Week 1 - Homework

STAT 420, Summer 2022, Ilya Andreev, iandre3

Exercise 1 (Subsetting and Statistics)

For this exercise, we will use the `msleep` dataset from the `ggplot2` package.

(a) Install and load the `ggplot2` package. **Do not** include the installation command in your `.Rmd` file. (If you do it will install the package every time you knit your file.) **Do** include the command to load the package into your environment.

(b) Note that this dataset is technically a `tibble`, not a data frame. How many observations are in this dataset? How many variables? What are the observations in this dataset?

(c) What is the mean hours of REM sleep of individuals in this dataset?

(d) What is the standard deviation of brain weight of individuals in this dataset?

(e) Which observation (provide the `name`) in this dataset gets the most REM sleep?

(f) What is the average bodyweight of carnivores in this dataset?

```r
library(ggplot2)
observations = nrow(msleep)
names = colnames(msleep)
variables = length(msleep)
mean_hours_rem = mean(msleep$sleep_rem, na.rm=TRUE)
sd_brain_weight = sd(msleep$brainwt, na.rm=TRUE)
name_most_rem_sleep = msleep[which.max(msleep$sleep_rem),]$name
carnivore_mean_bodyweight =
  mean(msleep[is.na(msleep$vore) == 0 & msleep$vore == "carni", ]$bodywt)
```

a) Done.
b) Number of observations: 83, variables: 11, observations: name, genus, vore, order, conservation, sleep_total, sleep_rem, sleep_cycle, awake, brainwt, bodywt.
c) Mean hours of REM sleep: 1.8754098.
d) Standard deviation of brain weight: 0.9764137.
e) Name with most REM sleep: Thick-tailed opposum.
f) Average body weight of carnivores 90.7511053.

Exercise 2 (Plotting)

For this exercise, we will use the `birthwt` dataset from the `MASS` package.

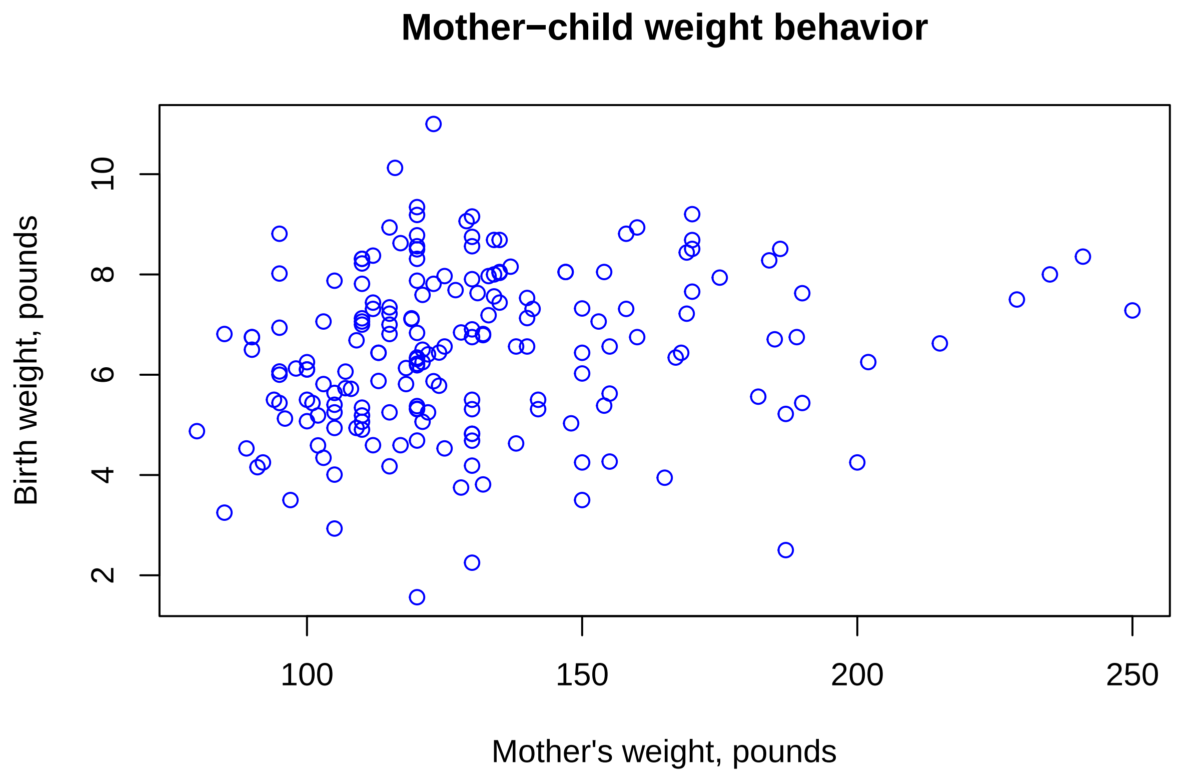
(a) Note that this dataset is a data frame and all of the variables are numeric. How many observations are in this dataset? How many variables? What are the observations in this dataset?

