WEEK 2: GETTING HELP

THOMAS ELLIOTT

1. R Help Files

All functions you can use in R come with help files (a requirement for publishing a package to CRAN is that all publicly available functions in your package must be documented). Much like Stata help files, R help files describe how to use a function. They typically include a brief description of what the function does, a listing of the parameters that you can pass to the function, and details about what the function returns. You can access help files in several different ways. If you know the name of a function, you can simply type `?nameoffunction` into the R command line and hit enter. If you are working in your OS's command line program, a plain text based version of the help file will be displayed in the command line window, which you can scroll using the up and down arrows. Typing q exits out of the help file. In RStudio, the help pane will activate, showing a PDF version of the help file. You can get the same functionality by typing `help("nameoffunction")`, which also allows you to pass additional parameters to the help function (type `?help` to find out more).

If you don't know the exact name of the function, you can type `??nameofsomething` - the two question marks means you want R to search the help files and return a list of relevant ones. You can then click on one of the results (if you are in RStudio) to take you to the help page. Alternatively, you can type `help.search("nameofsomething")`, which would allow you to include spaces in the query.

Below I go into more detail on each section of a help file. To follow along, type `?lm` into R to bring up the help file for the function used to run OLS regression.

Description: The description is usually one to two paragraphs briefly describing what the function is to be used for.

Usage: Usage lists out the function and all its parameters, including any default values the parameters take. If a parameter has a default value, that parameter is optional when you call the function — if you don't include it, it will take on the default value.

Arguments: This section provides more details about each of the parameters. It should tell you what type of object each argument is expecting, whether the argument is required when you call the function, and how the argument alters the behavior of the function.

Details: Not all help files include this section. This section often contains more explanation of how the function works, especially how various arguments work, should be formatted, etc. This is often done so that the Arguments section does not become overly crowded or complicated.

Date: January 7, 2016.

Value: This section describes what the function returns. For the lm function, it returns an object of class "lm" (which is just a list with a class attribute) that contains various elements, all of which are listed and briefly described in the help file.

References: This section contains formatted references for any works cited in the help file.

See Also: This section will list other functions that might be relevant to the current function the help file is for. This section of the lm help file contains references to other functions that are useful for manipulating the returned "lm" object, as well as other regression functions.

Examples: This section contains R code that shows how the function works. This R code should be functional for you to run if you copy and paste it into the R command line, so this is often helpful for getting a feel for new functions. These R code examples will often provide examples of what you can do with the returned values of the function as well. For example, in the lm help file, there is R code demonstrating how to plot the residuals of a fitted model.

2. Vignettes

A less common, but often more useful, form of help in R are vignettes - documents showing in detail how you might use a package or function. Vignettes often accompany packages and are often written at the package level — so rather than documenting individual functions, vignettes often document entire packages, showing how the various functions included in a package are to be used together. For example, the package dplyr (a very useful package developed by people at RStudio for data wrangling) comes with a number of different vignettes to document different aspects of the package. If you know the name of a vignette for a package, you can open it by typing

```
vignette("nameofvignette",package="nameofpackage")
```

For example, you can open the rotated vignette in the package "grid" by typing

```
vignette("rotated",package="grid")
```

This vignette shows how to use functions in the grid package to rotate plots.

If you don't know the name of the vignette, you can type `browseVignettes("nameofpackage")`, which will open a webpage listing any vignettes available for the package. You can then click on one of the vignettes to read it. If no vignettes are found, a message will be displayed in the R command line.

3. Loading and Saving Data

To continue from last week, loading and saving data in R works a little differently than in Stata. While Stata data files contain a single data set, R data files can contain an entire workspace, including multiple data frames as well as other data structures.

3.1. **Saving.** To save a copy of the workspace, including all loaded objects, use the function `save.image()`. You will need to supply a file name to which the workspace will be saved:

```
save.image("my-data.Rdata")
```

By default, the R data file created is a compressed, binary version of the workspace (Stata's data files are uncompressed, and so the same data often takes up between 1/5 to 1/10 the space in an R data file than a Stata data file). To change these settings, including which compression algorithm to use, how much to compress the data, and options to save in older R formats, you can supply additional arguments to `save.image()`. See the function's help file for details. I rarely adjust these settings.

There are times when I'm working that I've created a lot of extraneous objects in my workspace that I don't want to save. I'll show you in a later week how to remove these from your workspace, but an alternative when you are saving is to tell R which objects you want to save. You can do this using the `save()` function. You use `save()` by, first, listing out the objects you want to save and then supplying the named argument `file=` to tell R where to save the objects.

```
save(articles,claims,talk.info,file="my-data.Rdata")
```

The `file=` argument must be named, otherwise R will interpret "my-data.Rdata" as another object you want to save rather than the filename. The data file created with this function is identical to the file created using `save.image()`, it just gives you more control over what, exactly, gets saved. This means you can adjust the settings for version, compression, etc., in the same way with `save()`.

3.2. **Loading.** To load an R data file, you use the `load()` function:

```
load("my-data.Rdata")
```

The objects contained in the data file are loaded into your workspace. Your workspace does not need to be empty to load a data file, so you can load multiple data files to populate a single workspace. If a data file contains an object with the same name as one already loaded in your workspace, the object in your workspace will be over-written by the object in the data file, so pay attention to what you are loading versus what is already loaded to avoid losing data.