Homework 3 Code Review

Zoe Vernon

01 October, 2020

Code review instructions

In section this week we will do a paired code review for PS3. In order for this to be beneficial, you will need to be as honest with your partner as possible. If something is actually hard to follow, tell them! If you thought that something they did was clever, also tell them!

Procedure

We will break into pairs of 3 (or 2 if necessary) using breakout rooms in Zoom.

Next, you will spend ~10 minutes reading one partner's code, asking questions to the person who's code you are reviewing. You will rotate this process until each person in the group has their code reviewed. I will float between breakout rooms to make sure things are going smoothly.

Finally, each of you will fill out the **bCourses assignment PS3 paired code review** summarizing the differences between yours and your two (or one) partner's work, noting anything that was particularly novel to you.

The bCourses assigment is worth 1 point, similar to in class group work or the unit check ins, and will be graded. You will receive credit as long as your response is thoughtful (e.g. at least a couple of sentences). If you choose not to come to section, please find a partner or two on your own to do the paired review with and submit the bCourses assigment by Wed Oct. 7 at 5:59pm to recieve credit.

Some Guiding Questions

1. Is the code visually easy to break apart? Can you see the entire body of the functions without scrolling up/down left/right? The general rule-of-thumb is no more than 80 in either direction (80 character width, 80 lines. The first restriction should be practiced more religiously)
2. Are the data structures easy to parse? Do you understand what information is in which objects? Is the type of data appropriate?
3. Are the functions easy to parse? Is it clear what they are doing, what information is being passed to each function, and what the return value/objects are?
4. Look at the regex. Are there any edge cases that were missed? Was anything handled particularly well? Any novel expressions that you didn't think of?
5. Look at the plots. Are they well labeled? Is it easy to read information from them? Do they provide an assurance that the code is running correctly? Was there anything particularly neat about your partner's solution to the bonus?
6. EXTRA. Look at problem 2, compare/contrast your approaches to OOP. Explain the logic behind your design.

General notes about homeworks

Below are a couple notes about to keep in mind when doing your problem sets.

1. Problem Statements
   You **have to** answer the questions on the homework. Do whatever you'd like beyond that, be as creative as you wish, but **you must answer the questions that were asked**.

2. Include Code
   Most people include it as they go along, that's fine. If you find that messy, put an appendix at the end with all of the function definitions. This applies to code for plots as well.

3. A Couple of Useful Tools in RStudio
   You can autoindent your code in RStudio using Command or Ctrl + I. You can also autoformat your code with Shift + Command or Ctrl + A. There are some other useful shortcuts in the Code menu.

4. Bonus/Extra Credit
   This is something additional, above and beyond the basic homework. Repeating the analysis already performed, but with different parameters, is not novel.

5. Please use `suppressPackageStartupMessages()` or set `warning = FALSE` and `message = FALSE` in a code chunk that **only** loads the packages. Please don't use these chunk options on other chunks. If something is going wrong you need to fix it, as opposed to not printing the warning.

6. Plots and Comments
   Plots need to have a title, axis labels, and a key if there are several types of data. Code and comment lines should be ~80 characters in length. e.g.

