Sugary Cereal Vectors

Kaitlyn Watson-Group 9

M2 ICA1

Introduction

In this assignment we will use the cereal dataset. The .Rmd and .csv file is uploaded in the Module 2 ICA folder (M2ICA1).First, save the Rmd file in a folder with the rest of your course work. Put the file `cereal.csv` in the same folder. Please check whether the data and .rmd file are saved in the same working directory. Go to Session > Set Working Directory > To Source File Location. Now you may run the below code that will load the data.

```
cereal <- read.csv("cereal.csv")
attach(cereal)
head(cereal)
```

```
                      name mfr type calories protein fat sodium fiber carbo
1                100% Bran   N    C       70       4   1    130  10.0   5.0
2         100% Natural Bran   Q    C      120       3   5     15   2.0   8.0
3                  All-Bran   K    C       70       4   1    260   9.0   7.0
4 All-Bran with Extra Fiber   K    C       50       4   0    140  14.0   8.0
5            Almond Delight   R    C      110       2   2    200   1.0  14.0
6   Apple Cinnamon Cheerios   G    C      110       2   2    180   1.5  10.5
  sugars potass vitamins shelf weight cups   rating  X
1      6    280       25     3      1 0.33 68.40297 NA
2      8    135        0     3      1 1.00 33.98368 NA
3      5    320       25     3      1 0.33 59.42551 NA
4      0    330       25     3      1 0.50 93.70491 NA
5      8     -1       25     3      1 0.75 34.38484 NA
6     10     70       25     1      1 0.75 29.50954 NA
```

The object `cereal` is a data frame with 77 rows and 17 columns. The 77 cereals have information recordered about their nutrients on a per serving basis. Information is also given about the number cups in one serving. Full details on the variable names are given below.

- **name**: name of cereal
- **mfr**: manufacturer of cereal
  - A = American Home Food Products;
  - G = General Mills
  - K = Kelloggs
  - N = Nabisco
  - P = Post
  - Q = Quaker Oats

  - R = Ralston Purina

- **type**: (C) cold, (H) hot
- **calories**: calories per serving
- **protein**: grams of protein
- **fat**: grams of fat
- **sodium**: milligrams of sodium
- **fiber**: grams of dietary fiber
- **carbo**: grams of complex carbohydrates
- **sugars**: grams of sugars
- **potass**: milligrams of potassium
- **vitamins**: vitamins and minerals - 0, 25, or 100, indicating the -typical percentage of FDA recommended
- **shelf**: display shelf (1, 2, or 3, counting from the floor)
- **weight**: weight in ounces of one serving
- **cups**: number of cups in one serving
- **rating**: a rating of the cereals

The R command `attach(cereal)` allows you to access each vector in the data frame by simply typing the variable's name in R (you need not use the format <data.frame>$).

Operators

Assignment operator

Objects and variables are created with the **assignment operator**, `<-` or `=`. Remeber that in RStudio you can use the shortcut `Alt + -` to create the `<-` assignment operator. The shortcut includes convenient spacing before and after the assignment operator. While the equal sign will work, it is best practice to stick to using `<-`.

Let's look at the assignment operator in action.

Compute the protein to fat ratio per serving and name the new variable protein.to.fat.Check the first 6 entries of the new variable. Hint: use the **head** function.

```
headc<- head(cereal)
protein.to.fat <- headc[, 5]/ headc[, 6]
protein.to.fat
```

```
[1] 4.0 0.6 4.0 Inf 1.0 1.0
```

Calculate the protein to carbohydrate ratio.

```
headc<- head(cereal)
protein.to.carbohydrate <- headc[, 5]/ headc[, 9 ]
protein.to.carbohydrate
```

```
[1] 0.8000000 0.3750000 0.5714286 0.5000000 0.1428571 0.1904762
```

Try to be descriptive and concise with your variable names. For example, `protein.to.fat` is better than `ptf`.

