Introduction to R

PNB 3XE3, Fall 2018

*Ali Hashemi**

2018-12-05

**Some of these examples are based off of the wonderfully thorough notes by Patrick Bennett in his offering of Graduate Statistics (PSYCH 710). His course is an excellent introduction to understanding statistical analyses using linear models while introducing students to R. His material is openly available for use, and is updated on a yearly basis:* http://psycserv.mcmaster.ca/bennett/psy710/index.html

Introduction

Download & Install

This document is intended to be a quick primer for how to navigate R using the RStudio IDE. To get the best out of this introduction, you need to install both R and RStudio, which can be downloaded for free from the websites linked below. Note that although you are installing two programs, RStudio is just a "wrapper" for R which makes it more user-friendly and adds a few capabilities you might need down the road.

Both programs are freely available for OSX/macOS, Windows, and Linux operating systems.

- R: https://cloud.r-project.org/
- RStudio: https://www.rstudio.com/products/rstudio/download/#download

Navigating RStudio

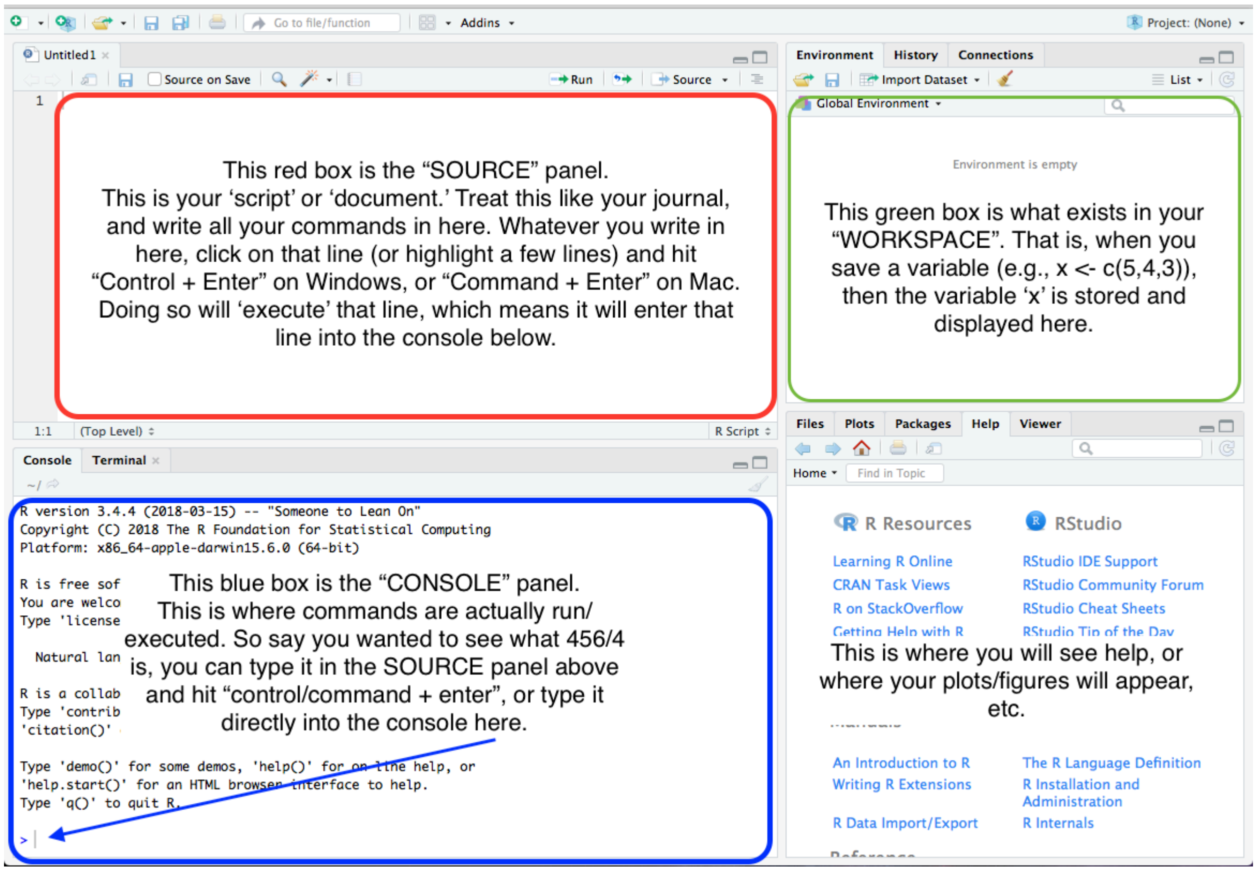
RStudio's default layout is shown in Figure 1. The best practice is to write your code in the SOURCE panel, and use the keyboard shortcuts `Command+Enter` (mac) or `Control+Enter` (windows) to execute that line in the CONSOLE. This way, you can save the script and refer to it, or resume working on it, later.

