**CMU MSP 36601 Working in R Sep. 13, 2017**

1) **Big ideas**
   a) ***The golden rule***: Whatever you do, you will need to do it again, due to:
      i) Corrected data
      ii) New data
      iii) Coding error
      iv) Misunderstanding about the data
   b) ***The silver rule***: Whatever you do, someone (possibly "future you") will need to modify your code and run the modified version.
      i) Document the project's goals, assumptions, and approach
      ii) Document requirements (file structure, required files, OS limitations)
      iii) Document each function's input, output, and approach (if needed)
      iv) Avoid spaghetti code!
      v) Name your constants
      vi) Don't repeat yourself
      vii) Consider extreme cases
      viii) Use exception handling
      ix) Do not use absolute file addresses except URLs or for shared company file systems
      x) Consider clear error messages for inappropriate input
      xi) Avoid packages with few benefits
2) **A suggested R workflow for small to large projects**
   a) In GitHub create a new repository
      i) Consider carefully if it should be public or private
      ii) Initialize with a README
      iii) Add the provided R-specific .gitignore
      iv) Consider adding a license
   b) In GitHub, use "upload files" to load the initial files to the repository. Commit the change.
   c) In R, create a new project using "Version Control" and "Git". The dialog box shows:
      i) the repository URL (full, e.g., https://github.com/hseltman/myProject)
      ii) the project's directory name (defaults to the last part of the URL)
      iii) the directory **under which** a new project directory will be created
   d) Work in RStudio
      i) At appropriate times, use the Git/Commit button to pull up the RStudio Git GUI
         (1) In "Changes" mode, you can:
            (a) choose a file and use the "Ignore" button to add a file to .gitignore,
            (b) choose a file, stage it, and view changes
            (c) use the "revert" button on a file to irretrievably go back to the last snapshot
            (d) add a commit message and "commit" the current snapshot
            (e) "push" or "pull" changes from the repository
         (2) In "History" mode you can view all the commits and commit messages
      ii) If you want to use branching or other features not in the GUI, use Tools/Shell from the menu to get a shell, and then use ordinary git commands. It is best to close the RStudio Git GUI while working in the shell.

**3) Choices and specific goals**

a) Choices: Script file vs. Markdown (Notebook)

b) Specific goals: 1) follow both the golden and silver rules, so that your code file is readable and changeable, **and** 2) be sure your code just needs to be run (`source("myCode.R")` or click the "knit" button) to get readable output.

**4) R Script files**

a) Should be called ".R". May include `source("myCode1.R")`, but be sure to include these in "requirements".

b) Note that a line containing only the name of a variable will print that variable at the console, but does not print anything when `source()`is used. Use `print()` or `cat()` as appropriate.

i) Generally, if `print()` shows "[1]", use `cat()`.

ii) E.g., `cat("The p-value is", m$pvalue, "\n")` or `cat("p=", m$pvalue, "\n", sep="")`

iii) `paste()` is often helpful:

(1) `cat(paste("age =", ages, "-> yhat =", yhat, "\n"))` gives something like:

```
age = 22 -> yhat = 12.5
 age = 23 -> yhat = 13.7
 age = 24 -> yhat = 15.9
```

(2) `cat(paste("age =", ages, "-> yhat =", yhat, "\n"), sep="")` gives:

```
age = 22 -> yhat = 12.5
age = 23 -> yhat = 13.7
age = 24 -> yhat = 15.9
```

(3) `cat("Bad variables:", paste(badVars, collapse=", "))` gives:

```
Bad variables: a, b, c
```

iv) Consider surrounding all plot commands with, e.g., `png("myFile.png");;dev.off()`.

v) ♦**Important**: Before sharing, select "clear objects from workspace", "clear plots", "clear console", and use the "Source" button for a full test of your code.

vi) Running your code as a batch file (no RStudio involved)

(1) At a command prompt use:
`R CMD BATCH --no-save --no-restore --quiet myInFileName.R`
myInFileName.Rout is created to hold the results

(2) Note that you can make a Windows shortcut with this same command

(3) Or on a Mac:

(a) Create a plain text batch file with

```
#!/bin/bash
cd myDirectory
R CMD BATCH --no-save --no-restore --quiet myInFileName.R
exit
```

(b) In the terminal use `chmod u+x myBatchFile` to make the batch file executable.

## 5) R Markdown (new Notebook mode)

a) R Markdown ***combines some nice text formatting with R code and interspersed R output***, including both text and graphics. (You can suppress the code, if you like.) This is great for homework in some classes. For formal reports, you will need to suppress the code (or at least most of it) and add extra output to identify what text output you are generating. Or, depending on the requirements of the report, you may need or want to write the actual report in Word or LaTeX.

b) In R, use the menu item ***File / New File / R Notebook***. Enter your title in the document and erase everything below the header.

c) ***Break your code into logical "chunks"***. Use the menu item Code / Insert Chunk (or its keyboard shortcut) to put the following code chunk delimiters into your code:

```
```{r}
```
```

You could also just type these nine characters yourself. It is best (for later problem solving) to enter a space after the "r" and then any one or more words describing what the chunk does.

Enter normal R code inside the chunk. You do not need the extra `print()` statements for printing objects. You may still want to use `cat()` for nicer output. Use `dev.copy(png, file="myFile.png"); dev.off()` if you need a named copy of some key graphs. (When running myProg.Rmd, you can find *all* graphs in myProg-files/figure-html or -latex or -docx.)

d) Outside the chunks, enter "markdown" text for **interpretation** of results. See https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf.

e) Use the Code menu, keyboard shortcuts, or the buttons to the right of each chunk to **run or re-run chunk code** as you go along.

f) Use the Preview button to **create the full output file**. Use the little arrow next to preview to choose html, pdf or Word output.

g) ♦**Important**: Before sharing the Rmd file, select "clear objects from workspace", "clear plots", "clear console", and use the "Run / Restart R and Run All Chunks" button for a full test of your code.

h) Strong suggestion: ***Add a better Rmd Template***.

i) Find the folder containing "github_document" on your computer (if there is more than one, choose the one with the higher R version number). On a Mac look here: Macintosh HD / Library / Frameworks / .Rframework / Versions / 3.4 / Resources / library / rmarkdown / rmarkdown / templates.

ii) Download http://www.stat.cmu.edu/~hseltman/files/cache_with_help.zip into that folder.

iii) Unzip the file and delete the zip file. This will add a "cache_with_help" folder.

iv) In the future, use the menu item File / New File / R Markdown, select "From Template" and choose "Cache with Help". (The "Preview" button changes to "Knit".)

v) This improves cache handling for large files (top of template) and includes brief help on the key markdown features (bottom of file).

**i) Quick example of Rmd Markdown code**

````
## Simulate normal data for $R \sim N(\mu, \sigma^2)$
```{r sim}
x = rnorm(1000)
```

## Analyze these `r length(x)` simulated values
```{r analyze}
summary(x)
hist(x, breaks=31, ylab="x", main="")
```
````