(b) Create a scatter plot of birth weight (y-axis) vs mother's weight before pregnancy (x-axis). Use a non-default color for the points. (Also, be sure to give the plot a title and label the axes appropriately.) Based on the scatter plot, does there seem to be a relationship between the two variables? Briefly explain.

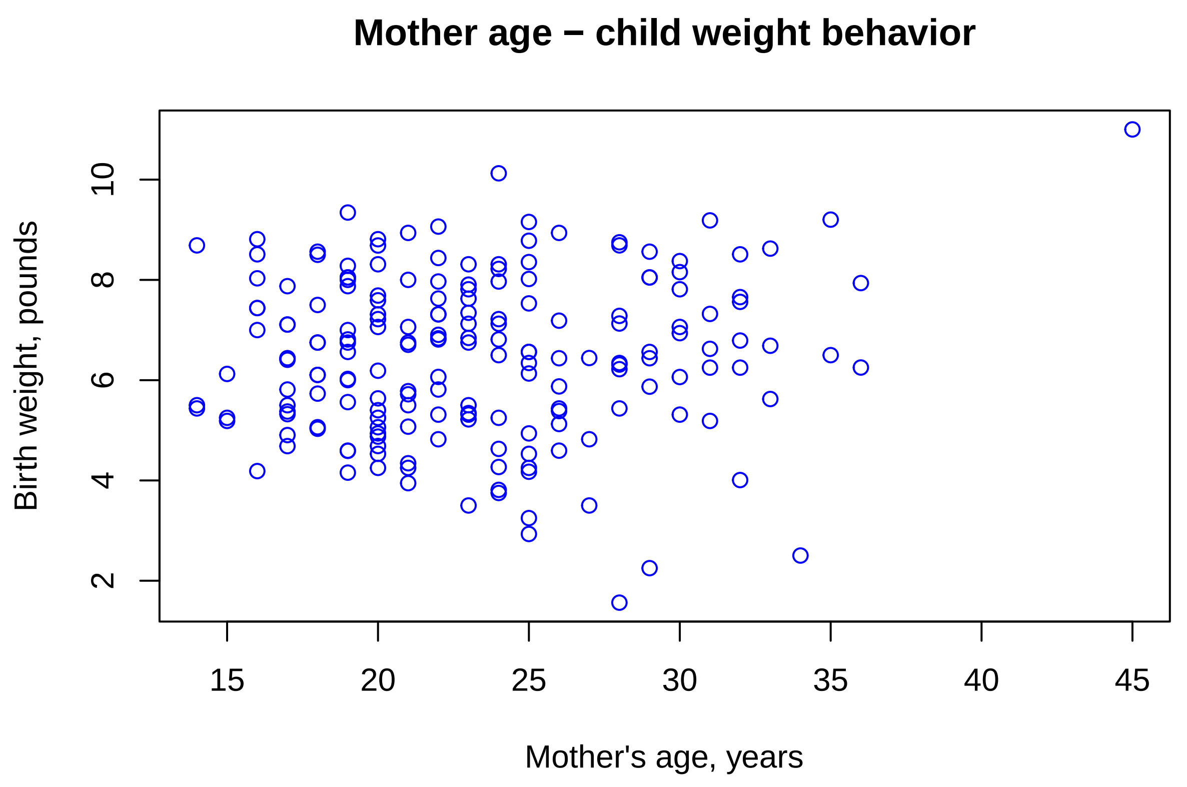
(c) Create a scatter plot of birth weight (y-axis) vs mother's age (x-axis). Use a non-default color for the points. (Also, be sure to give the plot a title and label the axes appropriately.) Based on the scatter plot, does there seem to be a relationship between the two variables? Briefly explain.

