Materials

For all of the following, you will need:

- Download the latest R https://www.r-project.org/
- A fairly recent version of RStudio
  (check this http://www.r-bloggers.com/setting-up-new-r-notebook/)
- A GitHub account
- git installed on your computer

Step 1: Create your own fork of the Ewen2015/ChinaMeteorology repository

Make sure you are logged into your GitHub account.

Go to https://github.com/Ewen2015/ChinaMeteorology.

Near the top right, there will be a "Fork" button. Click it. This will create your own fork of the repository.

Having your own fork lets you change the code however you want without it affecting the code in the main repository. When you are ready for your new code to be incorporated back into the main repository, you can issue a pull request and one of the contributors to the main repository can review your code and merge it in.

Step 2: Clone your fork of the repository to your computer

Open a bash command line on your computer. If you have a Mac, you can do this by opening "Terminal".

Use the `cd` (change directory) command to move into the directory where you'd like to save this repository. For example, if I wanted to save it on my Desktop, which is a subdirectory of my home directory, I'd run:

```
cd Desktop
```

Check that you're in the directory you'd like to be in with the `pwd` command (print working directory):

```
pwd
```

Clone your fork of the repository to this directory on your local computer (replace [your github handle] with your actual handle; for example, I would replace with geanders):

```
git clone git@github.com:[your github handle]/rnoaa.git
```

Also, add the original repository as your "upstream" branch, so you'll be able to pull in changes from there later in the hackathon. Use the following code to do that:

```
git remote add upstream git@github.com:ropenscilabs/rnoaa.git
```

Once you have your fork of the repository on your computer, you can use RStudio to make changes to it locally using something like RStudio. Then you can push those changes back up to your online GitHub fork of the repository.

Step 3: Open your local clone of the repository in RStudio

You will now have a new directory called "ChinaMeteorology" on your local computer. Inside it will be a file called "ChinaMeteorology.Rproj". Open that file with RStudio.

The .Rproj file saves this repository as an R project. When you open that file, it opens the whole project. This does some very convenient things, like set your working directory to the project's directory, show you the files in the project, and give you some nice extra functions, like a git box (provided you've set up git to run with RStudio).

Step 4: Open one of the user test case .Rmd files

Find the "Files" tab in your RStudio window (typically one of the tabs in the bottom right pane).

Click on the "inst" directory in "Files".

Click on the "user_test_cases" directory in "inst".

Open the .Rmd file for the test case you are working on.

These user test case files are in RMarkdown. We have some RMarkdown cheatsheets available if you'd like to use one to help with the syntax. Each team will be working to try to achieve some of the tasks in a user test case and will document what works and what doesn't (and what questions come up) along the way. We will be converting these into project-oriented vignettes to include our fork of the `rnoaa` package, either on the second day of the hackathon or after the hackathon.

Step 5: Add yourself as an author to the .Rmd file

At the top of the .Rmd file, there's some information like title and authors. Add your name and an abbreviation that no one has used yet to the "authors" section. For example, I would add: Enqun Wang (EW).

As you edit the file, if you want to add comments on things like data sources, tips for others working on the test case, etc., start those off with your abbreviation.

Step 6: Add comments and code

Try to go through your parts of the user test case. If you can get things to work, add code in code chunks, like:

```
```{r}
miami <- ghcnd_search(…
[more code …]
```
```

If you want to show code that does not work, or that takes a long time to run, you can use the `eval = FALSE` (do not evaluate the code) option for the code chunk:

```
```{r eval = FALSE}
miami <- ghcnd_search(…
[more code …]
```
```

Try running "Knit HTML" (there's a button at the top of the RMarkdown file in RStudio) as you go, to make sure the document is compiling.

Use the RMarkdown cheatsheet to help remember RMarkdown syntax.

Step 7: Commit often

Save changes as you go by commiting them in RStudio.

Go to the "Git" tab in RStudio (usually in the upper right pane). Any files that you have changed but not yet committed will be listed here.

Click the "Commit" button. This will open a new window.

Check all the files you want to commit. On the right of the window, add a one-line commit message describing what you've changed. If you need to write a longer commit message, skip a line under the one-line short message and then write more describing the commit. You have to include a commit message to make a commit.

Press the "Commit" button under the "Commit message" pane.

Step 8: Occasionally push to GitHub

When you are ready to push your local changes to GitHub, push the green up arrow in either the git pane of your main RStudio window or in the commit window that pops up when you commit changes.

Pushing your changes takes everything that you've changed and committed on your computer and pushes it back up to your fork of the repository on GitHub. You can go

online once you've pushed changes to see the up-to-date version in your online account if you want.

Step 9: Occasionally create a pull request to the Ewen2015/ChinaMeteorology repository

When you are ready for me to add your changes to the .Rmd back into the main repository, you can issue a pull request.

Go to your fork of the repository on GitHub. This should be at:
https://github.com/[your github handle]/ChinaMeteorology

There is a green button that says "New pull request".

This will take you to a page to submit a new pull request. Add a message describing the changes you've made and then submit the request.

Let me know that you've made a pull request so I can look over your changes and pull it into the main repository.

Step 10: If you ever need to, pull and merge changes in the main repository into your fork

Open a bash shell (e.g., Terminal in Mac), use `cd` to change into your local version of the repository, and then use the following code to bring in any changes in the main repository:

```
git fetch upstream
git merge upstream/master
```

If you have merge conflicts (you'll get a message), let me know, and I'll help you out.

Once you've merged the changes locally, go into RStudio and make sure you push to your GitHub fork, to get that up-to-date.