Problem set 2: Worked solutions

Statistics and statistical programming
Northwestern University
MTS 525

Aaron Shaw

October 5, 2020

Contents

Programming challenges

PC 1: Import data from a .csv file

So, the nice thing about running `set.seed()` before I chose my random group number in the previous problem set is that I can replicate the **exact same** random draw again:

```
set.seed(220920)
sample(x=c(1:20), size=1)
```

```
## [1] 3
```

My dataset number for the rest of this programming challenge is, once again, 3. I'll read the dataset in using the `read.csv()` command (note that I assign it to an object).

```
w4 <- read.csv(url("https://communitydata.science/~ads/teaching/2020/stats/data/week_04/group_03.csv"))
```

PC 2: Explore and describe the data

Note: You can complete this programming challenge in numerous ways. What follows is one path that uses a number of the commands introduced in the r tutorials provided for the course

Let's start by just seeing what's here:

```
dim(w4)
```

```
## [1] 100   5
```

```
lapply(w4, class)
```

```
## $x
## [1] "numeric"
##
## $j
## [1] "integer"
##
## $l
## [1] "integer"
##
## $k
## [1] "integer"
##
## $y
## [1] "numeric"
```

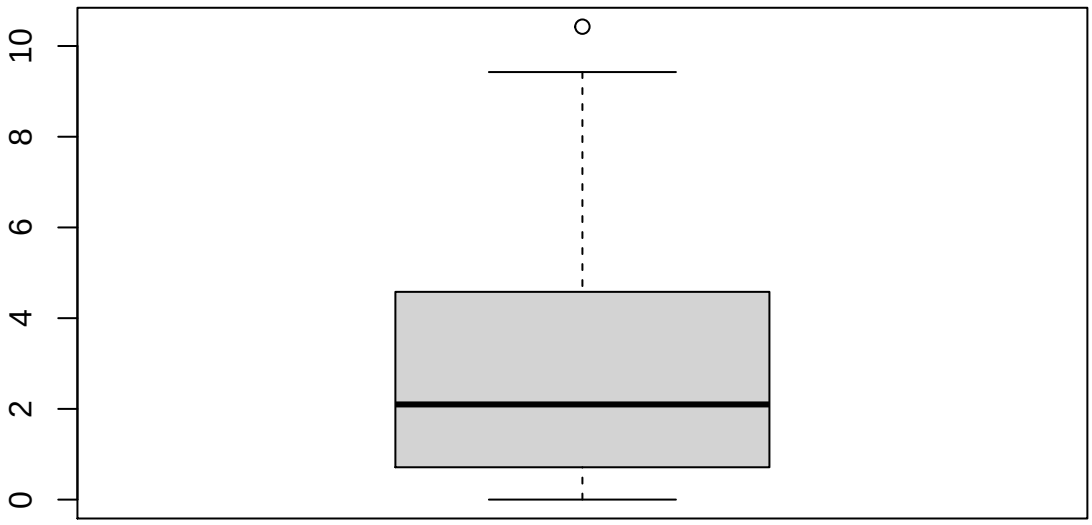
Summarizing the variables:

```
summary(w4)
```

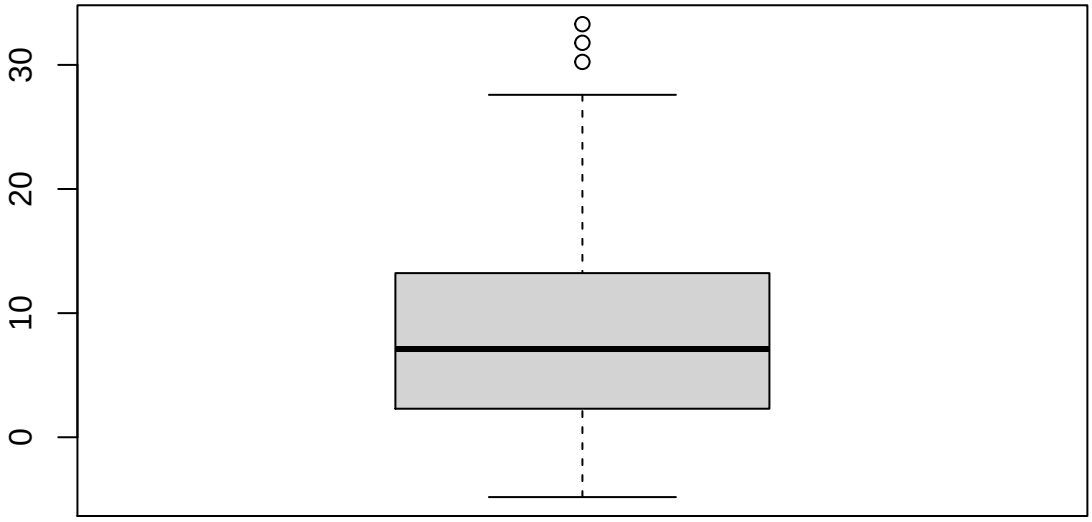
```
##        x                    j              l              k
##  Min.   : 0.001016   Min.   :0.00   Min.   :0.00   Min.   :1.00
##  1st Qu.: 0.713532   1st Qu.:0.00   1st Qu.:0.00   1st Qu.:1.00
##  Median : 2.098321   Median :1.00   Median :0.00   Median :2.00
##  Mean   : 2.845581   Mean   :0.52   Mean   :0.45   Mean   :1.99
##  3rd Qu.: 4.580239   3rd Qu.:1.00   3rd Qu.:1.00   3rd Qu.:3.00
##  Max.   :10.425881   Max.   :1.00   Max.   :1.00   Max.   :3.00
##  NA's   :5
##        y
##  Min.   :-4.820
##  1st Qu.: 2.300
##  Median : 7.110
##  Mean   : 8.774
##  3rd Qu.:13.225
##  Max.   :33.280
##  NA's   :5
```

