Homework 2

Brady Lamson

2/6/2022

Problem 1

a) Write R commands that create three vectors:

- **TrueAndMissing** containing values **TRUE** and **NA**.
- **FalseAndMissing** containing values **FALSE** and **NA**.
- **Mixed** containing values **TRUE**, **FALSE** and **NA**.

b) Apply the functions **any()** and **all()** to each of the vectors of part *a* and report the results.

```
TrueAndMissing <- c(TRUE, NA)
FalseAndMissing <- c(FALSE, NA)
Mixed <- c(TRUE, FALSE, NA)
```

```
## -----[ TrueAndMissing ]-----
## any() outputs TRUE
## all() outputs NA
## -----[ FalseAndMissing ]-----
## any() outputs NA
## all() outputs FALSE
## -----[ Mixed ]-----
## any() outputs TRUE
## all() outputs FALSE
```

Problem 2

Consider the following data on populations, illiteracy rates, and murder rates for 10 states of the United States: population in thousands, percent illiteracy, and murders per 100,000 population.

State	Population	Illiteracy	Murder
Alabama	3615	2.1	15.1
Alaska	365	1.5	11.3
Arizona	2212	1.8	7.8
Arkansas	2110	1.9	10.1
California	21198	1.1	10.3
Colorado	2541	0.7	6.8
Connecticut	3100	1.1	3.1
Delaware	579	0.9	6.2
Florida	8277	1.3	10.7
Georgia	4931	2.0	13.9

Write commands involving the comparison operators and either square brackets or the subset() function that do the following.

a) Return the names of the states whose populations are less than 2,500 (thousand).

b) Return illiteracy rates that are greater than the median illiteracy rate.

c) Return the murder rates for states whose illiteracy rate is greater than the median illiteracy rate.

```
# -----[ a ]-----
subset(df$State, pop > 2500)
```

```
## [1] "Alabama"     "California"  "Colorado"    "Connecticut" "Florida"
## [6] "Georgia"
```

```
# -----[ b ]-----
subset(df$Illiteracy, illit > median(illit))
```

```
## [1] 2.1 1.5 1.8 1.9 2.0
```

```
# -----[ c ]-----
subset(df$Murder, illit > median(illit))
```

```
## [1] 15.1 11.3  7.8 10.1 13.9
```

Problem 3

Consider again the data on murder rates from the previous exercise.

a) Write a command involving **which()** that determines the indices of the murder rates that are greater than 12.

```
which(df$Murder > 12)
```

```
## [1]  1 10
```

b) The following command does the same thing as which(murder > 12)

```
(1:10)[murder > 12]
```

```
## [1]  1 10
```

Why do you think the parentheses are included in the expression? Experiment a little.

Answer: This is a weird one. I tested out a few different things. None of them really worked according to my assumptions.

```
(1)[df$Murder > 12]
```

```
## [1]  1 NA
```

```
(4)[df$Murder > 12]
```

```
## [1]  4 NA
```

```
# Single values in parentheses seem to not do anything. Ranges work fine though!
# The below code just does some recycling,
# 11 is equivalent to the 1 index, 20 is equivalent to 10 and so on.
(1:50)[df$Murder > 12]
```

```
##  [1]  1 10 11 20 21 30 31 40 41 50
```

```
# The same does not work without parentheses though. It throws a warning
# and just returns the 1:10 vector.
1:10[df$Murder > 12]
```

```
## Warning in 1:10[df$Murder > 12]: numerical expression has 2 elements: only the
## first used
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
# I'll be frank, I cannot for the life of me figure out why the
# parentheses are so important!
```

Problem 4

For each of the following "*logical*" expressions, guess whether it will evaluate to **TRUE** or **FALSE**, then check your answer.

a) $(4 > 5 \;\&\; 8 < 9) \mid 2 > 3$

- **FALSE**

b) $4 > 5 \;\&\; (8 < 9 \mid 2 > 3)$

- **FALSE**

c) $(4 > 5 \;\&\; 2 > 3) \mid 8 < 9$

- **TRUE**

d) $(4 > 5 \;\&\; (2 > 3 \mid 8 < 9)) \mid 7 < 6$

- **FALSE**

```
## Checking my work:
## (4 > 5 & 8 < 9) | 2 > 3 outputs to FALSE
## 4 > 5 & (8 < 9 | 2 > 3) outputs to FALSE
## (4 > 5 & 2 > 3) | 8 < 9 outputs to TRUE
## (4 > 5 & (2 > 3 | 8 < 9)) | 7 < 6 outputs to FALSE
```

Problem 5

An exception to the rule that **NAs** in operators necessarily produce **NAs** is given by the **&** and **|** operators.

a) What are the values of the following code and why?

```
NA | TRUE
```

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

```
FALSE & NA
```

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

Thankfully these boolean tests can function as normal because the output here is independent of the **NA**. Regardless of what was in the place of **NA**, within reason, both of those lines of code would output the same thing.

b) Why is the output of the following code different?

```
NA | FALSE
```

```
## [1] NA
```

```
TRUE & NA
```

```
## [1] NA
```

Here we run into a problem because the output is *dependent* on whats on the opposite side of the boolean operator. We can't rely entirely on the TRUE and FALSE and, as such, we get **NA** as output because the output of these statements is unknown or, in other words, ***not available***.