The top-right shows your "environment", also known as your WORKSPACE. When you create variables or load in data, they will show up and be stored here. The bottom-right is where you will view results when you 1) ask for help or 2) plot a figure.

R as a calculator

R's most basic but fundamental purpose is as a calculator. You can compute the simplest to the most complex arithmetics, as can be seen in the examples below.

```r
2+2
3*9
3-1
4^2
sqrt(10)
12/6
(3+5)/2
log(120) # natural log (ln)
log10(120) # base 10
log(8,base=2) # base 2
log10(8)/log10(2)
exp(6)
(11+4)/3 # follows BEDMAS
```



This red box is the “SOURCE” panel.
This is your ‘script’ or ‘document.’ Treat this like your journal, and write all your commands in here. Whatever you write in here, click on that line (or highlight a few lines) and hit “Control + Enter” on Windows, or “Command + Enter” on Mac. Doing so will ‘execute’ that line, which means it will enter that line into the console below.

This green box is what exists in your “WORKSPACE”. That is, when you save a variable (e.g., x <- c(5,4,3)), then the variable ‘x’ is stored and displayed here.

This blue box is the “CONSOLE” panel.
This is where commands are actually run/executed. So say you wanted to see what 456/4 is, you can type it in the SOURCE panel above and hit “control/command + enter”, or type it directly into the console here.

This is where you will see help, or where your plots/figures will appear, etc.

```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free sof
You are welco
Type 'license

  Natural lan

R is a collab
Type 'contrib
'citation()'

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Figure 1: Screenshot of RStudio’s default layout.

```r
(4-1)^4 # equivalent to 3^4
4-1^4 # importance of using brackets properly
```

R can also store variables, so you can use these variables in your calculations. There are also some very handy functions, such as mean(), sd(), var(), and sqrt(), that will calculate what their names imply. Functions are what make R, and other programming languages, useful. A function is a pre-defined algorithm/script that computes a specific series of calculations on the data you provide, and return to you a final product. For example, `mean(myData)`, will sum all of the values in the variable myData and divide them by the N (the number of values in myData), returning the computed mean. Functions often do *much more complicated* things than that, including computing your t-tests/ANOVAs and making figures.

```r
## Vectors & Variables

# you can save anything into a "variable"
k <- 4^2 #  store value into k using the <-
k # print output of k
k <- 5^2 # store value into k... note that this REPLACES original k
k # see that it replaces original k

x <- 12
y <- 8
theSum <- x + y
theProduct <- x * y
theSum
theProduct
X # does not exist because variables are cAsE-sEnSiTiVe
X <- 53
X
x
(X*x) / y

x <- c(-5, 20, 5, 1, -2, -3, 13) # create a small dataset!
x
x / 2
x + 10
x^2
y <- c(5, 2)

## Functions
mean(x)
mean(c(56, 78, 103, 42))
sd(x)
var(x)
sqrt(x) # can't take square root of a negative value
sqrt(abs(x)) # take square root of absolute value of x
sqrt(y)
length(x) # length calcualtes n (number of values)

sd(x) / sqrt(length(x)) # Q: does this formula look familiar?
```

Getting Help

The best way to learn R (or any programming language), is through trial-and-error. The best resource for learning how to use functions is through R's built-in `help()` function and online resources (e.g., Google, StackOverflow). Below are some examples of how to ask R for help on using the `log()` and `t.test` functions.

