Introduction to R and RStudio

Katherine Thompson (`katherine.thompson@uky.edu`)
Department of Statistics, University of Kentucky
September 15, 2016

Introduction to R

- *Getting Started:* R is a free software environment for statistical computing and graphics and can be downloaded from `http://www.r-project.org/`.
- According to wikipedia.com: "The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls and surveys of data miners are showing R's popularity has increased substantially in recent years."
- *Advantages of R:* The R language is part of the GNU project which means that
  - the program is freely distributed,
  - the source code is available, and
  - any users can submit code/libraries so that other users can use the methods they have developed
- It is a bad idea to type commands directly into R, since these commands are often hard to track. Also, if a mistake is made in a command, it is hard to find and fix. *Instead*, use a text editor to write a script (e.g. filename.R) and either copy and paste the commands into the R console or use the `source()` function to run the script in R. If you use Windows, editors such as RStudio exist.

Introduction to RStudio

- *Getting Started:* RStudio is a free R-editor that can be used along with R. It can be downloaded from `http://www.rstudio.com/`. RStudio can be found under the start menu and the programs tab.
- There are four panels in the main RStudio window.
  1. *Console:* This is the place you can type R commands line-by-line.
  2. *Script Window:* This is where you can type R commands and save them so that you can reproduce or reanalyze your results.
     - To run commands, highlight the code you want to run and press `Ctrl + R` or click "Run" in the upper right hand corner of the panel.
  3. *Workspace/History:* Workspace shows all of the variables currently loaded in RStudio. History gives a list of all of the commands you have typed in this R Session.
  4. *Various Extra Features:* The two tabs I use most often are "Plots," which shows the current plot from R, and "Help," which displays the help for a function already built-in to R.

Help within R

The help files (as well as google) are very useful when learning about functions.

- If you know a function name (for instance, `mean()`) you can use either `help(mean)` or `?mean`.
- If you do not know a function name, search for applicable functions for what you want to do using either `help.search("mean")` or `??mean`.

There are *many* other resources for general help with R.

Setting the Working Directory

Before reading in data, it is convenient to set a "working directory." This specifies a default location for you to read files in from and write files to during a session. Using RStudio, you can set the working directory in two ways, a menu-driven way or by command.

Using Menus:

- In the bottom right window of the console, check to make sure the folder containing the data is shown. (If the folder is not shown, click "..." in the upper right corner of this window, and browse to the location of the folder.) Click "More" and "Set As Working Directory".

Using Commands:

Use the function `setwd()` to set the working directory path (see example below).

```
setwd("C:/Users/ukystat/Dropbox/QIPSR_RWorkshop") # Command to set working directory
getwd() # Displays current working directory

## [1] "C:/Users/ukystat/Dropbox/QIPSR_RWorkshop"
```

Notes:

- The "/" are forward slashes instead of backslashes here! Two backslashes, "\\" will also work.
- The function `getwd()` will display your current working directory.
- Using `file.choose()` will bring up a window so that you can browse directly to the file you are reading in and use this path.

Reading in data

Each time you analyze data in R, you will need to call in the data at the beginning of your script. The two functions I most commonly use are `read.table()` and `write.table()`.

```
read.table() # Reads data into R from a file
scan() # Reads data into R from a file (good when you need to specify a data type
       # for each column.)
write.table() # Writes data from R to a file
```

Note: There are other "flavors" of `read.table` that we will not use (such as `read.csv`) since `read.table` is flexible enough (if you change the arguments) to include comma delimited data.

In RStudio: Select Tools – Import Dataset – From...