Problem 6

Recall that is.na(x) indicates whether or not each element of x is a missing value (NA). Use is.na(), square brackets [ ], and ! (R's version of "not") to write one or more commands that extract all the non-missing values (non-NA's) from x. For this problem, do not assume that you know where in x the NAs are.

```
x <- c(12, 4, 8, NA, 9, NA, 7, 12, 13, NA, 10)
x[!is.na(x)]
```

```
## [1] 12  4  8  9  7 12 13 10
```

Problem 7

The ***geometric mean*** can be defined as the nth root of the product of n positive numbers $x_1, x_2, ..., x_n$, i.e.

$$\text{Geometric Mean} = (x_1 \cdot x_2 \cdot \ ... \ \cdot x_n)^{\frac{1}{n}}$$

Write a function **gm()** that takes a vector argument x containing positive numbers and returns their geometric mean, but gives appropriate feedback when the input is not numeric or contains non-positive values. Test your **gm()** function by passing it:

a) A numeric vector of positive values.

b) A vector containing one or more non-positive values

c) A character vector.

```
geometric_mean <- function(vector) {

    if(!is.numeric(vector) | any(vector <= 0)) {

        cat(
            "-----[ ERROR ]-----
              \nInput vector must be a numeric vector with only positive values.\n"
        )

        stop()
    }

    return_value <- prod(vector)^(1 / length(vector)) %>%
        round(digits = 3)

    return(
        glue::glue("The geometric mean of this vector is approximately {return_value}")
    )
}
```

```
# -----[ a ]-----
geometric_mean(c(1:10))
```

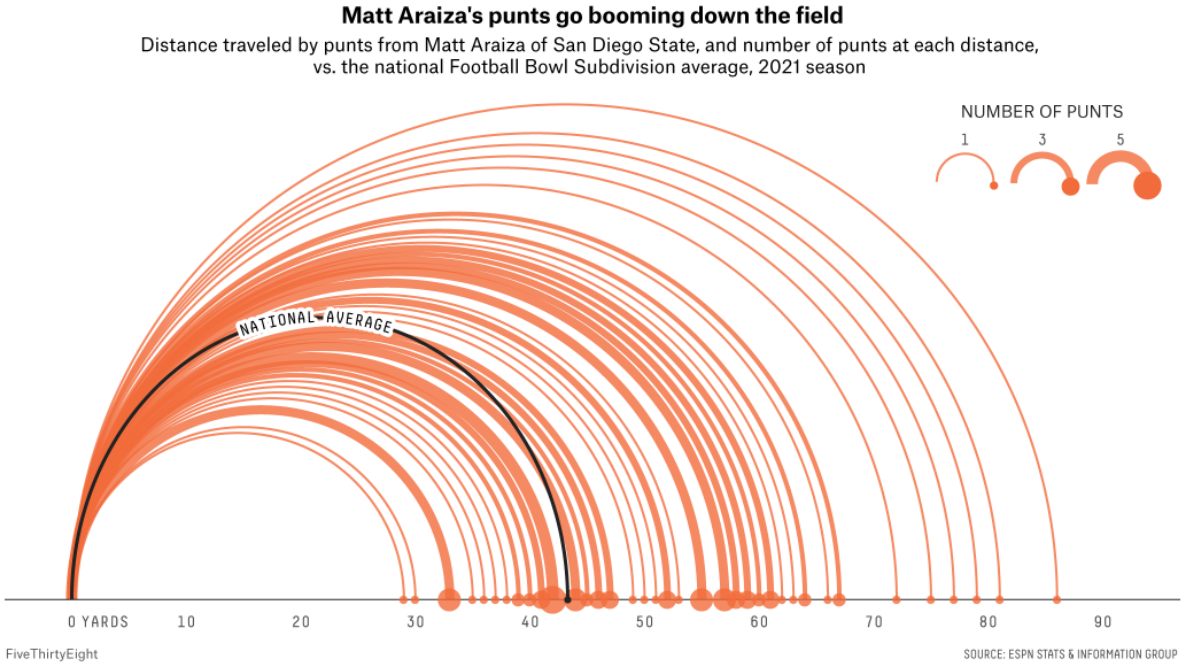
```
## The geometric mean of this vector is approximately 4.529
```

```
# -----[ b ]-----
# geometric_mean(c(5:-5))
# This code of course throws the provided error and will not compile.
# -----[ Error ]-----
#[1] "Input vector must be a numeric vector with only positive values."
#Error in geometric_mean(c(5:-5)) :
```