It looks like `x` and `y` are continuous. That last call to `summary()` gave us the range, median, mean and IQR for both. I'll go ahead and create some boxplots to see which measure of spread might be most appropriate:

```
boxplot(w4$x)
```




```
boxplot(w4$y)
```



Both are a little skewed with a longer "right" tail. Given that the skew isn't totally egregious, I'll calculate the standard deviation for each as well (and remember the `na.rm=TRUE` bit otherwise this won't work).

```
sd(w4$x, na.rm=TRUE)
```

```
## [1] 2.667537
```

```
sd(w4$y, na.rm=TRUE)
```

```
## [1] 8.965738
```

Despite the class R attributes to the other variables, it sure looks like j, l, and k are categorical/discrete variables (note the nice round numbers in the range), so let's look at the first few values of each to see if that seems right:summarize those with contingency tables

```
head(w4[, c("j","l", "k")])
```

```
##   j l k
## 1 1 0 1
## 2 1 0 2
## 3 0 1 1
```

```
## 4 0 1 1
## 5 0 1 1
## 6 1 0 2
```

This confirms my intuitions, so I'll go ahead and create univariate contingency tables for each:

```
table(w4$j)
```

```
##
##  0  1
## 48 52
```

```
table(w4$l)
```

```
##
##  0  1
## 55 45
```

```
table(w4$k)
```

```
##
##  1  2  3
## 36 29 35
```

PC 3: Use and write user-defined functions

For starters, I'll go copy-paste that `my.mean()` function from the tutorial. Remember, that unless I execute the code that defines the function, R doesn't know that it exists! Once I've executed it, I can call it and calculate the value for my `x` variable.

```
my.mean <- function(z){
  z <- z[!is.na(z)]
  sigma <- sum(z)
  n <- length(z)
  out.value <- sigma/n
  return(out.value)
}

my.mean(w4$x)
```

```
## [1] 2.845581
```

Creating a function to calculate the median is quite a bit harder! Medians can be complicated because the calculation depends on whether there's a midpoint of the data or not. Luckily in this case, there should be 95 non-missing values in the x vector, so a midpoint exists (the 48th value) and the following should work:

```
my.median <- function(z){
  z <- z[!is.na(z)]
  midpoint <- (length(z) / 2) + .5
  sorted <- sort(z)
  output <- sorted[midpoint]
  return(output)
}

my.median(w4$x)
```

```
## [1] 2.098321
```

I can check my work here using R's built-in functions:

```
mean(w4$x, na.rm=TRUE)
```

```
## [1] 2.845581
```

```
median(w4$x, na.rm=TRUE)
```

```
## [1] 2.098321
```

Note that the functions here work for this problem set and the `x` variable provided here, but might break under other circumstances. Can you think of how they might break? Can you imagine ways that you might rewrite either the `my.mean()` or the `my.median()` functions presented here (or in your own solutions) to be more robust to these potential problems?