(d) Create side-by-side boxplots for birth weight grouped by smoking status. Use non-default colors for the plot. (Also, be sure to give the plot a title and label the axes appropriately.) Based on the boxplot, does there seem to be a difference in birth weight for mothers who smoked? Briefly explain.

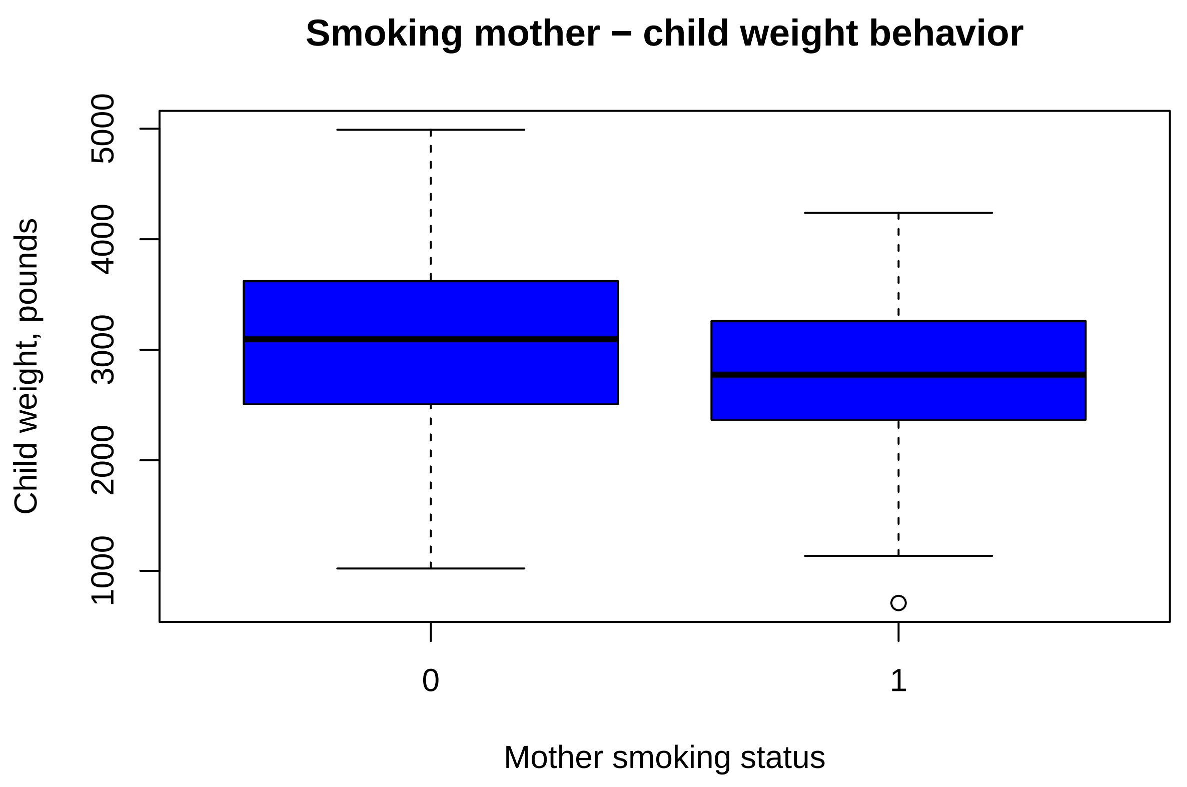
```
library(MASS)
observations = nrow(birthwt)
names = colnames(birthwt)
variables = length(birthwt)
plot(birthwt$lwt,
     birthwt$bwt * 0.00220462262185,
     xlab="Mother's weight, pounds",
     ylab="Birth weight, pounds",
     main="Mother-child weight behavior",
     col="blue")
```



```
plot(birthwt$age,
     birthwt$bwt * 0.00220462262185,
     xlab="Mother's age, years",
     ylab="Birth weight, pounds",
     main="Mother age - child weight behavior",
     col="blue")
```



```
boxplot(bwt ~ smoke,
        data=birthwt,
        main="Smoking mother - child weight behavior",
        xlab="Mother smoking status",
        ylab="Child weight, pounds", col="blue")
```