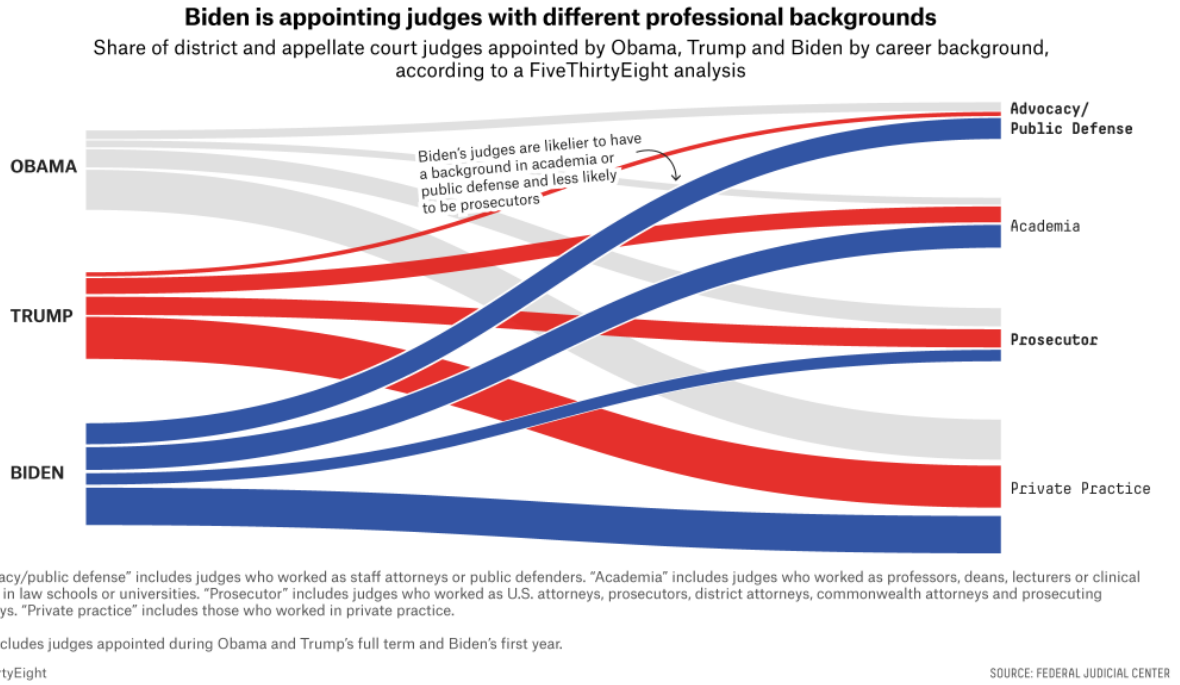
```
# -----[ b ]-----
# geometric_mean(c("dishwasher, laundry, dog, cat"))
# This code of course throws the provided error and will not compile.
# -----[ Error ]-----
# [1] "Input vector must be a numeric vector with only positive values."
#Error in geometric_mean(c("dishwasher, laundry, dog, cat")) :
```

Textbook Chapter 2 Problem 3

Find two graphs published in a newspaper or on the internet in the last two years, one you find compelling and the other that you don't. Discuss aspects of the graphs work and don't work respectively.



This visualization I like quite a lot. I particularly enjoy how the kick is displayed as an arc that grows as the distance increases. It's a fun way to help visualize the data *as* the punts themselves which is a fantastic use of position and length. It's neat and has a touch of artistic flair. They also overall kept it visually simple which I think works as not much information is really on display here. I do have a slight issue though and that's with the size of the circles on the axis. The book states that area isn't a fantastic way to demonstrate differences and I think that's on display here. Since there are only 3 categories it isn't *too* bad, but it is a little difficult to tell the differences in circle size. The axis is also quite visually busy and it can be hard to make out individual data points.



I mentioned area being a bad strategy previously and, at least with the last visualization it wasn't the most important part. Here though it is the only information we get! There is no way for me to differentiate between the sizes of these lines with any precision whatsoever. The lines crossing over each other also does this graphic no favors. It's somehow so visually busy and confusing despite having little information to warrant it. The overall visual presentation here is a total mess. This data would be better served as a simple bar chart.