Week 1 - Homework

STAT 420, Summer 2020, D. Unger

Directions

Students are encouraged to work together on homework. However, sharing, copying or providing any part of a homework solution or code is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible.

- Be sure to remove this section if you use this `.Rmd` file as a template.
- You may leave the questions in your final document.

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.

```r
library(ggplot2)
```

(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?

```r
#nrow(msleep)
#ncol(msleep)
msleep
```

```
## # A tibble: 83 x 11
##    name  genus vore  order conservation sleep_total sleep_rem sleep_cycle awake
##    <chr> <chr> <chr> <chr> <chr>              <dbl>     <dbl>       <dbl> <dbl>
##  1 Chee~ Acin~ carni Carn~ lc                  12.1      NA        NA      11.9
##  2 Owl ~ Aotus omni  Prim~ <NA>                17         1.8      NA       7
##  3 Moun~ Aplo~ herbi Rode~ nt                  14.4       2.4      NA       9.6
##  4 Grea~ Blar~ omni  Sori~ lc                  14.9       2.3       0.133   9.1
##  5 Cow   Bos   herbi Arti~ domesticated         4         0.7       0.667  20
##  6 Thre~ Brad~ herbi Pilo~ <NA>                14.4       2.2       0.767   9.6
##  7 Nort~ Call~ carni Carn~ vu                   8.7       1.4       0.383  15.3
##  8 Vesp~ Calo~ <NA>  Rode~ <NA>                 7        NA        NA      17
##  9 Dog   Canis carni Carn~ domesticated        10.1       2.9       0.333  13.9
## 10 Roe ~ Capr~ herbi Arti~ lc                   3        NA        NA      21
## # ... with 73 more rows, and 2 more variables: brainwt <dbl>, bodywt <dbl>
```

There are 83 Observations and 11 Variables in Mammals Dataset.

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

```r
mean(msleep$sleep_rem, na.rm = TRUE)
```

```
## [1] 1.87541
```

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

```r
sd(msleep$brainwt, na.rm = TRUE)
```

```
## [1] 0.9764137
```

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

```r
msleep$name[(which.max(msleep$sleep_rem))]
```

```
## [1] "Thick-tailed opposum"
```

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

```r
mean(subset(msleep, vore == "carni")$bodywt, na.rm = TRUE)
```

```
## [1] 90.75111
```