PC 4: Compare two vectors

For starters, I have to load and cleanup my week 3 variable again:

```
load(url("https://communitydata.science/~ads/teaching/2020/stats/data/week_03/group_03.RData"))

d[d < 0] <- NA
d <- log1p(d)
```

Onwards to some comparisons. At first glance, they look very similar...

```
summary(w4$x)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
##  0.001016  0.713532  2.098321  2.845581  4.580239 10.425881         5
```

```
summary(d)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
##  0.001016  0.713532  2.098321  2.845581  4.580238 10.425881         5
```

But closer inspection suggests something is off...

```
table(w4$x == d)
```

```
##
## FALSE
##    95
```

It might be good to look at the first few values of each to see if there's anything noteworthy...

```
head(w4$x)
```

```
## [1] 1.794517 0.572927 3.256708 1.009964 0.744927 6.120917
```

```
head(d)
```

```
## [1] 1.7945167 0.5729273 3.2567082 1.0099642 0.7449274 6.1209172
```

It looks like the vectors might match if I round them to six decimal places. I can create a table comparing the sorted rounded values to check this.

```
table(round(w4$x, 6) == round(d, 6))
```

```
##
## TRUE
##   95
```

Note that you should be able to explain what each part of that last line of code is doing!

PC 5: Cleanup/tidy your data

```
w4$j <- as.logical(w4$j)
w4$l <- as.logical(w4$l)

### Note that I am creating a new column in my w4 dataframe so I can double check this:
w4$k.factor <- factor(w4$k,
                      levels=c(0,1,2,3),
                      labels=c("none", "some", "lots", "all"))

### Spotcheck to make sure it looks good
head(w4, 10)
```

```
##           x     j     l k     y k.factor
## 1  1.794517  TRUE FALSE 1  9.38     some
## 2  0.572927  TRUE FALSE 2  4.72     lots
## 3  3.256708 FALSE  TRUE 1 14.27     some
## 4  1.009964 FALSE  TRUE 1  6.03     some
## 5  0.744927 FALSE  TRUE 1 -2.77     some
## 6  6.120917  TRUE FALSE 2 17.86     lots
## 7  0.711756  TRUE  TRUE 2 -2.86     lots
## 8  6.433661  TRUE  TRUE 3 18.30      all
## 9  8.108382 FALSE FALSE 3 20.83      all
## 10 0.336649  TRUE  TRUE 2  2.51     lots
```

```
summary(w4$k)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    2.00    1.99    3.00    3.00
```

```
### Now cleanup my extra column:
w4$k <- w4$k.factor
w4$k.factor <- NULL

head(w4)
```

```
##          x     j     l    k     y
## 1 1.794517  TRUE FALSE some  9.38
## 2 0.572927  TRUE FALSE lots  4.72
## 3 3.256708 FALSE  TRUE some 14.27
## 4 1.009964 FALSE  TRUE some  6.03
## 5 0.744927 FALSE  TRUE some -2.77
## 6 6.120917  TRUE FALSE lots 17.86
```

PC 6: Calculate conditional summary statistics

As usual, there are many ways to approach this. My approach here uses the `tapply()` function, passing in calls to the `summary()` and `sd()` functions respectively:

```
tapply(w4$x, w4$k, summary)
```

```
## $none
## NULL
##
## $some
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## 0.001016 0.865321 2.304208 2.591130 3.256708 7.775751        3
```

```
##
## $lots
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0613  0.5729  2.4867  2.9398  4.7787  9.4254
##
## $all
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
##  0.01998  0.71531  1.44077  3.01726  4.92328 10.42588        2
```

```
tapply(w4$x, w4$k, sd, na.rm=TRUE)
```

```
##     none     some     lots      all
##       NA 2.229753 2.706289 3.068719
```

Here's a tidyverse example that achieves the same goal:

```
library(tidyverse)

w4 %>%
  group_by(k) %>%
  summarize(
    n= n(),
    min_x = min(x, na.rm=T),
    max_x = max(x, na.rm=T),
    mean_x = mean(x, na.rm=T),
    sd_x = sd(x, na.rm=T)
  )
```

```
## # A tibble: 3 x 6
##   k         n   min_x max_x mean_x  sd_x
##   <fct> <int>   <dbl> <dbl>  <dbl> <dbl>
## 1 some     36 0.00102  7.78   2.59  2.23
## 2 lots     29 0.0613   9.43   2.94  2.71
## 3 all      35 0.0200  10.4    3.02  3.07
```

Note that the tidyverse has "silently" dropped our "none" level from the summary because there are no values of it in the dataset. This could be altered by passing an argument of `.drop=FALSE` to the `group_by()` call. It produces very ugly results, so I have ommitted it here.