Comparison operators

Comparisons are binary operators; they take two objects and give a boolean (TRUE or FALSE) response.

Command	Description
`>`	greater than
`<`	less than
`>=`	greater than or equal to
`<=`	less than or equal to
`==`	equal to (recall that `=` is for assignment and not checking equality)
`!=`	not equal to
`&`	and (ex: `(5 > 7) & (6*7 == 42)` will return the value FALSE)
`|`	or (ex: `(5 > 7) | (6*7 == 42)` will return the value TRUE)

Exercises

1. Give an example using each of the comparison operators.

```
cereal[, 5] > cereal[, 6]
```

```
 [1]  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[13] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
[25]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
[37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[73]  TRUE FALSE  TRUE  TRUE  TRUE
```

```
cereal [, 5] < cereal [, 6]
```

```
 [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
[13]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[73] FALSE FALSE FALSE FALSE FALSE
```

```
cereal [, 5] >= cereal[, 6]
```

```
 [1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[13] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[73]  TRUE  TRUE  TRUE  TRUE  TRUE
```

```
cereal [, 5] <= cereal [, 5]
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[76] TRUE TRUE
```

```
cereal [, 5] == cereal [, 5]
```

```
 [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[16] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[31] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[46] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[61] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[76] TRUE TRUE
```

```
cereal [, 5] != cereal [, 5]
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[73] FALSE FALSE FALSE FALSE FALSE
```

```
cereal[, 5] > cereal[, 6] & cereal [, 5] >= cereal[, 6]
```

```
 [1]  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[13] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
[25]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
[37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[73]  TRUE FALSE  TRUE  TRUE  TRUE
```

```
cereal[, 5] > cereal[, 6] | cereal [, 5] >= cereal[, 6]
```

```
 [1]  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
[13] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
[37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[73]  TRUE  TRUE  TRUE  TRUE  TRUE
```

2. How many cereals have more than 100 calories per serving?

```
cereal [, 4] >100
```

```
 [1] FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
[13]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE
[25]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE
[37]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
[49]  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
[61] FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE
[73]  TRUE  TRUE FALSE FALSE  TRUE
```

```
 # 47 cereals have more than 100 calories
```

3. How many cereals have 0 grams of sugar per serving?

```
cereal [, 10]==0
```

```
 [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[49] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
[61] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
[73] FALSE FALSE FALSE FALSE FALSE
```

```
# 7 cereals have 0 grams of sugar
```

NA, NaN, Inf, etc.

For each of the following expressions, first guess what the output will be when the expression is evaluated in R, then enter the expression in R.

a. 1/0 = INF
b. 0/0 = INF
c. 1/0 + 5 = INF
d. 1/0 - 1/0 = INF
e. -2^(10000) = INF
f. exp(Inf)= INF
g. exp(-Inf)= 0
h. 0^0= 1
i. Inf^0= 1
j. 0^Inf= 0
k. log(0) = Err
l. log(-Inf)= Err

```
1/0
```

```
[1] Inf
```

```
0/0
```

```
[1] NaN
```

```
1/0 + 5
```

```
[1] Inf
```

```
1/0 - 1/0
```

```
[1] NaN
```

```
-2^(10000)
```

```
[1] -Inf
```

```
exp(Inf)
```

```
[1] Inf
```

```
exp(-Inf)
```

```
[1] 0
```

```
0^0
```

```
[1] 1
```

```
Inf^0
```

```
[1] 1
```

```
log(0)
```

```
[1] -Inf
```

```
log(-Inf)
```

```
[1] NaN
```

Floating point arithmetic

R, as does most software, uses floating point arithmetic, which is not the same as the arithmetic we learn, because computers cannot represent all numbers exactly. For example, while testing for equality of numbers,this has an important consequence.

Here are a few examples.