a) Number of observations: 189, variables: 10, observations: low, age, lwt, race, smoke, ptl, ht, ui, ftv, bwt.
b) There seem to be a linear relationship between the mother's and child's weight.
c) There seem to be an inverse relationship between the mother's age and child's weight.
d) There seem to be a relationship where children of non-smoking mothers are born on average heavier than those of smoking mothers.

Exercise 3 (Importing Data, More Plotting)

For this exercise we will use the data stored in `nutrition-2018.csv`. It contains the nutritional values per serving size for a large variety of foods as calculated by the USDA in 2018. It is a cleaned version totaling 5956 observations and is current as of April 2018.

The variables in the dataset are:

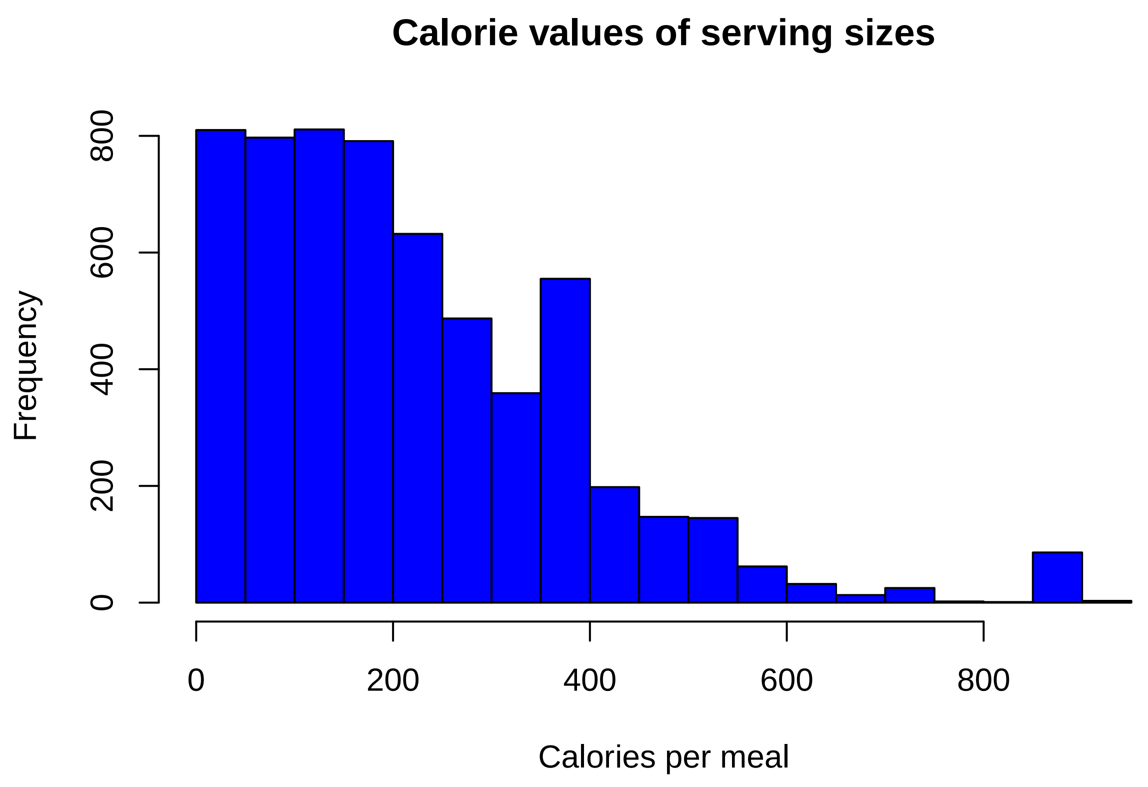
- `ID`
- `Desc` - short description of food
- `Water` - in grams
- `Calories` - in kcal
- `Protein` - in grams
- `Fat` - in grams
- `Carbs` - carbohydrates, in grams
- `Fiber` - in grams
- `Sugar` - in grams
- `Calcium` - in milligrams
- `Potassium` - in milligrams
- `Sodium` - in milligrams
- `VitaminC` - vitamin C, in milligrams
- `Chol` - cholesterol, in milligrams
- `Portion` - description of standard serving size used in analysis