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?

```r
library(MASS)
#nrow(birthwt)
#ncol(birthwt)
birthwt
```

```
##     low age lwt race smoke ptl ht ui ftv  bwt
## 85    0  19 182    2     0   0  0  1   0 2523
## 86    0  33 155    3     0   0  0  0   3 2551
## 87    0  20 105    1     1   0  0  0   1 2557
## 88    0  21 108    1     1   0  0  1   2 2594
## 89    0  18 107    1     1   0  0  1   0 2600
## 91    0  21 124    3     0   0  0  0   0 2622
## 92    0  22 118    1     0   0  0  0   1 2637
## 93    0  17 103    3     0   0  0  0   1 2637
## 94    0  29 123    1     1   0  0  0   1 2663
## 95    0  26 113    1     1   0  0  0   0 2665
## 96    0  19  95    3     0   0  0  0   0 2722
## 97    0  19 150    3     0   0  0  0   1 2733
## 98    0  22  95    3     0   0  1  0   0 2751
## 99    0  30 107    3     0   1  0  1   2 2750
## 100   0  18 100    1     1   0  0  0   0 2769
## 101   0  18 100    1     1   0  0  0   0 2769
## 102   0  15  98    2     0   0  0  0   0 2778
## 103   0  25 118    1     1   0  0  0   3 2782
## 104   0  20 120    3     0   0  0  1   0 2807
## 105   0  28 120    1     1   0  0  0   1 2821
## 106   0  32 121    3     0   0  0  0   2 2835
```

```
## 107    0  31 100    1    0  0  0  1    3 2835
## 108    0  36 202    1    0  0  0  0    1 2836
## 109    0  28 120    3    0  0  0  0    0 2863
## 111    0  25 120    3    0  0  0  1    2 2877
## 112    0  28 167    1    0  0  0  0    0 2877
## 113    0  17 122    1    1  0  0  0    0 2906
## 114    0  29 150    1    0  0  0  0    2 2920
## 115    0  26 168    2    1  0  0  0    0 2920
## 116    0  17 113    2    0  0  0  0    1 2920
## 117    0  17 113    2    0  0  0  0    1 2920
## 118    0  24  90    1    1  1  0  0    1 2948
## 119    0  35 121    2    1  1  0  0    1 2948
## 120    0  25 155    1    0  0  0  0    1 2977
## 121    0  25 125    2    0  0  0  0    0 2977
## 123    0  29 140    1    1  0  0  0    2 2977
## 124    0  19 138    1    1  0  0  0    2 2977
## 125    0  27 124    1    1  0  0  0    0 2922
## 126    0  31 215    1    1  0  0  0    2 3005
## 127    0  33 109    1    1  0  0  0    1 3033
## 128    0  21 185    2    1  0  0  0    2 3042
## 129    0  19 189    1    0  0  0  0    2 3062
## 130    0  23 130    2    0  0  0  0    1 3062
## 131    0  21 160    1    0  0  0  0    0 3062
## 132    0  18  90    1    1  0  0  1    0 3062
## 133    0  18  90    1    1  0  0  1    0 3062
## 134    0  32 132    1    0  0  0  0    4 3080
## 135    0  19 132    3    0  0  0  0    0 3090
## 136    0  24 115    1    0  0  0  0    2 3090
## 137    0  22  85    3    1  0  0  0    0 3090
## 138    0  22 120    1    0  0  1  0    1 3100
## 139    0  23 128    3    0  0  0  0    0 3104
## 140    0  22 130    1    1  0  0  0    0 3132
## 141    0  30  95    1    1  0  0  0    2 3147
## 142    0  19 115    3    0  0  0  0    0 3175
## 143    0  16 110    3    0  0  0  0    0 3175
## 144    0  21 110    3    1  0  0  1    0 3203
## 145    0  30 153    3    0  0  0  0    0 3203
## 146    0  20 103    3    0  0  0  0    0 3203
## 147    0  17 119    3    0  0  0  0    0 3225
## 148    0  17 119    3    0  0  0  0    0 3225
## 149    0  23 119    3    0  0  0  0    2 3232
## 150    0  24 110    3    0  0  0  0    0 3232
## 151    0  28 