**Setup and Workflow for 2016 Hake assessment document using GitHub**

**Starting out:**

1. Make sure you have GitHub for windows installed and set up for your account (<https://desktop.github.com/>)
2. Under the windows->Start->programs->GitHub Inc. there should be a **Git Shell.** This is what I will use, and explain steps with.
3. Log into your GitHub account and Fork the hake assessment. Go to <https://github.com/cgrandin/hake-assessment> and press FORK on the top right
4. Now go to your GitHub repositories and you’ll see hake-assessment there. This is your copy of the repository. Changing it will not change my copy, until I merge yours in.
5. I have a directory called **C:\github** that I created to hold all of my git repositories so they are all in place. Create that on your machine, and then open the **Git Shell** and navigate to that directory.
6. Type: **git clone** [**https://github.com/YOUR-GITHUB-ID/hake-assessment**](https://github.com/YOUR-GITHUB-ID/hake-assessment) in the Git Shell, where **YOUR-GITHUB-ID** is your login ID for the GitHub site. You can just copy the URL from the browser address bar when on your own hake-assessment repository site (NOT mine).
7. Once cloned, you are ready to build the document and/or make changes. If you want to build the document yourself, you need to have MikTex and R version 3.2.2 or later, and the knitr and xtable packages. Also you need to create a **models** directory and drop one of the models on the Hake JTC Google Drive that Aaron has placed. Just start with **00\_2015hake\_basePreSRG**. The **models** directory must reside in the **C:\github\hake-assessment\** directory.
8. Put Rscript location on the PATH. Mine is here: **C:\R\R-3.2.2\bin\x64\Rscript.exe**. Rscript is used to knit the document from the command line.
9. Open Rgui and set the working directory to the **C:\github\hake-assessment\doc\r** directory and type **source(“all.r”)**. Answer **y** to the questions and the model will be loaded, calculations done, and the forecasting models defined in **all.**r will be run and loaded. Once the routine finishes, close Rgui, choosing to save the environment when it asks, or as Andy suggests, type **save.image()**. This wil create an .RData file in **C:\github\hake-assessment\doc\r**. That is the source of everything for the document.
10. In **C:\github\hake-assessment\doc**, run buildtex.bat. This will do everything necessary to build the document. It will probably fail, when it does look at the two output logs **knitrOutput.log** and **latexOutput.log**. They will hold the error messages. Cleantex.bat cleans up all the junk from the build so you can start fresh.

**Once you are set up:**

1. Make some changes to a section and save the file. Start with something trivial for the first try.
2. Open the Git Shell and change to the **C:\github\hake-assessment** directory
3. Type: **git status**. It will say that one file has changed, it will be the one you just edited.
4. Type: **git diff**. It will show you exactly what you changed. This is what you will commit if you choose to do so now.
5. Type: **git commit –a –m “Your detailed commit message”**. The message should explain simply but well what you have done.
6. Type: **git push**. This will send your changes to your GitHub repository.
7. I will be constantly watching for these and merge the changes into the master.

**When you want to get the most up-to-date version and merge with yours:**

1. Add me as a remote. When in Git Shell in the hake-assessment directory, type **git remote –v**. You will see two entries for **origin**, fetch and push. This is your remote GitHub repo. To add mine, type:   
   **git remote add cgrandin** [**https://github.com/cgrandin/hake-assessment**](https://github.com/cgrandin/hake-assessment). Now check your remotes again using **git remote –v**, there should now be four entries, two for **origin** and two for **cgrandin**.
2. Make sure you have no changes, type: **git status.** If it tells you it is clean and up-to-date, you can fetch my version. Type: **git fetch cgrandin**. This brings the changes down but doesn’t merge anything.
3. Look at the differences between your version and mine. Type: **git diff cgrandin**. Once you are happy with what you are about to merge, type: **git merge cgrandin/master**. If it says something along the lines of fast-forward merge success then you have successfully merged the changes and you can type: **git push** to make your GitHub repository up-to-date. This is the ideal situation with no conflicts.
4. If there was a merge conflict, it will tell you which file has the conflict. Open that file and do a text search for a bunch of less than (<) signs. Git inserts these as headers in this format:  
   **<<<<<<<<<<**   
   Code in your repo that conflicts with mine  
   ….  
   **==========**  
   Code in my repo that conflicts with yours  
   ….  
   **>>>>>>>>>> cgrandin/master**  
   It’s up to you to figure out what should stay, what should go, or if they should be combined somehow. You need to manually delete the header stuff as well. Once you have sorted out the conflict, you need to issue another commit and push: **git commit –a –m “Fixed conflict with…..”**. Then, **git push**. Note that Git only looks at line numbers for conflicts, so if two people add something to the same file at the same time, there will likely be a conflict, especially if it is near the end of the file.
5. Check the network graph on GitHub. Go to your repository webpage, click **Graphs** tab, then **Network**. This graph can be difficult to read when there are a lot of commits, but the day of the month is on top, and the latest commit should be the furthest right. You can hover over each of the commits to see what each of us have recently done. Since I am holding the master, I should have the latest commit much of the time.

--------------------------------------------------------------------------------------------------------------------

When time allows, a more in-depth training repo I made up is here: <https://github.com/cgrandin/git-workshop>.

You can fork and clone that if you want, there is a nice power point file in it called GIT.pptx which has more details.

Also, I created a basic template project to outline the use of latex and knitr together: <https://github.com/cgrandin/latex-knitr>

This repo is what I started with when making the hake-assessment repo. Of course you can fork and clone that as well if you want.