A Two Page Guide to R
For Emerging Applied Researchers and Program Evaluators

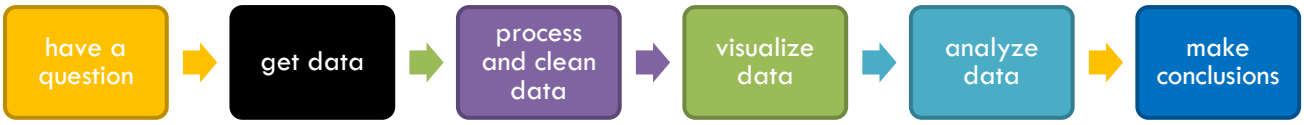
Andy Grogan-Kaylor

November 17, 2016

Why?

R is open source, and therefore free, statistical software that is particularly good at obtaining, analyzing and visualizing data. R has a reputation for being difficult to learn, and a lot of that reputation is deserved. However, I believe that it is possible to teach R in an accessible way, and that **a little bit of R can take you a long way**.

A great deal of data analysis and visualization involves the same core set of steps: get some data, clean it up a little, run some descriptive statistics, run some bivariate statistics, create a table or a graph or a visualization.



Given the fact that we often want to apply the same core set of tasks to new questions and new data, there are ways to overcome the steep learning curve and learn a replicable set of commands that can be applied to problem after problem.

Get R

R is available at https://www.r-project.org/. R is a lot easier to run if you run it from RStudio, http://www.rstudio.com.

Sometimes it is helpful to use **the menus** of RCommander to do your analyses, although the syntax produced by RCommander can be cryptic. This whole process is described in more detail here.

NB that R is originally a command or syntax based program, and many advanced functions are only available via syntax.

R Commands are stored in a *script* or *code* file that usually ends in .R, e.g. **myRscript.R**. The command file is distinct from your actual data, stored in an **.RData** file, e.g. **mydata.RData**.

The same 5 to 10 lines of R code can often be tweaked over and over again for multiple projects.

Get Data

Data often comes from other types of data files like SPSS, Stata, or Excel. Especially in beginning R programming, getting the data into R can be the most complicated part of your program.

```
library(foreign) # library for importing data from statistics packages

mydata <- read.spss("mySPSSfile.sav") # SPSS
mydata <- read.spss("myStatafile.dta") # Stata

library(readxl) # library for importing Excel files

mydata <- read_excel("mySpreadsheet.xls")
```

Manage Data[1]

```
mydata$x[mydata$x == -9] <- NA # change values that are supposed to be missing to NA
```

R makes a strong distinction between *continuous numeric* variables that measure scales like mental health or neighborhood safety, and *categorical factor variables* that measure non-ordered categories like religious identity or gender identity. Many statistical and graphical procedures are designed to recognize and work with the variable type.

```
mydata$z <- factor(mydata$z, # original numeric variable
                   levels = c(0,1), # levels of numeric variable
                   labels = c("Group A", "Group B")) # labels
```

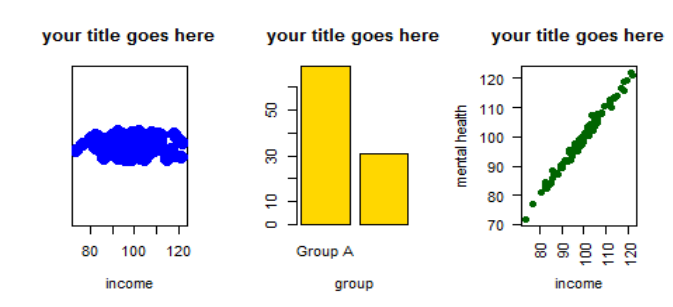
[1] The $ sign is a kind of "connector". mydata$x means: "The variable x in the dataset called mydata".

Visualize Data[2]

```
stripchart(mydata$x, # what I'm graphing
     main = "your title goes here", # title
     pch = 19, # Plot Character
     cex = 2.0, # Character EXpansion (how big the dots are)
     method = "jitter", # jitter the dots
     xlab = "income", # label for x axis
     col = "blue") # color

barplot(table(mydata$z), # what I'm graphing
        main = "your title goes here", # title
        xlab = "group", # label for x axis
        col = "gold") # color

plot(mydata$x,mydata$y, # plot x and y
     main = "your title goes here", # title
     xlab = "income", # label for x axis
     ylab = "mental health", # label for y axis
     pch = 19, # Plot CHaracter, 19 is filled dots
     las = 2, # LAbel Style, 2 is "perpendicular"
     col = "darkgreen") # color
```



Analyze Data

Descriptive Statistics

```
summary(mydata) # for continuous or factor variables
##        x                y                z
##  Min.   : 74.19   Min.   : 71.62   Group A:69
##  1st Qu.: 93.37   1st Qu.: 92.06   Group B:31
##  Median : 98.01   Median : 98.03
##  Mean   : 98.86   Mean   : 98.77
##  3rd Qu.:104.83   3rd Qu.:104.66
##  Max.   :121.65   Max.   :121.87
table(mydata$z) # especially suitable for factor variables
##
## Group A Group B
##      69      31
```

t-Tests and Correlations

A t-test is a test of whether there are differences between two groups. A correlation is a measure, varying between -1 and +1 of the relationship between two continuous variables.

```
t.test(mydata$y~mydata$z) # y is numeric; z is a binary factor
cor(mydata$x,mydata$y) # get the correlation
```

Regressions

A regression is an analysis of the influence of one or more independent variables on an outcome of interest.

```
lm(y ~ x, data = mydata) # regression of y on x
##
## Call:
## lm(formula = y ~ x, data = mydata)
##
## Coefficients:
## (Intercept)            x
##     -0.5435       1.0046
```

[2] When scatterplots have fewer dots than you think they should have, often due to "over-printing", adding some random noise, or "jittering" the dots in the scatterplot may help: `plot(jitter(mydata$y, factor = 5000) ~ mydata$x)`. Experiment with different sizes of *factor*.