140    1    0  0  0  0    0 3234
## 154    0  26 133    3    1  2  0  0    0 3260
## 155    0  20 169    3    0  1  0  1    1 3274
## 156    0  24 115    3    0  0  0  0    2 3274
## 159    0  28 250    3    1  0  0  0    6 3303
## 160    0  20 141    1    0  2  0  1    1 3317
## 161    0  22 158    2    0  1  0  0    2 3317
## 162    0  22 112    1    1  2  0  0    0 3317
## 163    0  31 150    3    1  0  0  0    2 3321
## 164    0  23 115    3    1  0  0  0    1 3331
## 166    0  16 112    2    0  0  0  0    0 3374
## 167    0  16 135    1    1  0  0  0    0 3374
```

```
## 168   0  18 229    2     0   0  0  0    0 3402
## 169   0  25 140    1     0   0  0  0    1 3416
## 170   0  32 134    1     1   1  0  0    4 3430
## 172   0  20 121    2     1   0  0  0    0 3444
## 173   0  23 190    1     0   0  0  0    0 3459
## 174   0  22 131    1     0   0  0  0    1 3460
## 175   0  32 170    1     0   0  0  0    0 3473
## 176   0  30 110    3     0   0  0  0    0 3544
## 177   0  20 127    3     0   0  0  0    0 3487
## 179   0  23 123    3     0   0  0  0    0 3544
## 180   0  17 120    3     1   0  0  0    0 3572
## 181   0  19 105    3     0   0  0  0    0 3572
## 182   0  23 130    1     0   0  0  0    0 3586
## 183   0  36 175    1     0   0  0  0    0 3600
## 184   0  22 125    1     0   0  0  0    1 3614
## 185   0  24 133    1     0   0  0  0    0 3614
## 186   0  21 134    3     0   0  0  0    2 3629
## 187   0  19 235    1     1   0  1  0    0 3629
## 188   0  25  95    1     1   3  0  1    0 3637
## 189   0  16 135    1     1   0  0  0    0 3643
## 190   0  29 135    1     0   0  0  0    1 3651
## 191   0  29 154    1     0   0  0  0    1 3651
## 192   0  19 147    1     1   0  0  0    0 3651
## 193   0  19 147    1     1   0  0  0    0 3651
## 195   0  30 137    1     0   0  0  0    1 3699
## 196   0  24 110    1     0   0  0  0    1 3728
## 197   0  19 184    1     1   0  1  0    0 3756
## 199   0  24 110    3     0   1  0  0    0 3770
## 200   0  23 110    1     0   0  0  0    1 3770
## 201   0  20 120    3     0   0  0  0    0 3770
## 202   0  25 241    2     0   0  1  0    0 3790
## 203   0  30 112    1     0   0  0  0    1 3799
## 204   0  22 169    1     0   0  0  0    0 3827
## 205   0  18 120    1     1   0  0  0    2 3856
## 206   0  16 170    2     0   0  0  0    4 3860
## 207   0  32 186    1     0   0  0  0    2 3860
## 208   0  18 120    3     0   0  0  0    1 3884
## 209   0  29 130    1     1   0  0  0    2 3884
## 210   0  33 117    1     0   0  0  1    1 3912
## 211   0  20 170    1     1   0  0  0    0 3940
## 212   0  28 134    3     0   0  0  0    1 3941
## 213   0  14 135    1     0   0  0  0    0 3941
## 214   0  28 130    3     0   0  0  0    0 3969
## 215   0  25 120    1     0   0  0  0    2 3983
## 216   0  16  95    3     0   0  0  0    1 3997
## 217   0  20 158    1     0   0  0  0    1 3997
## 218   0  26 160    3     0   0  0  0    0 4054
## 219   0  21 115    1     0   0  0  0    1 4054
## 220   0  22 129    1     0   0  0  0    0 4111
## 221   0  25 130    1     0   0  0  0    2 4153
## 222   0  31 120    1     0   0  0  0    2 4167
## 223   0  35 170    1     0   1  0  0    1 4174
## 224   0  19 120    1     1   0  0  0    0 4238
## 225   0  24 116    1     0   0  0  0    1 4593
```