First, would you expect `0.2 == 0.6/3` to return TRUE or FALSE? True Test it out.

```r
0.2 == 0.6/3
```

```
[1] FALSE
```

Explain why you are getting the answer that you are getting.

```r
#There is not enough accuracy in the precision presented in the answer above and therefore it cannot co
```

Next consider the vector `point3` created below.

```r
point3 = c(0.3, 0.4 - 0.1, 0.5 - 0.2, 0.6 - 0.3, 0.7 - 0.4)
point3
```

```
[1] 0.3 0.3 0.3 0.3 0.3
```

Vectors

Vector creation

The function `c` combines its arguments to form a vector. All arguments are coerced to a common type which is the type of the returned value, and all attributes except names are removed.

Create a vector of school names `MSU, UM, OSU, PSU` and numbers `50085, 43625, 58322, 45518`.

```r
c("MSU", "UM", "OSU", "PSU", 50085, 43625, 58322, 45518)
```

```
[1] "MSU"   "UM"    "OSU"   "PSU"   "50085" "43625" "58322" "45518"
```

Vector subsetting

Vectors can be subset by position or name (when applicable). For example, consider the vector `x`, where

```r
x <- letters[1:10]
x
```

```
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```r
x[5]
```

```
[1] "e"
```

```r
x[c(1, 5, 9)]
```

```
[1] "a" "e" "i"
```

```
x[-c(4, 6, 10)]
```

```
[1] "a" "b" "c" "e" "g" "h" "i"
```

```
x[11]
```

```
[1] NA
```

As another example,

```
grades <- c(99, 85, 89, 92)
names(grades) <- c("STT", "CMSE", "MTH", "STT")
grades
```

```
 STT CMSE  MTH  STT
  99   85   89   92
```

```
grades["STT"]
```

```
STT
 99
```

```
grades["CMSE"]
```

```
CMSE
  85
```

```
grades[4] <- 82
grades[4]
```

```
STT
 82
```

Some built-in functions that work on vectors

Assume that `vec` is a vector of appropriate variable type.

Command	Description
`sum(vec)`	sums up all the elements of `vec`
`mean(vec)`	mean of `vec`
`median(vec)`	median of `vec`
`min(vec), max(vec)`	the smallest or largest element of `vec`
`sd(vec), var(vec)`	the standard deviation and variance of `vec`
`length(vec)`	the number of components in `vec`
`sort(vec)`	returns the `vec` in ascending or descending order
`order(vec)`	returns the index that sorts `vec`
`unique(vec)`	lists the unique elements of `vec`

Command	Description
`summary(vec)`	computes the five-number summary

Exercises

a. Let stat.grade be vector with elements (78,87,65,98,95,99,70,85,75, 80). Input the vector and give the results for the above mentioned commands.

```
stat.grade <- c(78,87,65,98,95,99,70,85,75, 80)
sum(stat.grade)
```

```
[1] 832
```

```
mean(stat.grade)
```

```
[1] 83.2
```

```
median(stat.grade)
```

```
[1] 82.5
```

```
min(stat.grade)
```

```
[1] 65
```

```
max(stat.grade)
```

```
[1] 99
```

```
sd(stat.grade)
```

```
[1] 11.71703
```

```
var(stat.grade)
```

```
[1] 137.2889
```

```
length(stat.grade)
```

```
[1] 10
```

```
sort(stat.grade)
```

```
 [1] 65 70 75 78 80 85 87 95 98 99
```

```
order(stat.grade)
```

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

```
unique(stat.grade)
```

```
 [1] 78 87 65 98 95 99 70 85 75 80
```

```
summary(stat.grade)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  65.00   75.75   82.50   83.20   93.00   99.00
```

b. Comment on what is the difference between the sort and order outputs.

Sort represents the order placement of the original value from least to greatest. Order shows the original value positions

c. Below is a helpful example on subsetting a vector with a logical vector.

```
x <- c(1, 4, -5, sqrt(3), exp(2))
y <- c(T, F, T, T, F)
x[y]
```

```
[1]  1.000000 -5.000000  1.732051
```

Comment (1-2 sentences) on what `x[y]`did?