(a) Create a histogram of `Calories`. Do not modify `R`'s default bin selection. Make the plot presentable. Describe the shape of the histogram. Do you notice anything unusual?

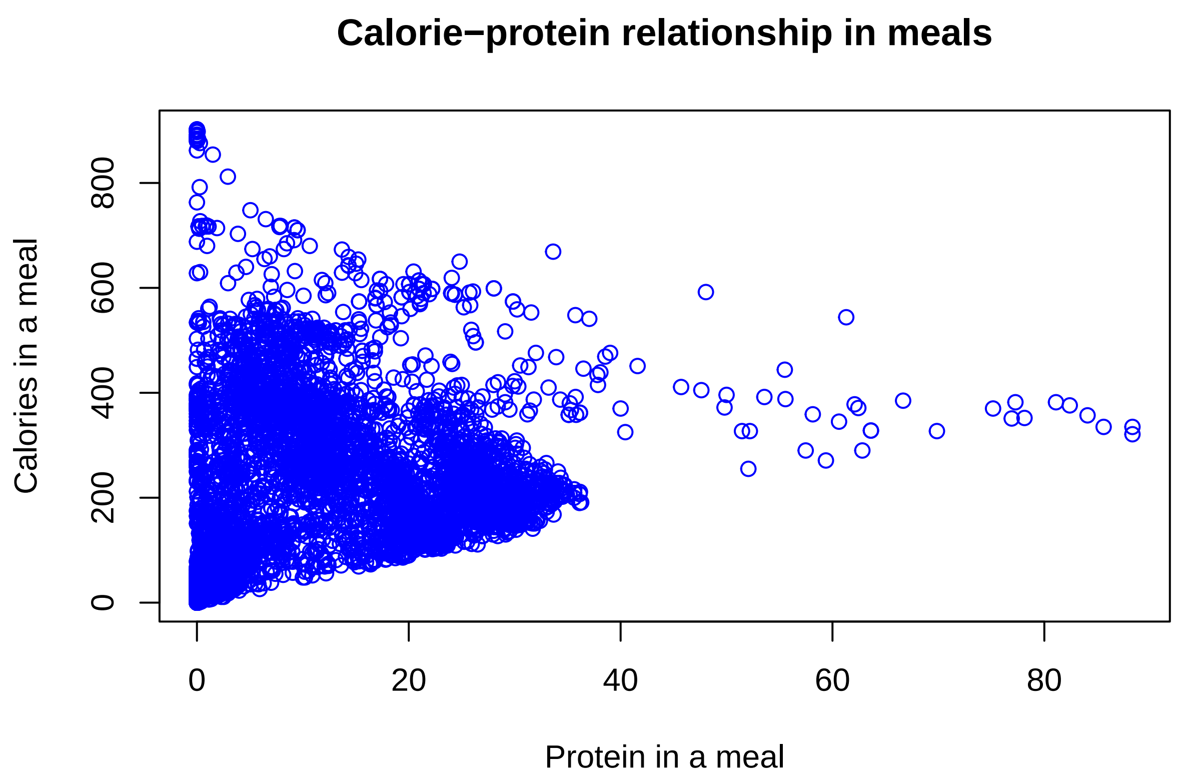
(b) Create a scatter plot of calories (y-axis) vs protein (x-axis). Make the plot presentable. Do you notice any trends? Do you think that knowing only the protein content of a food, you could make a good prediction of the calories in the food?