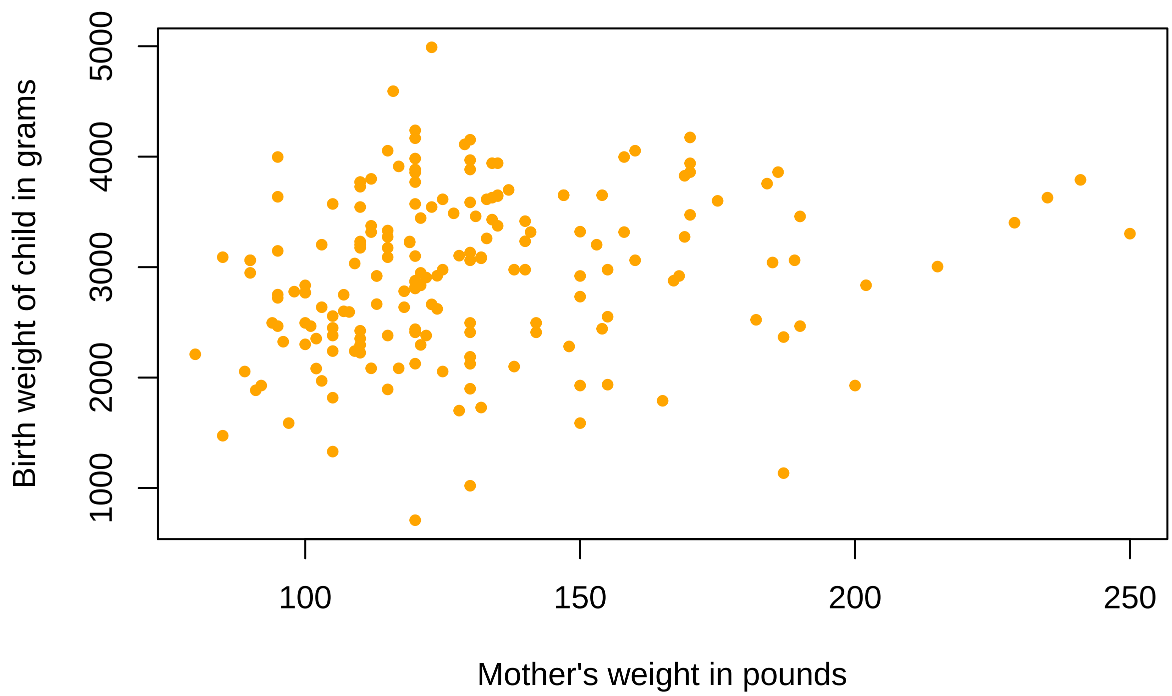
```
## 226   0  45 123    1     0   0  0  0   1 4990
## 4     1  28 120    3     1   1  0  1   0  709
## 10    1  29 130    1     0   0  0  1   2 1021
## 11    1  34 187    2     1   0  1  0   0 1135
## 13    1  25 105    3     0   1  1  0   0 1330
## 15    1  25  85    3     0   0  0  1   0 1474
## 16    1  27 150    3     0   0  0  0   0 1588
## 17    1  23  97    3     0   0  0  1   1 1588
## 18    1  24 128    2     0   1  0  0   1 1701
## 19    1  24 132    3     0   0  1  0   0 1729
## 20    1  21 165    1     1   0  1  0   1 1790
## 22    1  32 105    1     1   0  0  0   0 1818
## 23    1  19  91    1     1   2  0  1   0 1885
## 24    1  25 115    3     0   0  0  0   0 1893
## 25    1  16 130    3     0   0  0  0   1 1899
## 26    1  25  92    1     1   0  0  0   0 1928
## 27    1  20 150    1     1   0  0  0   2 1928
## 28    1  21 200    2     0   0  0  1   2 1928
## 29    1  24 155    1     1   1  0  0   0 1936
## 30    1  21 103    3     0   0  0  0   0 1970
## 31    1  20 125    3     0   0  0  1   0 2055
## 32    1  25  89    3     0   2  0  0   1 2055
## 33    1  19 102    1     0   0  0  0   2 2082
## 34    1  19 112    1     1   0  0  1   0 2084
## 35    1  26 117    1     1   1  0  0   0 2084
## 36    1  24 138    1     0   0  0  0   0 2100
## 37    1  17 130    3     1   1  0  1   0 2125
## 40    1  20 120    2     1   0  0  0   3 2126
## 42    1  22 130    1     1   1  0  1   1 2187
## 43    1  27 130    2     0   0  0  1   0 2187
## 44    1  20  80    3     1   0  0  1   0 2211
## 45    1  17 110    1     1   0  0  0   0 2225
## 46    1  25 105    3     0   1  0  0   1 2240
## 47    1  20 109    3     0   0  0  0   0 2240
## 49    1  18 148    3     0   0  0  0   0 2282
## 50    1  18 110    2     1   1  0  0   0 2296
## 51    1  20 121    1     1   1  0  1   0 2296
## 52    1  21 100    3     0   1  0  0   4 2301
## 54    1  26  96    3     0   0  0  0   0 2325
## 56    1  31 102    1     1   1  0  0   1 2353
## 57    1  15 110    1     0   0  0  0   0 2353
## 59    1  23 187    2     1   0  0  0   1 2367
## 60    1  20 122    2     1   0  0  0   0 2381
## 61    1  24 105    2     1   0  0  0   0 2381
## 62    1  15 115    3     0   0  0  1   0 2381
## 63    1  23 120    3     0   0  0  0   0 2410
## 65    1  30 142    1     1   1  0  0   0 2410
## 67    1  22 130    1     1   0  0  0   1 2410
## 68    1  17 120    1     1   0  0  0   3 2414
## 69    1  23 110    1     1   1  0  0   0 2424
## 71    1  17 120    2     0   0  0  0   2 2438
## 75    1  26 154    3     0   1  1  0   1 2442
## 76    1  20 105    3     0   0  0  0   3 2450
## 77    1  26 190    1     1   0  0  0   0 2466
```

```
## 78     1  14 101     3     1    1  0   0    0 2466
## 79     1  28  95     1     1    0  0   0    2 2466
## 81     1  14 100     3     0    0  0   0    2 2495
## 82     1  23  94     3     1    0  0   0    0 2495
## 83     1  17 142     2     0    0  1   0    0 2495
## 84     1  21 130     1     1    0  1   0    3 2495
```