Example: Let's read in some practice data. The data file is 'practicedata.txt' and can be downloaded from `http://web.as.uky.edu/statistics/users/klthomd/practicedata.txt`.

```
practicedata = read.table('practicedata.txt', # Give filename first
              header=TRUE, # If filename has variable names, set header to TRUE.
                    # Otherwise, use header=FALSE
              sep=",", # Symbol separating data values (comma here)
              na.strings="NA", # Characters used to denote missing values
              #comment.char='#', # Character used to indicate comments in your file
              #skip=0, # number of lines of data file to skip before reading in data
              #nrows=1000 # maximum number of lines of data file to read in
               )
```

Once the data is read in, we can check to see what it looks like by clicking on 'practicedata' in the upper right panel of the RStudio window.

```
practicedata[1:5,] # Prints first 5 rows of the data
practicedata[,1:2] # Prints first 2 columns of the data

practicedata[,"expvar"] # One way to call the variable, expvar
practicedata$expvar # Another way to call the variable, expvar
```

Suppose the first 50 data points are from a control group and the last 50 are from a treatment group, and you want to consider only the treatment group. This means you need to define a new variable (we'll call it `trtmtdata`) containing only the data associated with the treatment group.

```
k=50 # number of observations in each group
n=100 # total number of observations
trtmtdata = practicedata[(k+1):n, ] # Save the 51st through 100th rows of the data
```

Alternatively, we can use the `subset` function to subset the data according to the group variable. This does not depend on the ordering of the data.

```
trtmtdata = subset(practicedata , groupvar=='Treatment')
controldata = subset(practicedata , groupvar=='Control')
```

Writing Data to Files

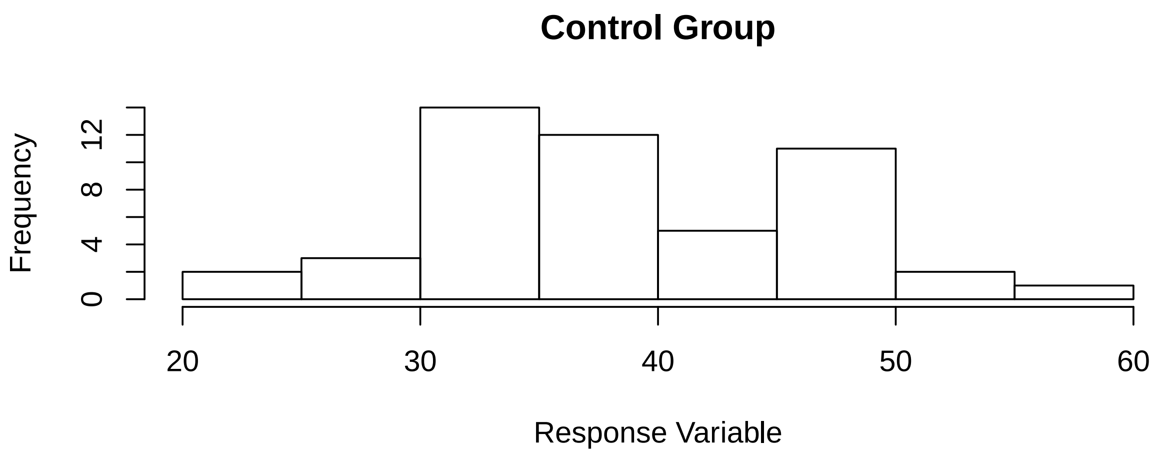
To write data to a file, the function `write.table()` is very flexible in terms of data formatting in the new file. As a default, the new file is created in the current working directory. For example, suppose you want to write the variables, `expvar, groupvar`, and the natural log of `respvar` to a new file. (Although, by saving your R script, it is not necessary to save the natural log of your data. If you need it again, you can re-run that line of code.)

```
getwd() #Check current working directory

## [1] "C:/Users/ukystat/Dropbox/QIPSR_RWorkshop"

resp.log=log(practicedata$respvar) # take the natural log of the response variable
data.to.write=cbind(practicedata$expvar,practicedata$groupvar,resp.log)
          # Binds the columns (variables) together
colnames(data.to.write) # print out column names of the new data

## [1] ""         ""         "resp.log"

colnames(data.to.write)<-c('expvar','groupvar','logrespvar') # rename the columns of
          # the new dataset
colnames(data.to.write) # print out column names of the new data again

## [1] "expvar"     "groupvar"   "logrespvar"

##To write data to a new .csv file:
write.table(data.to.write, # data to write to a file
  file='logdata.csv', # name of file you want to save data in
  quote=FALSE, # whether or not to put quotations around data
  col.names=TRUE, # whether or not to write column names to file
  row.names=FALSE, # whether or not to write row names to file
  sep=',', # what you want to put between data entries (commas and spaces are common)
  append=FALSE, # whether or not to append existing data to the current file
  na='NA' # string to use for missing values
  )
```