(c) Create a scatter plot of `Calories` (y-axis) vs `4 * Protein + 4 * Carbs + 9 * Fat` (x-axis). Make the plot presentable. You will either need to add a new variable to the data frame, or use the `I()` function in your formula in the call to `plot()`. If you are at all familiar with nutrition, you may realize that this formula calculates the calorie count based on the protein, carbohydrate, and fat values. You'd expect then that the result here is a straight line. Is it? If not, can you think of any reasons why it is not?

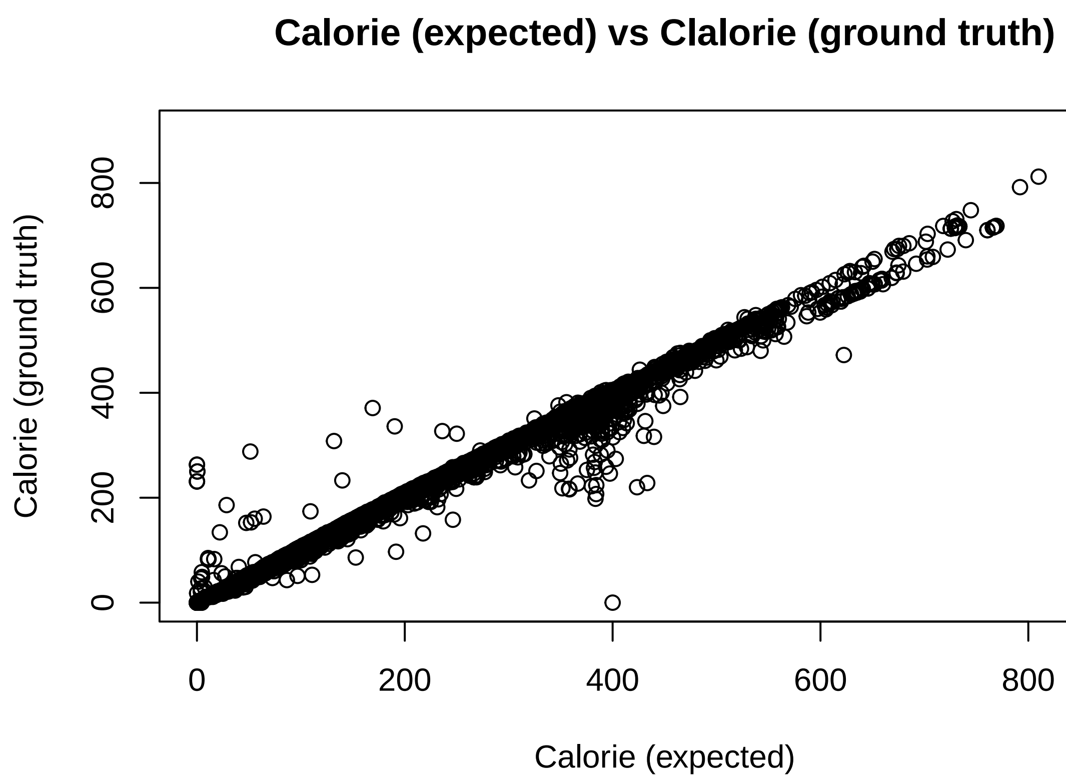
```r
data <- read.csv("nutrition-2018.csv", header=TRUE)
hist(
  data$Calories,
  main="Calorie values of serving sizes",
  xlab="Calories per meal",
  col="blue")
```



```
plot(
  data$Protein,
  data$Calories,
  main="Calorie-protein relationship in meals",
  xlab="Protein in a meal",
  ylab="Calories in a meal",
  col="blue")
```



```
plot(
  4 * data$Protein + 4 * data$Carbs + 9 * data$Fat,
  data$Calories,
  main="Calorie (expected) vs Clalorie (ground truth)",
  xlab="Calorie (expected)",
  ylab="Calorie (ground truth)")
```



a) The more calories per meal, the fewer meals fit the criteria. The only exception is the mid-range of caloric value of meals, where an unexpectedly high number of meals have around 400 calories.
b) There seems to be a category of food (meat? unlikely – in that categories protein and calories are inversely related) where protein is a strong predictor of calories. But for most meals protein is a poor predictor of calories.
c) The expected and factual calorie values are almost always equal. The outliers can be attributed to measurement errors.