PC 7: Create a bivariate table

```
table(w4$k, w4$j)
```

```
##
##        FALSE TRUE
##   none     0    0
##   some    15   21
##   lots    18   11
##   all     15   20
```

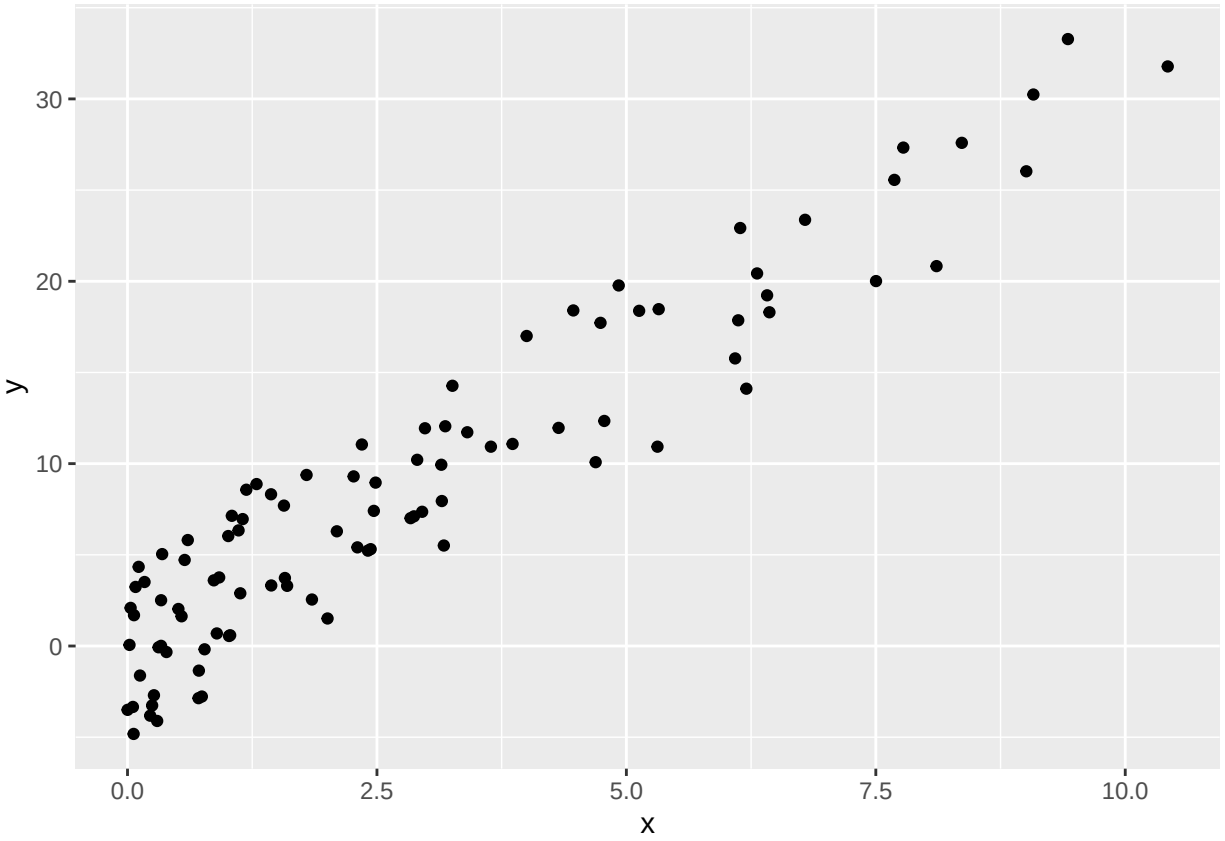
PC 8: Create a bivariate visualizations

First create a basic plotting object. Then add the call to `geom_point()` to show just the x and y:

```
p <- ggplot(w4, mapping=aes(x=x, y=y))

p + geom_point()
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```



Now, let's add some additional dimensions to the plot using our other variables. Note that the following line of code works very well with the recoded version of the dataset, but will produce errors if I try to run it on the original version that was treating `j`, `k`, and `l` as integers:

