STAT 209 Git Mini-Lab

Much of this tutorial is borrowed from Jenny Bryan at the University of British Columbia.

Installation of `git`

If you are using your own local installation of RStudio, you may need to install the `git` program. This should not be necessary if you are on Linux, or a Mac running OS X 10.9 or later, as `git` is preinstalled. If you are on Windows or an earlier version of OS X… just use the RStudio server for now at least.

Configuring RStudio to work with `git`

Exercise 1 From RStudio, go to "Tools > Global options", select Git/SVN, and make sure that "Enable version control interface …" is checked.

Creating a GitHub account

You can use `git` to track changes in your own work without using GitHub. But if you want to use `git` collaboratively, and in particular, if you want to use it to turn in work for this class, then you need a copy of your repo to exist "in the cloud". GitHub is by far the most popular choice for this.

Exercise 2 Visit http://www.github.com (http://www.github.com) and, if you do not already have an account, create one.

Linking your RStudio and `git` accounts

Git associates your name and e-mail address with each commit, which helps when multiple people collaborate on a project. To configure your name and e-mail address in `git`, go to Tools > Shell…, and type the following (substituting your name and the email address you used to register a GitHub account)

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

Creating a new project

You can either create a new project first in RStudio and then "upload" it to GitHub, or you can create a project in GitHub and "download" it to RStudio. The latter is a bit easier.

Exercise 3 Log in to your GitHub account, and select either "Start a Project" or "New Repository" (or click the "+" button next to your account dropdown menu in the upper right). Give your new repo a name (no spaces!), such as `stat209-git-mini-lab`. Write a short description in the box. With a free GitHub account, your repo will need to be public, though later when you are committing graded work for the course, I will provide access to private repositories. Check "Initialize this repository with a README", and under the `.gitignore` dropdown menu, select "R" (this tells

GitHub not to track changes to certain auto-generated files that are intended to store temporary data). Click “Create Repository”

Downloading your new repo to the RStudio server

You now have a minimal project with one file: the `README.md` documentation (this file is a Markdown file and you can use the same syntax to format it that you use to format RMarkdown documents, minus the code chunks).

But if you actually want to do anything with this project you need a copy of the repo on the machine where you use RStudio.

Exercise 4

From RStudio, select File > New Project… (or use the Project dropdown menu in the upper right). Save any unsaved changes in your workspace that you want to keep, and then choose “Version Control”, followed by “Git”. Copy the URL of the repo you created on GitHub (it should look something like `https://github.com/crdawson/stat209-git-mini-lab.git`), and provide a name for the directory that will be created in your RStudio account to store your project files (avoid spaces and special characters other than `-` and `_`!). This should create a directory, and download the `README.md` file.

Making local changes and committing them

Exercise 5

Open the `README.md` file in RStudio, add a short description of the project, and save the file. Now select the “Git” tab in the upper right pane. You should see `README.md` appear in the list with an “M” next to it (this tells us that the file has been `M`odified since the last commit. To tell git that we want to commit these changes, check the box next to the file to put it in the “staging” area (the changes are not committed yet, but they are ready to be committed when you give the go ahead). To see the differences between the archived and current version of the file, click “Diff”. When you’re ready, click “Commit”. You will need to add a commit message. Now the current state of your project is registered in your *local* repo. (They are not yet uploaded to GitHub, but you have created a snapshot of your project that you can revert to if needed)

You should commit changes often; every time you finish some meaningful chunk of work

“Pushing” the changes to GitHub

If you are collaborating on a project, your collaborators can only see changes that you have “uploaded” or `push`ed to the shared, remote copy of the repo.

Caution: Before you `push`, you should *always* `pull`! This way, if someone else has made changes since your last push, you will be informed of any conflicts that need to be resolved. If you try to push while your local repo is out of date, you will get an error which can be non-trivial to resolve.

Exercise 6

Click the “Pull” button in the Git tab in RStudio (the blue down arrow). Nothing will happen, except that you will get a message saying that your local repo is already

up-to-date, but it is good to establish this habit. Now you are ready to “Push” (the green up arrow).

Adding an RMarkdown document to your project

Exercise 7 Create a new `.Rmd` file, change the title and author, save it, and Knit to `.html`. Add the new files (as well as the auto-generated `figures` sub-directory) to the staging area, then `commit`, `pull` and `push`.

Exercise 8 Edit the plot command in the RMarkdown document to use `ggplot()` instead of the base R `plot()` function. Knit, stage, commit, pull and push. (In reality you will not necessarily push every time you commit, and you will probably not commit every time you change a single line, but we are developing muscle memory here)

Reverting undesired changes

Exercise 9 Introduce a typo or some other undesirable change to your Markdown file. In the “Git” tab, do a Diff on the file, and click “Revert”, to restore the file to the state it was in the last time you committed.

Deleting a file

Exercise 10 Create a new Rscript called `scratchpad.R`. Add a comment, and save it. Stage the file, commit, pull and push. Now suppose you didn’t actually want to keep that file. Delete it by checking the box next to the file in the “Files” tab and clicking “Delete”. In the “Git” tab, you should now see a red “D” for “Deleted”. The next time you commit, the latest snapshot in the repo will have that file removed. (Since it had previously been committed, however, it can still be recovered from an earlier snapshot if needed)

Reviewing your recent work

Exercise 11 Click the “clock” icon in the Git tab. This is the git “History” command (not to be confused with the History tab in RStudio), and will show you a record of all the commits you’ve made so far.