Exercise 4 (Writing and Using Functions)

For each of the following parts, use the following vectors:

```
a = 1:10
b = 10:1
c = rep(1, times = 10)
d = 2 ^ (1:10)
```

(a) Write a function called `sum_of_squares`.

- Arguments:
  - A vector of numeric data `x`
- Output:
  - The sum of the squares of the elements of the vector $\sum_{i=1}^{n} x_i^2$

Provide your function, as well as the result of running the following code:

```
sum_of_squares = function(x) {
  sum(x * x)
}
```

```r
sum_of_squares(x = a)
```

```
## [1] 385
```

```r
sum_of_squares(x = c(c, d))
```

```
## [1] 1398110
```

(b) Using only your function `sum_of_squares()`, `mean()`, `sqrt()`, and basic math operations such as `+` and `-`, calculate

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - 0)^2}$$

where the x vector is `d`.

```r
sqrt(sum_of_squares(d - 0) / length(d))
```

```
## [1] 373.9118
```

(c) Using only your function `sum_of_squares()`, `mean()`, `sqrt()`, and basic math operations such as `+` and `-`, calculate

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$$

where the x vector is `a` and the y vector is `b`.

```r
sqrt(sum_of_squares(a - b) / length(a))
```

```
## [1] 5.744563
```

Exercise 5 (More Writing and Using Functions)

For each of the following parts, use the following vectors:

```r
set.seed(42)
x = 1:100
y = rnorm(1000)
z = runif(150, min = 0, max = 1)
```

(a) Write a function called `list_extreme_values`.

- Arguments:
  - A vector of numeric data `x`
  - A positive constant, `k`, with a default value of `2`
- Output:
  - A list with two elements:
    - `small`, a vector of elements of `x` that are k sample standard deviations less than the sample mean. That is, the observations that are smaller than $\bar{x} - k \cdot s$.
    - `large`, a vector of elements of `x` that are k sample standard deviations greater than the sample mean. That is, the observations that are larger than $\bar{x} + k \cdot s$.

Provide your function, as well as the result of running the following code:

```
list_extreme_values = function(x, k = 2) {
  small = x[x < mean(x) - k * sd(x)]
  large = x[x > mean(x) + k * sd(x)]
  list(small, large)
}
```

```
list_extreme_values(x = x, k = 1)
```

```
## [[1]]
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##
## [[2]]
##  [1]  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98
## [20]  99 100
```

```
list_extreme_values(x = y, k = 3)
```

```
## [[1]]
## [1] -3.371739
##
## [[2]]
## [1] 3.229069 3.211199 3.495304
```

```
list_extreme_values(x = y, k = 2)
```

```
## [[1]]
##  [1] -2.656455 -2.440467 -2.414208 -2.993090 -2.699930 -2.113200 -2.188835
##  [8] -2.071388 -2.138368 -2.461335 -2.170247 -3.017933 -2.192786 -2.253132
## [15] -2.277778 -2.292971 -2.206485 -2.553825 -2.082814 -2.958780 -2.136025
## [22] -2.183149 -3.371739
##
## [[2]]
##  [1] 2.018424 2.286645 2.701891 2.059539 2.036972 2.049961 2.459594 2.212055
##  [9] 2.422163 2.019891 2.965865 2.098031 2.241904 2.041313 3.229069 2.223534
## [17] 3.211199 2.623495 2.727196 2.178668 3.495304
```

```
list_extreme_values(x = z, k = 1.5)
```

```
## [[1]]
## [1] 0.001703130 0.077464589 0.047054933 0.060877148 0.009629518 0.004321658
## [7] 0.028495955 0.005327612 0.041129370
##
## [[2]]
##  [1] 0.9899656 0.9521815 0.9741261 0.9474009 0.9586979 0.9756436 0.9954564
##  [8] 0.9517322 0.9342643 0.9310075
```

(b) Using only your function `list_extreme_values()`, `mean()`, and basic list operations, calculate the mean of observations that are greater than 1.5 standard deviation above the mean in the vector `y`.

```
mean(list_extreme_values(x = y, k = 1.5)[[2]])
```

```
## [1] 1.970506
```