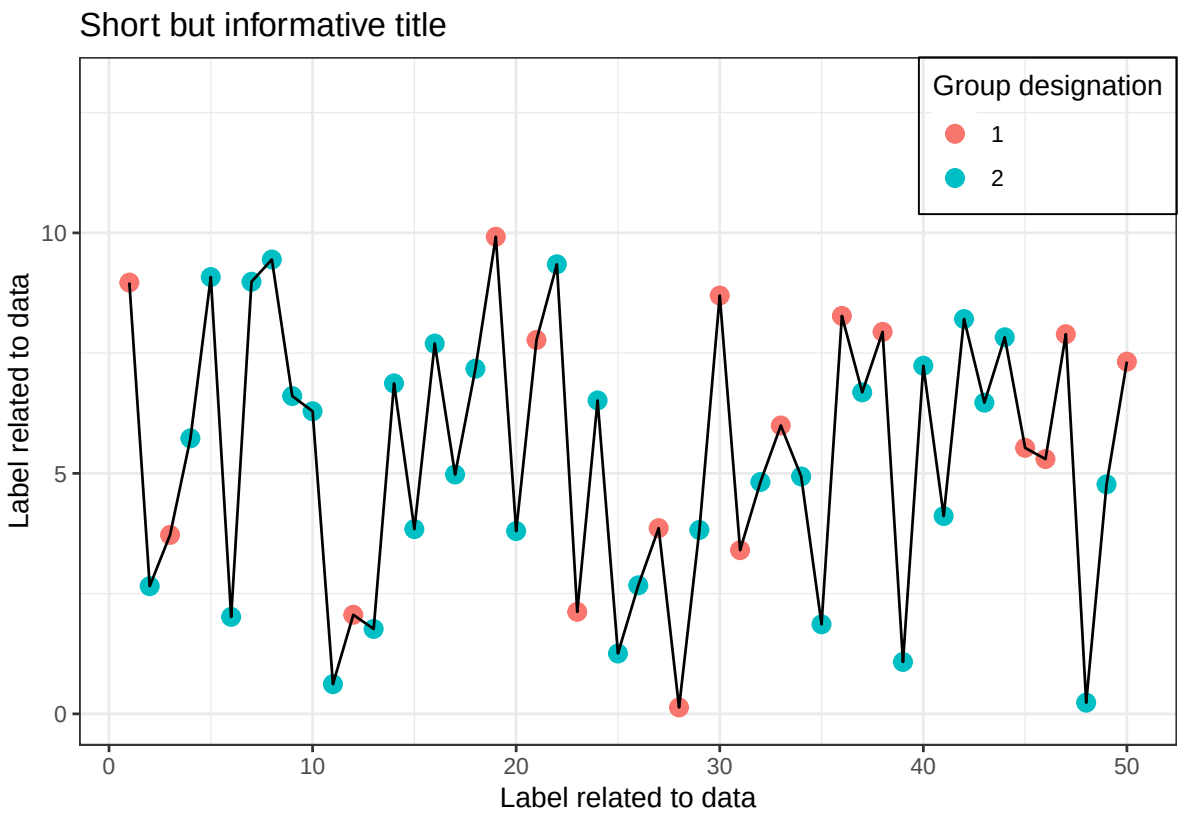
```
####################
## Make Comment point
####################
# This is a bad comment
# Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labo

# This is a good comment
# Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor
# incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
# nostrud exercitation ullamco laboris nisi ut aliquip ...etc

####################
## Good plotting
####################
# setup things
n <- 50
x <- 1:n
y <- runif(n = n, min = 0, max = 10)
col_factor <- sample(x = c(1, 2), size = n, replace = T, prob = c(1, 2))
df <- data.frame(x, y, color = as.factor(col_factor))

# make the plot
ggplot(df, aes(x = x, y = y)) +
  geom_point(aes(color = color), size = 3) +
  geom_line() +
  theme_bw() +
  labs(x = "Label related to data", y = "Label related to data") +
  ggtitle("Short but informative title") +
  scale_color_discrete(name = "Group designation") +
```

```
theme(legend.position = c(1, 1),
      legend.justification = c(1, 1),
      legend.background = element_blank(),
      legend.box.background = element_rect(colour = "black")) +
ylim(0, 13)
```



7. Vectorizing

   "Vectorizing" in R implies using functions that operate in parallel over certain objects. R is vectorized in many operations (addition, subtraction, multiplication, division). Many functions have built-in vectorization as well, make sure to check the function documentation.

```
####################
## vectorized division
####################
x <- 1:100
y <- 100:1

mHold <- mapply(FUN = "/", x, y)
dHold <- x/y

all.equal(mHold, dHold)
```

```
## [1] TRUE
```

```
identical(mHold, dHold)
```

```
## [1] TRUE
```

```
####################
## vectorized multinomial
####################
myprobs <- matrix(data = c(0.1,0.1,0.8, 1/3,1/3,1/3), nrow = 2, byrow = T)

set.seed(0)
apHold <- t(apply(X = myprobs, MARGIN = 1, FUN = rmultinom, n = 1, size = 100))

set.seed(0)
edHold <- extraDistr::rmnom(n = 2, size = 100, prob = myprobs)

all.equal(apHold, edHold)
```

```
## [1] TRUE
```

```
identical(apHold, edHold) # why false?
```

```
## [1] FALSE
```

```
# short benchmark
microbenchmark::microbenchmark("apply" = apply(X = myprobs, MARGIN = 1,
                                               FUN = rmultinom, n = 1, size = 100),
                               "vector" = extraDistr::rmnom(n = 2, size = 100,
                                                            prob = myprobs),
                               times = 100)
```

```
## Unit: microseconds
##    expr    min      lq     mean  median      uq    max neval
##   apply 29.530 30.6865 34.72829 32.5500 35.7140 96.513   100
##  vector 10.418 11.2875 13.78223 13.3495 15.0135 34.945   100
```