X[y] pulled out only the true values of the vector x.

d. Now going back to the `cereal` dataset.Use the dataset to answer the questions that follow. You may find some of the above functions helpful. Use a single code chunk to answer each question.

1. How many cereals are in the data set `cereal`?

```
nrow(cereal)
```

```
[1] 77
```

2. What is the average number of grams of sugar per serving among all cereals?

```
mean(cereal[,10])
```

```
[1] 6.922078
```

3. What are the three largest grams of sugar per serving values?

```
sort (cereal[, 10]) #14, 15, 15 are the largest values grams per sugar.
```

```
 [1] -1  0  0  0  0  0  0  0  1  2  2  2  3  3  3  3  3  3  3  3  3  3  3  3
[26]  4  5  5  5  5  5  6  6  6  6  6  6  6  7  7  7  7  8  8  8  8  8  9  9  9
[51]  9 10 10 10 10 10 11 11 11 11 11 12 12 12 12 12 12 12 13 13 13 13 14 14 14
[76] 15 15
```

4. What are the five smallest grams of sugar per serving values?

```
sort (cereal[, 10])  #-1, 0, 0, 0, 0 are the smallest grams of sugar per serving (Note: -1 grams of sug
```

```
 [1] -1  0  0  0  0  0  0  0  1  2  2  2  3  3  3  3  3  3  3  3  3  3  3  3
[26]  4  5  5  5  5  5  6  6  6  6  6  6  6  7  7  7  7  8  8  8  8  8  9  9  9
[51]  9 10 10 10 10 10 11 11 11 11 11 12 12 12 12 12 12 12 13 13 13 13 14 14 14
[76] 15 15
```

5. Which four cereals have the most grams of sugar per serving?

```
order(cereal[, 10])
```

```
 [1] 58  4 21 55 56 64 65 66 12 17 51 62 16 22 34 41 44 54 63 68 70 72 73 75 76
[26] 35  3 10 24 33 69  1  9 39 42 48 57 61 14 20 27 50  2  5  8 60 77 13 32 40
[51] 49  6 23 28 37 52 26 36 38 45 46 11 18 29 30 43 59 74 15 19 25 47  7 53 71
[76] 31 67
```

```
cereal [58, 1]
```

```
[1] "Quaker Oatmeal"
```

```
cereal [4, 1]
```

```
[1] "All-Bran with Extra Fiber"
```

```
cereal [21, 1]
```

```
[1] "Cream of Wheat (Quick)"
```

```
cereal [55, 1]
```

```
[1] "Puffed Rice"
```

```
#  Quakes, All Bran, Cream of Wheat (Quick), Puffed
```

6. How many cereals have more than 4 grams of protein per serving and less than 3 grams of sugar per serving?

```
cereal$protein>4&cereal$sugars<3
```

```
 [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
[61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[73] FALSE FALSE FALSE FALSE FALSE
```

```
# 2 Cereals have more than 4 grams of protein and less than 3 grams of sugar
```

Where can I find the sugary cereals?

