient methods



1 Introduction

*3

In general, probability-density function (pdf) of mixture model with m -component are defined as follows

$$g(x; \psi) = \sum_{j=1}^M \pi_j f_j(x; \xi_j, \theta_j), \quad \sum_{j=1}^M \pi_j = 1, \quad x \in \mathbb{R}^p, \quad (1)$$

*4

It is usually assumed that we need to estimate the unknown mixture proportions $\pi_j \geq 0$ and the unknown pdf's f_j

are usually assumed to belong to parametric family $\mathcal{F} = \{f(\cdot; \theta), \theta \in \Theta\}$ indexed by an

Euclidean parameter ξ , so that the pdf g is as follows. If finite mixture model is given by

$$g(x; \psi) = \sum_{j=1}^M \pi_j f(x; \xi_j, \theta_j), \quad (2)$$

where $\psi = \{(\pi_j, \xi_j, \theta_j) : j=1, \dots, M\}$. The choice of a parametric family \mathcal{F} may be difficult for

non-parametric applied to mixture models. It is also important to note that the model (2)

will be more flexible if M is considered to be an unknown number; m would be







estimated in that case. Another way to make the model more flexible is to avoid parametric

assumption on \mathcal{F} if the number of components is specified but that little is known about







subpopulations. For example, we can consider $f_j \in \mathcal{F} = \{\text{continuous pdf on } \mathbb{R}^p\}$ for

Let's properly keep track of everything
























Commits on Mar 18, 2022

- Merge branch 'main' of https://github.com/arminhatefi/Elsayed_MUN
 ElsayedGhanem committed 23 hours ago  5b8b04c 
- Update Graph.R
 ElsayedGhanem committed 23 hours ago  9f4167e 

Commits on Mar 15, 2022

- update2
 arminhatefi committed 4 days ago  acce98e 
- Update .DS_Store
 arminhatefi committed 5 days ago  507df5e 

Commits on Mar 13, 2022

- march13_2
 arminhatefi committed 6 days ago  6dd98ce 
- 51100pdf
 arminhatefi committed 6 days ago   9147748 
- 151pdf
 arminhatefi committed 6 days ago   583ee95 
- d2
 arminhatefi committed 6 days ago  0eecc90 
- Merge branch 'main' of https://github.com/arminhatefi/Elsayed_MUN
 arminhatefi committed 6 days ago  437b6fd 
- de1
 arminhatefi committed 6 days ago  2624710 
- Update .DS_Store
 arminhatefi committed 6 days ago  e14a2b4 

Git and GitHub

- Git is open source software for version control.
- Using Git, you can do things like see all previous versions of code you have ever created in a project.
- GitHub is the most popular service (others include GitLab) for collaborating on code using Git.
- It is possible to use Git without using GitHub, though most people combine the two.

Getting started ...

- Before you start using Git and GitHub, you need to set up your computer and install a few useful programs ...
- This is a one-time setup and once done, you will be able to easily create new projects and start collaboration.
- I found these instructions from different resources (particularly for Windows OS as I was not able to test); however, you should be able to troubleshooting easily ...
- A GitHub account
- Git on your computer
- GitHub Desktop
- R, RStudio and RMarkdown ...

Get a GitHub account

- Sign up for GitHub: <https://github.com/>
- Keep you username and password, we will need them later again.
- Type *git* on the command line. If not, download and install Git application on your machine from: <https://git-scm.com/downloads>
- Once installed, type the following on the command line.

```
$ git --version
```

Configure the GitHub Desktop

- Download and install GitHub Desktop Application from:
<https://desktop.github.com/>
- To authenticate an account, open the GitHub Desktop Application:
- **File > Options > Accounts > To the right of "GitHub.com," click Sign in .**
- In the Sign in pane, click "Sign in using your browser".
- For the instructions see:

[Installing and authenticating GitHub Desktop](#)

Clone and commit with GitHub Desktop

- Create a new repository **testRepo** on GitHub website.
- Clone **testRepo** repository via GitHub Desktop.
- Commit **README.md** on "GitHub.com" and pull it by GitHub Desktop.
- Create an R/Rmarkdown code in your local **testRepo**. Commit and Push by GitHub Desktop.
- Invite a collaborator to your **testRepo**:
On GitHub.com > testRepo > Settings > Collaborators (you need to add the username of your collaborator e.g. < *arminhatefi* >)
- Your **testRepo** repository is ready for collaboration ...