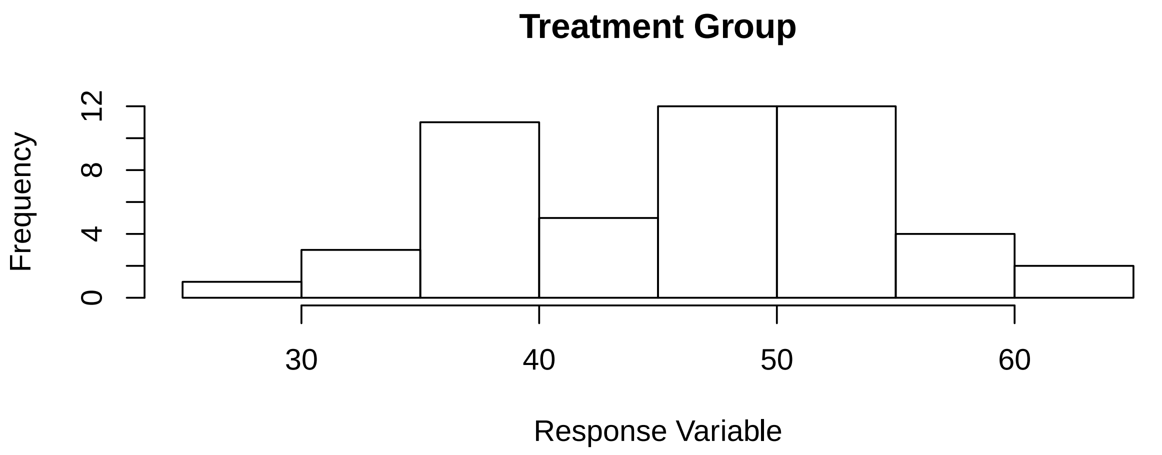
Some Basic Statistical Analyses

Suppose you are interested in determining if the response variable differs across treatment and control groups. To start, we can make histograms of the response variable for each group using the `hist` function. Next, we can find summary statistics about the data set using the `aggregate` function.

```r
## Histogram of control group data
hist(controldata$respvar,
    main='Control Group', # change the main title
    xlab='Response Variable'
    )
```



```r
## Histogram of treatment group data
hist(trtmtdata$respvar,
    main='Treatment Group', # change the main title
    xlab='Response Variable'
    )
```



```
## Find group means for response variable
aggregate((practicedata$respvar)~practicedata$groupvar,FUN=mean)

##   practicedata$groupvar (practicedata$respvar)
## 1               Control               38.58791
## 2             Treatment               45.91007

## Find group standard deviations for response variable
aggregate((practicedata$respvar)~practicedata$groupvar,FUN=sd)

##   practicedata$groupvar (practicedata$respvar)
## 1               Control               7.907639
## 2             Treatment               7.969457
```

Applied Statistics Lab (ASL):

The primary purpose of the ASL is to build bridges between statisticians and other investigators. We provide **statistical assistance** for grant submissions, study design, translational research, pilot studies, presentations, and publications.

- *To submit a request:* `https://redcap.uky.edu/redcap/surveys/?s=UurTv2mN49`
- *Email:* asl@uky.edu