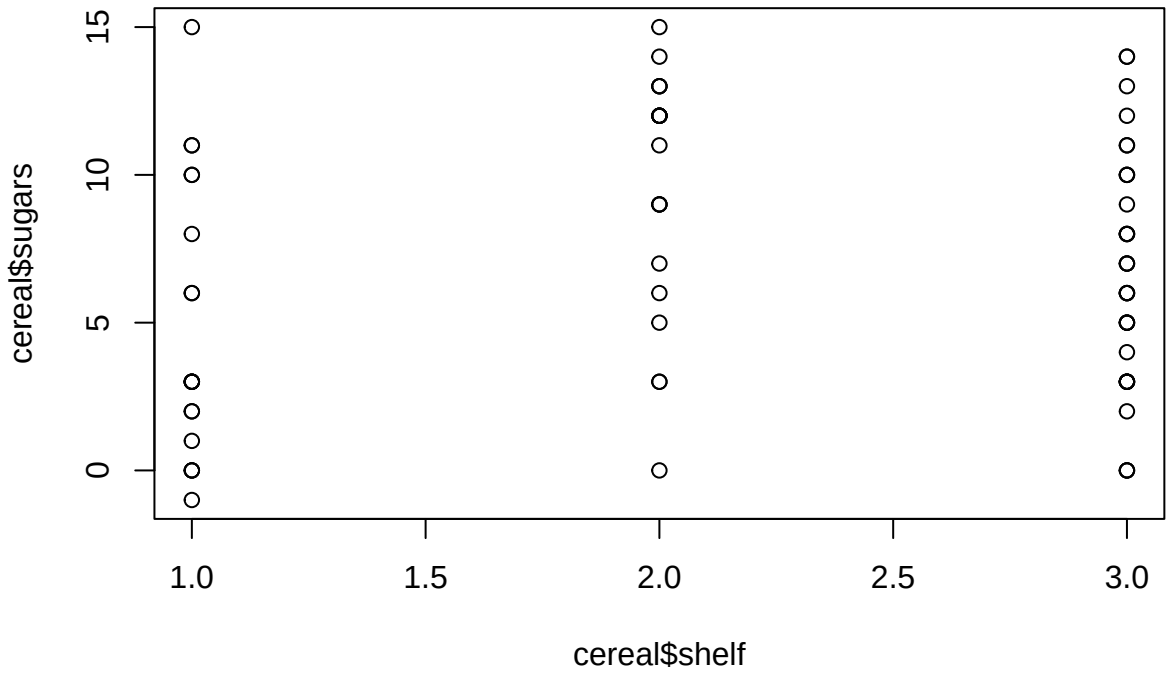
7. What is the mean amount of grams of sugar per serving for cereals on shelf 1, 2, and 3, respectively?

```
mean(cereal$sugars)
```

```
[1] 6.922078
```

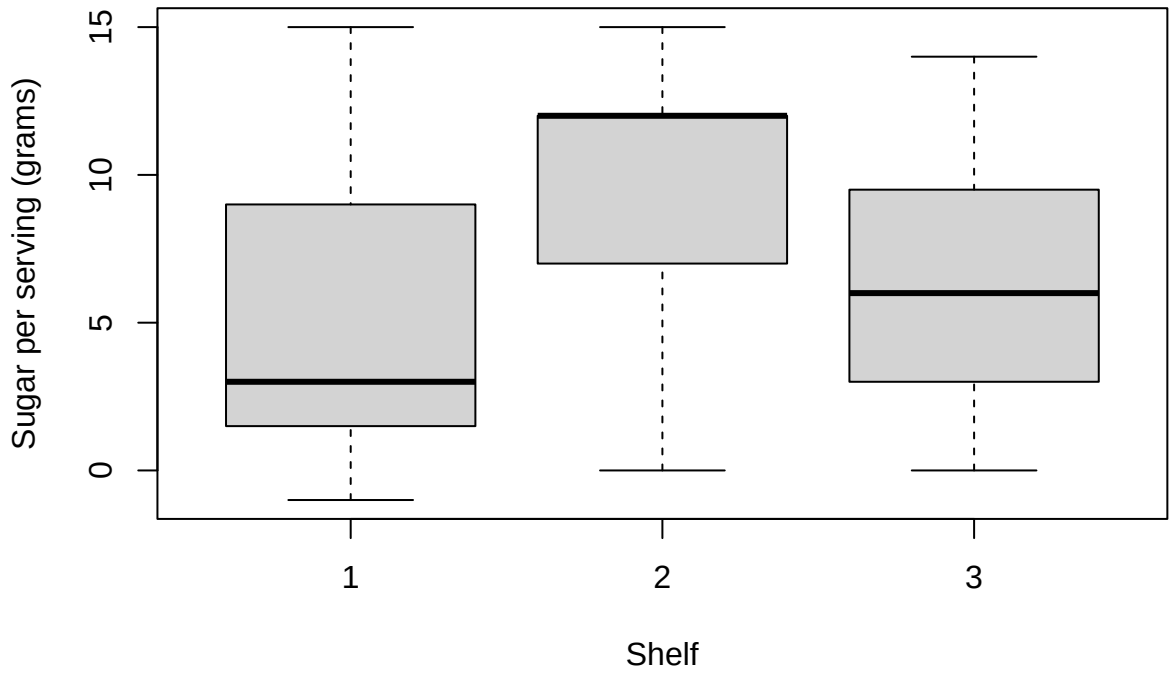
8. Plot (using the plot()) the relationship between shelf and grams of sugar per serving.

```
plot(cereal$shelf, cereal$sugars)
```