```r
?log
help("log")
help("+")
?help.search
help.search("t.test")
```

Introduction to experimental paradigm

Accurate face identification is a fundamental aspect of a healthy human's visual functioning. The human visual system is really good at identifying faces, with plenty of evidence to suggest that there are regions in the brain that are dedicated to face perception. Interestingly, humans are *really bad* are identifying faces when faces are presented upside down, even though they can accurately identify the same face when presented upright. The difference in accuracy to identifying upright and inverted faces is referred to as the Face Inversion Effect (FIE), and it is a highly robust finding.

I have provided a simulated dataset (download from avenue or the link below) from an experiment where participants were briefly shown one face, and then had to choose which face they just saw from a lineup of faces. Half of the participants saw the faces as upright, and the other half saw them as inverted. Two independent variables are provided: accuracy (`pc`) and reaction time (`rt`).

To load the data file into RStudio, you have two options.

Option 1. Download data file, save in a folder, and load in from that folder

```r
# Check what the current working directory is
getwd()

# set your working directory to the project folder that contains the data file
setwd("/Users/alihashemi/Dropbox/Academic/Guest Lectures/")

# check what is inside your directory (and confirm your data file is there)
dir()

# load in your data
faceData <- read.csv("faceData.csv", header=T)
```

Option 2: Load datafile directly from online source

```r
faceData <- read.csv(url(
  "https://raw.githubusercontent.com/HashemiScience/Rintro/master/faceData.csv"),
  header=T)
```

Inspect the data

It is always recommended that you get familiar with the data file you just loaded. You want to know what the data looks like, how many variables there are, what are the types of variables, etc… Below are some basic ways to inspect your data and make simple figures.

```r
# inspect your data
# *please ignore the column ageGroup for now*
faceData # shows everything... but this is too much info
head(faceData) # shows top 10 rows... just enough to know the structure of the data
class(faceData) # what class of variable is the whole dataset?
class(faceData$faceOrient) # what class is the variable "faceOrient" inside faceData?
summary(faceData) # summarizes everything

# boxplot the data
boxplot(pc ~ faceOrient, data=faceData)
# the above structure using "~"" (tilda) will be commonly used.
# tilda represents "as a function of" ... pc as a function of faceOrient.
# it will always be "dependent variable as a function of independent variable(s)"
# i.e., DV ~ IV

# make a bar graph of the group means
# calculate means / SE:
theMean <- with(faceData, tapply(pc, faceOrient, mean))
theSD <- with(faceData, tapply(pc, faceOrient, sd))
n <- with(faceData, tapply(pc, faceOrient, length))
theSE <- theSD / sqrt(n)

# plot bar graph
bp <- barplot(theMean, ylim=c(0,1))

# add error bars
segments(bp, theMean-theSE, bp, theMean+theSE) # add standard error bars
```

t-tests

Doing t-tests in R is pretty straightforward:

```r
# One sample t-test to see if mean PC is different than 0
t.test(faceData$pc)

# two independent sample t-test to see if PC is different between younger and older
t.test(pc ~ faceOrient, data=faceData, var.equal=TRUE)

# repeat above, but one-tailed test (is older PC less than younger PC?)
t.test(pc ~ faceOrient, data=faceData, var.equal=TRUE, alternative="less")

# what if data from two groups had unequal variances?
t.test(pc ~ faceOrient, data=faceData, var.equal=FALSE)

# what if our data was paired-samples design?
t.test(pc ~ faceOrient, data=faceData, paired=TRUE)
```

ANOVA

One reason R is seen as a learning tool for statistics rather than just an analysis tool is that it demonstrates, by design, how many seemingly different statistical tests are *actually the same thing!*

To better appreciate that fact, keep in mind the format of the t-test function…

