A Two Page Guide to R

Andy Grogan-Kaylor

June 06, 2018

1 *Why?*

R is open source, and therefore free, statistical software that is particularly good at obtaining, analyzing and visualizing data.

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**.

R has a reputation for being difficult to learn, and a lot of that reputation is deserved.

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.

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

2 *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`.

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`.

3 *Get Data*[1]

```
load("mydata.Rdata")  # data in R format

library(foreign)  # library for importing data
mydata <- read.spss("mySPSSfile.sav")  # SPSS
mydata <- read.spss("myStatafile.dta")  # Stata

library(readxl)  # library for importing Excel files
mydata <- read_excel("mySpreadsheet.xls")
```

[1] 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.

4 *Manage Data* [2]

```
mydata$x[mydata$x == -9] <- NA  # missing to NA

mydata$w <- factor(mydata$z, # original numeric variable
                   levels = c(0, 1, 2),
                   labels = c("Group A", "Group B", "Group C"),
                   ordered = TRUE) # whether order matters
```

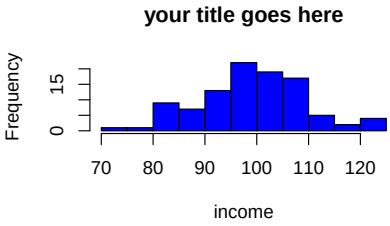
R makes a strong distinction between *continuous numeric* variables that measure scales like mental health or neighborhood safety, and *categorical factor variables* that measure non-ordered categories like religious identity or gender identity. Many statistical and graphical procedures are designed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

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

5 Visualize Data

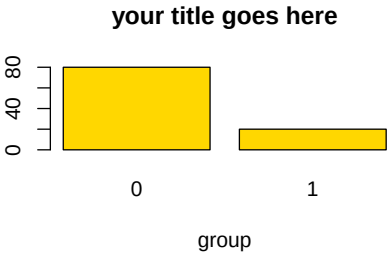
5.1 Histogram

```
hist(mydata$x, # what I'm graphing
     main = "your title goes here", # title
     xlab = "income", # label for x axis
     col = "blue") # color
```



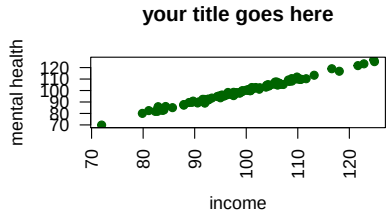
5.2 Barplot

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



5.3 Scatterplot

```
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
```



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*.

6 Descriptive Statistics

```
summary(mydata$x)  # for continuous or factor variables
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.
##   71.95   92.61   99.01   99.14  105.87
##    Max.
##  124.88
```

```
table(mydata$z)  # especially suitable for factor variables
```

```
##
##  0  1
## 80 20
```