Comparison box plots would be better.

```
boxplot(sugars ~ shelf, xlab = "Shelf", ylab = "Sugar per serving (grams)")
```



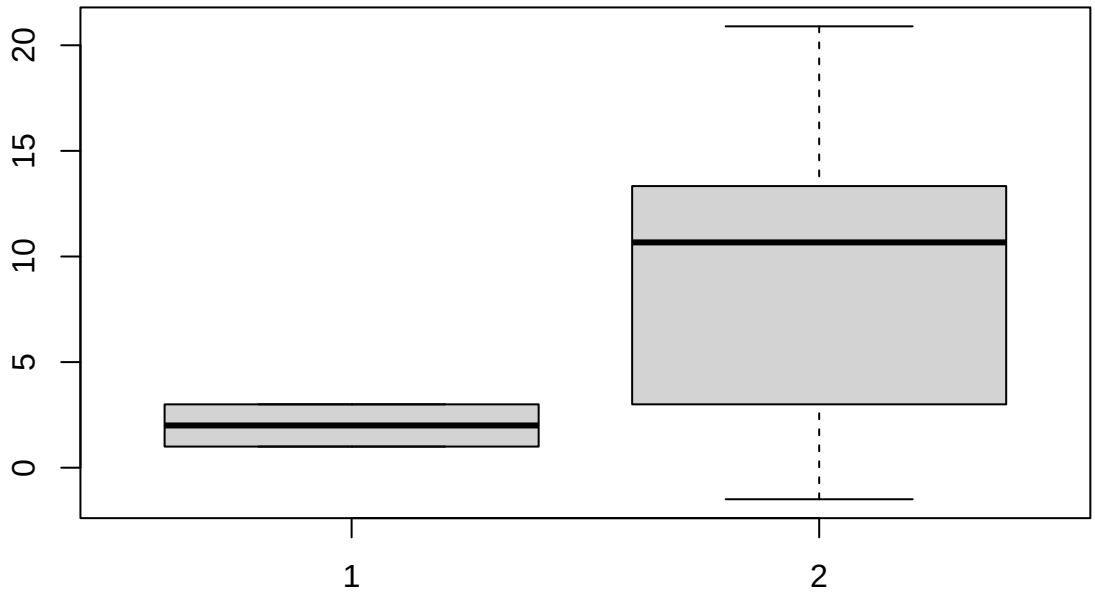
Comment on which plot is giving more information. Justify your answer in 2-3 sentences.

I would say a box plot provides more information as it shows the more visualization and a clearer distinction between the 3 shelves. It also provides more information in regards to the mean.

9. Do you see any issue with analyzing and comparing the data on a per serving basis? Think of and propose a better way given the available data.(Hint: Think of how can the data be normalized/ standardized.)

The data could be normalized by using cups instead of a standard serving size.

```
sugar.per.cup <- cereal$sugar/cereal$cups
boxplot(cereal$shelf, sugar.per.cup)
```



10. Redo the above boxplot using your new normalized metric from 9.

Do these results align with your expectations as to where you would find the sugary cereals?

Yes, the sugary cereals are located more on Shelf 2 which aligns with my expectations.

References

1. https://perso.telecom-paristech.fr/eagan/class/igr204/datasets