```r
# reminder of what a t-test looked like:
t.test(pc ~ faceOrient, data=faceData, var.equal=TRUE)

# conducting a one-way ANOVA
oneWay.01 <- aov(pc ~ faceOrient, data=faceData)
summary(oneWay.01)
```

```
# can all be done in step...
summary(aov(pc ~ faceOrient, data=faceData))

#  t-tests and ANOVAs *are the same test*
# but ANOVA allows for multiple groups and factors.
# When only 2 groups, F = t^2... or t = sqrt(F)
sqrt(12.08)
```

Factorial ANOVA

Factorial ANOVAs are just an extension of one-way ANOVAs.

Revisiting what the data in `faceData` looks like, we can see that there's another column, `ageGroup` that we have been ignoring. Turns out half of the participants in this simulated experiment were older adults, and half were younger adults. We know that older adults are not as good as younger adults at identifying faces. The quesion is, is the Face Inversion Effect different for younger and older adults? *What aspect of a factorial ANOVA will answer this question?*

```
# our data.frame has another variable, ageGroup
summary(faceData)

boxplot(pc ~ faceOrient, data=faceData)
boxplot(pc ~ ageGroup, data=faceData)
boxplot(pc ~ faceOrient:ageGroup, data=faceData)

## barplot the data
# calculate means / SE:
theMean <- with(faceData, tapply(pc, list(faceOrient,ageGroup), mean))
theSD <- with(faceData, tapply(pc, list(faceOrient,ageGroup), sd))
n <- with(faceData, tapply(pc, list(faceOrient,ageGroup), length))
theSE <- theSD / sqrt(n)

# plots bars; note that beside=T is needed to avoid stacked bars
bp <- barplot(theMean, ylim=c(0,1), beside=T)
segments(bp, theMean-theSE, bp, theMean+theSE) # add standard error bars

factorial.01 <- lm(pc ~ faceOrient + ageGroup + faceOrient:ageGroup, data=faceData)
anova(factorial.01)

# easier way, where * factorially crosses all groups
anova(lm(pc ~ faceOrient*ageGroup, data=faceData))
```

Within-subjects ANOVA

Within-subject ANOVAs are typically more powerful than between-subject ANOVAs because the same participant provides data in 2 or more conditions. Therefore, unlike between-subject ANOVAs, we don't have to worry about random inter-subject differences confounding our group differences.

```
# modify data to have within subject ID
faceData$subjID2 <- factor(c(1:10,1:10, 11:20,11:20), label="s")

# confirm it was added to dataframe
head(faceData)
```

```
# create the analysis of variance model
within.01 <- aov(pc ~ faceOrient + Error(subjID2), data=faceData)

# compute f-tests
summary(within.01)
```

Correlation

```
# data has two dependent variables: percent correct and reaction time
head(faceData)
with(faceData, plot(pc, rt))

cor.01 <- lm(rt ~ pc, data=faceData)
abline(cor.01)
summary(cor.01)

with(faceData, cor.test(x=pc, y=rt))

# note that Pearson's r is just square-root of Regression R [sqrt(R^2)]
sqrt(0.3155) # should equal 0.56

# analysis of covariance (ANCOVA)
# note that both faceOrient and ageGroup are treated as between subj here
summary(aov(pc ~ rt + faceOrient*ageGroup, data=faceData))
```

Using Packages

One reason R is a a great teaching tool is that using it well actually aides in your learning, unlike other programs such as SPSS that treat the analysis like a black-box, and you have to follow a sequence of arbitrary button-clicks to get a certain output. R, by design, requires that most of what you do is intentional.

With that said, and with R's growing popularity, there is a massive community of developers extending the capabilities and usability of R through what are called `packages`. I've described some of these below. They are good shortcuts, but I really advise you do learn using the methods above, and only transition to these easier shortcuts after you know what you are doing.

ez packages

This package includes several easy-to-use functions (as the name implies) for doing inferential statistics. The most useful function is the `ezANOVA()` function, which lets you do one-way, factorial, within-subject, an mixed-design ANOVAs. One extra benefit is that it automatically calculates measures of effect size (generalized eta squared) for all main effect and interactions. Where needed (within-subject factors with >2 levels) it also tests for sphericity and makes the appropriate correction (Greenhouse-Geiser or Huynh-Feldt, you choose which to report).

Note: `packages` must be installed on each computer once, but loaded into the library each time you open R and want to use the package.