Basic Commands

- `$ git init` //Initialize local git repository
- `$ git add < file >` //Add file(s)
- `$ git status` //Check status of working Tree
- `$ git commit` //Commit changes
- `$ git push` //Push to remote repository
- `$ git pull` //Pull latest from remote repository
- `$ git clone` //Clone repository into a new directory

Clone and commit via cmd

- Create a new repo **testCMD** (with README.md) on GitHub.com
- Copy the **testCMD** HTTPS and clone into GitHub directory.

```
$ cd ~/github
```

```
$ git clone https://github.com/arminhatefi/testCMD.git
```

```
$ cd testCMD
```

```
$ vim README.md
```

- Create **carDemo.rmd** in your directory and knit it.

```
$ git add carDemo.rmd
```

```
$ git status
```

```
$ git commit -a -m "Added carDemo.rmd"
```

```
$ git push
```

Clone and commit via cmd

- Update .rmd file on GitHub.com and pull it.
Rscript Head(pressure)
\$ git pull
- Your repository is ready for any drag, drop and commit ...

Clone and commit with RStudio

- The first step is to install Git. See [Chapter 6 of Happy Git with R](#) shows the process for Mac, Windows, and Linux users.
- Create `testRepo1` repository on GitHub.com.
- Clone `testRepo1` either by cmd or Desktop into your local directory `/GitHub`.
- Open a new RStudio session in your local repository `testRepo1`.
- I am able to verify that I had Git installed using the **terminal tab in RStudio**.

`$ which git`

`$ git --version`

Clone and commit with RStudio

- Next we need to edit configuration/configure Git. See [Chapter 7 of Happy Git with R](#). **Type in R console:**
`library(usethis)`
`edit_git_config()`
- This leads you to your gitconfig file. Add your name and email and save this.
- Then create the new R project `arminRproject` via **New Directory** and save it in directory `testRepo1`.
- RStudio open a new session/ in your R project, **type in R console:**
`library(usethis)`
`use_git()`
- Now you should have a git repository (since the git tab appears in the top right panel).

Clone and commit with RStudio

- Now create as Rmarkdown file `testRmakdown` in your project. You will see `testRmakdown.rmd` has not been committed yet.
- The next step is to commit the changes and see the history of commits.
- Next we need to connect RStudio and GitHub. To do that we need to create a Personal Access Token (PAT) on GitHub.
- Note that here some inconsistencies can happen because of OS, or some may not be able to validate the token.
- **Type in the R console** of your R project the following:
`library(usethis)`
`create_github_token()`
- This leads you to **New personal access token** page on GitHub. You need a description for example (R_GitHub_PAT).

Clone and commit with RStudio

- This leads you to **New personal access token** page on GitHub. You need a description for example (R_GitHub_PAT).
- Now generate the token and then you have to copy the token (be careful the token is generated once. You will never see that again).
- In **R console of your R project**, type:
`library(gitcreds)`
`gitcreds_set()`
- Enter your Personal Access Token.
- Once you have done all of this, you have connected RStudio to GitHub!
- For any problem, see <https://usethis.r-lib.org/reference/github-token.html>.

Clone and commit with RStudio

- If you have not linked your R project with `testRepo1` (which is not the case for us), you can also type in your R console of your R project:

```
library(usethis)  
use_github()
```

- Now look at GitHub.com, `testRepo1` repository has been updated.

GitHub First

- The most straightforward way to use RStudio and GitHub together is to create a repo on GitHub first. `testRepo2`
- Then when you start a new project in RStudio, use the **version control** option, enter your repo URL. You can create wherever you want (e.g., Desktop) but let's create again in `~/GitHub`
- The `testRepo2` is ready, create a new R file/ Rmarkdown and commit and then push the changes.
- You can pull changes too!

Additional Resources

- For more information on Git with R, see <https://happygitwithr.com/>
- For an introduction to vim editor, see <https://vim.rtorr.com/>