1. The commit history is made up of three different commits. The first commit contains the three blank .R files. The second commit contains llr\_functions.R and test\_llr.R with the code copied from lab. The third commit contains the updated benchmark\_llr.R file, which measures the speed of the llr function. There is currently one branch which points to the most recent commit, and HEAD points to this branch.
2. The checkout function changed the branch that HEAD points to from the master branch to the speed-test-1 branch. If a new commit is made, the speed-test-1 branch will point to this new commit (since speed-test-1 is the branch that HEAD points to) while the master branch will remain pointing at the commit containing the updated benchmark\_llr.R file.
3. To turn the diagonal matrix into a vector containing the elements on the diagonal, I used the diag() function as this is its purpose. The homework says that 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*. With the diagonal matrix now a vector, we want to multiply each *i*th row of X by the *i*th element in this new vector. This can simply be accomplished by changing “%\*%” to “\*” for every DX in the code, as R multiples matrices element-wise, rather than the traditional linear algebraic way.
4. The following output comes from running the benchmark\_llr.R file on the commit the master branch points to (with the compute\_f\_hat function written in lab):

Text

Description automatically generated

The following output comes from running the benchmark\_llr.R file on the commit the speed-test-1 branch points to (with the compute\_f\_hat function edited for Question #3):

Text

Description automatically generated

As demonstrated in the output, the function edited for Question 3 has faster mean and median run times (in nanoseconds).

1. The code I used is sweep(x = X, MARGIN = 1, STATS = Wz, FUN = "\*"). The homework says that 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*. With the diagonal matrix now a vector, we want to multiply each *i*th row of X by the *i*th element in this new vector. My sweep function accomplishes this because it takes the matrix X (x = X) and multiplies (FUN = "\*") each row (MARGIN = 1) by its corresponding index in Wz (STATS = Wz). Wz %\*% y can be replaced by Wz \* y as R multiples matrices elementwise, rather than the traditional linear algebraic way.
2. The following output comes from running the benchmark\_llr.R file on the commit the speed-test-2 branch points to (with the compute\_f\_hat function edited for Question #5):

Text

Description automatically generated

Information on the speed of the other two version of the compute\_f\_hat function is shown above. In terms of median runtime, the version of the function using sweep() is equivalent to the function that speed-test-1 points to. However, the mean time for the function that speed-test-1 points to is faster, and the sweep() version of the function actually has the greatest mean runtime and maximum runtime of any of the three versions. However, many of these differences are small (the median runtimes range by .5 nanoseconds) and thus there is not a substantial speedup.

1. Running git log --graph –branches produces the following:

Graphical user interface, text, application, email

Description automatically generated

The above output shows the repository has the intended number of commits and branches. This command also offers details on the author and date of commits. As expected, HEAD points to the branch that points to the most recent commit (speed-test-2). The other branches point to the commits to which they were assigned.

**Github link:** https://github.com/ethanphilipweiland/STAT-S-610-Homework-5/