```
p + geom_point(aes(color=j, shape=l, size=k))
```

```
## Warning: Using size for a discrete variable is not advised.
```

```
## Warning: Removed 5 rows containing missing values (geom_point).
```

This is very visually cluttered and therefore probably not a great data visualization. It's hard to tell whether there might be any patterns other than the positive association of `x` and `y` reflected in the very clear upward slope of the points. That said, it's good to be able to incorporate multiple dimensions of variation into your plots and sometimes you can learn/convey a lot by doing so.

Statistical questions

SQ 1: Interpret bivariate analyses

1.1 Interpret conditional means The most noteworthy thing is that none of the riders logging miles in this subset of the dataset fell into the "none" category measuring how frequently they rode in weather defined to be "bad" by the NOAA. They must be a hardy bunch of cyclists! Bayond that, the only possible pattern seems like there might be a weak association between the distance ridden and riding on "all" bad weather days (the mean and max are slightly higher in this group than the others). However, this might just reflect the presence of a few outliers as the spread (standard deviation) of distances ridden among those in the "all" bad weather days ridden category is also a bit bigger than in other categories.

1.2 Interpret contingency table The contingency table doesn't seem to show much of a relationship between riding January-March and riding in bad weather days. Maybe it was a relatively mild winter in the year of data collection?

1.3 Interpret scatterplot The plot reveals a very strong positive relationship between average daily distance ridden and income. It is difficult to see any other patterns. This suggests that wealthier cyclists may be more likely to make further trips.

SQ 2: Birthdays revisited (optional bonus)

SQ2.1 Which bet? If you are willing to assume that birthdays in the class are independent (not a terrible assumption) and that birthdays are distributed randomly throughout the year (a terrible assumption, as it turns out!), you should take Bet #2. Here's a way to explain why:

Consider that 25 people can be combined into pairs $\binom{25}{2}$ ways (which you can read "as 25 choose 2"), which is equal to $\frac{25 \times 24}{2} = 300$ (and that little calculation is where those binomial coefficients I mentioned in my hint come in handy).

Generalizing the logic from part b of the textbook exercise last week problem, I have assumed that each of these possible pairings are independent and thus each one has a probability $P = (\frac{364}{365})$ of producing an *unmatched* set of birthdays.

Putting everything together, I can employ the multiplication rule from *OpenIntro* Ch. 3 and get the following:

$$P(any\ match) = 1 - P(no\ matches)$$

$$P(no\ matches) = (\frac{364}{365})^{300}$$

And I can let R do the arithmetic:

```
1-((364/365)^300)
```

```
## [1] 0.5609078
```

A fair coin flip would give you a lower probability of winning the bet.

SQ2.2 Once you have the answer above, this shouldn't be too hard. I'll start with the calculation for how many pairings exist in our seven person class:

```
choose(7, 2)
```

```
## [1] 21
```

Then I can plug that result into the formulas from SQ2.1 to calculate the probability of any shared birthdays in our class:

```
1-((364/365)^21)
```

```
## [1] 0.05598498
```

Empirical paper questions: Emotional contagion in social networks

The responses below about the Kramer et al. paper may be taken as a guide, but are intended neither to be comprehensive nor the only possible accurate responses.

EQ 1: Research questions and objectives

The paper asks: does emotional contagion occur via social networks (online)? It seeks to answer this question in human populations (i.e., the target population seems to be "people").

EQ 2: Sample and experiment design

The study uses a random sample of 689,003 Facebook accounts (drawn from the population of internal Facebook account ID numbers as of a week in January 2012). The treatment groups received a reduced portion of posts in their news feed with either positive or negative words in them respectively. The control groups had random posts removed from their news feeds. The experimental manipulation consists in a probabilistic reduction the proportion of news feed posts with either positive or negative language in them.

EQ 3: Data and variables

The unit of analysis is a Facebook account. The key variables used in the analysis are the treatment/control indicator (categorical, dichotomous) and the dependent variables: "the percentage of all words produced by a given account that was either positive or negative during the experimental period" (continuous, numeric).

EQ 4: Results

The study finds evidence of emotional contagion in soicial networks among English language Facebook users in 2012. Accounts exposed to fewer posts with positive (negative) language tended to produce few posts with positive (negative) language than accounts with randomly removed posts. The four panels of Figure 1 capture the comparisons of the treatment and control conditions by each of the two outcome measures. They show that the treatments had the anticipated effects with reduced positive posts reducing/increasing the proportion of positive/negative words in subsequent posts and reduced negative posts reducing/increasing the proportion of negative/positive words in subsequent posts.

EQ 5: Interpretation and contribution (significance)

The authors argue that the paper demonstrates evidence of emotional contagion in social networks in a particularly novel and compelling way. They underscore the scale of the study and resulting precision of the estimates that support the existence of an infrequently observed phenomenon (emotional contagion) in a context dominated by large numbers of short computer-mediated communications (which could be expected to have reduced emotional immediacy/effects).

With respect to generalization, the authors seem to base the core of their claim on the scale and scope of the Facebook user base (as of 2012). However, things get a bit iffy (despite the sample size) given the likely biases of the sample in comparison to the target population of the study. English language Facebook users have a number of attributes that differentiate them from even a broader English language speaking population in the U.S. and Facebook users and it's likely that some of these attributes systematically covary with the outcomes of interest in this study. Covariance (non-independence) between these factors and the outcomes of interest could bias the effects reported in the paper.