

# Stat 610 Homework 6

Due Wednesday, October 30

## Assignment

In this assignment, you'll practice code profiling and using git.

You'll start with the code you wrote in the testing lab (<https://jfukuyama.github.io/teaching/stat610/assignments/lab3.pdf>) try out some modifications in an attempt to make it faster.

- Make a github repository, and clone a copy to your computer following the instructions on github. Once this is done, if you type `git status` in the terminal, you should get the output

```
On branch master
```

```
No commits yet
```

```
nothing to commit (create/copy files and use "git add" to track)
```

- Add an initial R file called `llr_functions.R`, testing file called `test_llr.R`, and a file called `benchmark_llr.R`.

Once you've created these files, but before you've added them to the staging area or committed them, if you type `git status`, you should see something like

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

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

```
benchmark_llr.R
```

```
llr_functions.R
```

```
test_llr.R
```

```
nothing added to commit but untracked files present (use "git add" to track)
```

Once you're at this stage, add the three files to the staging area using `git add benchmark_llr.R`, `git add llr_functions.R`, `git add test_llr.R`.

At this point, if you type `git status`, you should see something like

```
On branch master
```

```
No commits yet
```

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: benchmark\_llr.R

new file: llr\_functions.R

new file: test\_llr.R

Finally, commit these files by typing `git commit -m 'initial commit'`. Feel free to make a better commit message.

Once you have done that, typing `git status` should give you output

On branch master

nothing to commit, working tree clean

- Fill out the `llr_functions.R` and `test_llr.R` files like we did in lab, so that you have a working implementation of `llr` and tests of the functions.

Add these files to the staging area using `git add` and then commit them with `git commit -m 'your commit message'`.

- Add code to the `benchmark_llr.R` file that computes how long your `llr` function takes to run and prints it out (hint: use the function `cat` and one of `bench::mark` or `microbenchmark::microbenchmark`).

Add these files to the staging area using `git add` and then commit them with `git commit -m 'your commit message'`.

**Question 1:** What does your commit history look like now? What branches do you have, and what commits are they pointing to? Where does HEAD point?

- Make a new branch called `speed-test-1` by typing `git branch speed-test-1`.

You can check that you made the branch by typing `git branch`, which should give you output

```
* master
  speed-test-1
```

which indicates that you have two branches, one called `master` and one called `speed-test-1`, and that `master` is *checked out*, i.e., HEAD points to `master`.

- Check out `speed-test-1` by typing `git checkout speed-test-1`.

**Question 2:** What changed when you checked out `speed-test-1`? Where does HEAD point now? If you make changes and commit them, where will the `master` and `speed-test-1` branches point?

- *Speeding up llr: take 1:* In the code we wrote in lab, we are using standard matrix multiplication to multiply a diagonal matrix by a dense matrix. If  $D$  is a diagonal matrix and  $X$  is any matrix,  $DX$  results in the  $i$ th row of  $X$  being multiplied by  $D_{ii}$ , and so there are potentially

faster ways of computing  $DX$  than the standard matrix multiply. We will try some and see if they actually are faster.

Change the line of code

```
Wz = make_weight_matrix(z, x, omega)
```

so that  $Wz$  is a vector of weights instead of a matrix with weights on the diagonal.

Then change the line of code

```
f_hat = c(1, z) %>% solve(t(X) %>% Wz %>% X) %>% t(X) %>% Wz %>% y
```

so that you use the `apply` function in place of `Wz %>% X` and `Wz %>% y`.

**Question 2:** What function did you use? Why is it equivalent to the matrix multiply?

Add your changes to the staging area using `git add llr.R` and then commit them with `git commit -m 'your commit message'`.

- Use your `benchmark_llr.R` script to check how fast the new version of `llr.R` is. Switch between the master and `speed-test-1` branches by using `git checkout master` and `git checkout speed-test-1` and run `Rscript benchmark_llr.R`.

**Question 3:** Which version of the function is faster?

- *Speeding up llr: take 2.* Now we'll try another way of speeding up `llr`. Create a new branch for the new version called `speed-test-2` and switch to that branch using

```
git checkout speed-test-1
```

```
git branch speed-test-2
```

```
git checkout speed-test-2
```

Change the line of code

```
f_hat = c(1, z) %>% solve(t(X) %>% Wz %>% X) %>% t(X) %>% Wz %>% y
```

so that you use the `sweep` function instead of `Wz %>% X` and use a vectorized function instead of `Wz %>% y`.

**Question 4:** What function did you use? Why is it equivalent to the matrix multiplication?

Add your changes to the staging area using `git add llr.R` and then commit them with `git commit -m 'your commit message'`.

- Use your `benchmark_llr.R` script to check how fast the new version of `llr.R` is. Switch between the master, `speed-test-1`, and `speed-test-2` branches by using `git checkout <branch-name>` and running `Rscript benchmark_llr.R`.

**Question 5:** Which version of the function is the fastest? Do you get a substantial speedup?

- **Question 6:** Run

```
git log --graph --branches
```

What output do you get? What does it tell you about the commit history?

## **Submission parameters**

Submit a pdf with the answers to the six bold-faced questions and a link to your github repository.

Your github repository should have branches `master`, `speed-test-1`, and `speed-test-2`. If someone clones your repository, when they check out the different branches they should see the different implementations of the `llr` function. If they run `Rscript benchmark_llr.R`, they should get timing information about the version of the function in the branch they've checked out.