```
# must be installed only once on your computer using this:
# install.packages("ez") # uncomment me to install, then comment back to avoid re-install

# on every session, must load ez to library
library(ez)
```

```
# get help of what functions are included
?ez

# get help on how to use ezANOVA
?ezANOVA

# run one-way ANOVA
ezANOVA(data=faceData, dv=pc, wid=subjID, between=faceOrient)

# run 2x2 factorial ANOVA
ezANOVA(data=faceData, dv=pc, wid=subjID, between=.(faceOrient,ageGroup))

# run one-way within-subject ANOVA
ezANOVA(data=faceData, dv=pc, wid=subjID2, within=faceOrient)

# run a mixed-design factorial ANOVA (faceOrient within, ageGroup between)
ezANOVA(data=faceData, dv=pc, wid=subjID2, within=faceOrient, between=ageGroup)
```

ggplot2 package

The default plotting tools in R are really capable, but they can get very cumbersome. One really handy and widely used package is ggplot2, and it is excellent for making all types of figures. I should note that ggplot is part of what is called the Tidyverse. The tidyverse is a whole suite of libraries that use a similar language and work really well together. Other popular packages part of the tidyverse are `dplyr`, `tidyr`, and `tibble`.

Note that packages part of the tidyverse, and many packages not shipped with the base R, tend to use a slightly different syntax. Some prefer these, others don't. What's clear is that it takes a bit of time to get ued to it, but once you do get used to it, these packages are extremely helpful.

```
# install.packages("ggplot2") # uncomment me for first time install
library(ggplot2)

# make barplot like we did earlier, but use ggplot
ggplot(data=faceData, aes(x=ageGroup, y=pc, fill=faceOrient, group=faceOrient)) +
  stat_summary(fun.y = mean, geom = "bar", position="dodge") +
  stat_summary(fun.data = mean_se, geom = "errorbar", position="dodge")

# do scatter of rt ~ pc, and fit 1 line to everyone combined
ggplot(data=faceData) +
  geom_point(aes(x=pc, y=rt, col=ageGroup, shape=faceOrient)) +
  stat_smooth(aes(x=pc, y=rt), method="lm")

# remake scatter plot, but fit line to younger and older separately
ggplot(data=faceData) +
  geom_point(aes(x=pc, y=rt, col=ageGroup, shape=faceOrient)) +
  stat_smooth(aes(x=pc, y=rt, col=ageGroup), method="lm")
```

Other cool things & useful links

You might be surprised to learn that I made this entire document within RStudio, using something called `RMarkdown` and `knitr` (google them to learn!). I will continue to improve this document and create other resources, and will upload everything to a GitHub repository linked below. I've linked the ".Rmd" file used to create this document, which is the script that is capable of executing R code, as well as write PDF/HTML documents.

Useful links:

1. Ali Hashemi's repository for storing this document (and other resources in the future as I develop them)

- https://github.com/HashemiScience/Rintro

2. A work-in-progress statistics textbook written in R

- https://crumplab.github.io/statistics/

3. Patrick Bennett's online resources for graduate statistic taught in McMaster's PNB department:

- http://psycserv.mcmaster.ca/bennett/psy710/index.html

4. A gallery of nice R-made figures, and code so you can make them too:

- https://www.r-graph-gallery.com/

5. Using R for data science:

- https://r4ds.had.co.nz/

6. A series of online modules teaching descriptive/inferential statistics created by folks from the PNB department (headed by Dr. Scott Watter):

- https://macblog.mcmaster.ca/statistics/