There are 189 Observations and 10 Variables in "birthwt" dataset.

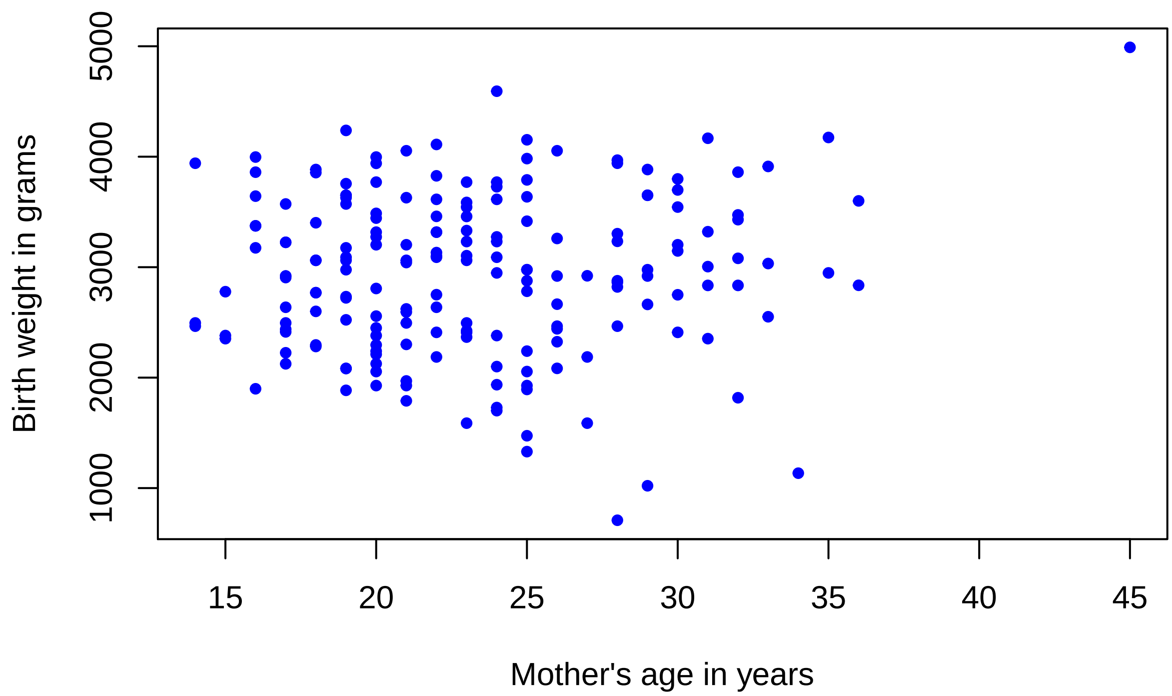
(b) Create a scatter plot of birth weight (y-axis) vs mother's weight before pregnancy (x-axis). Use a non-default color for the points. (Also, be sure to give the plot a title and label the axes appropriately.) Based on the scatter plot, does there seem to be a relationship between the two variables?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

```
#names(birthwt)
#?birthwt
plot(bwt ~ lwt, data = birthwt, xlab = "Mother's weight in pounds", ylab = "Birth weight of child in gra
```



(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.

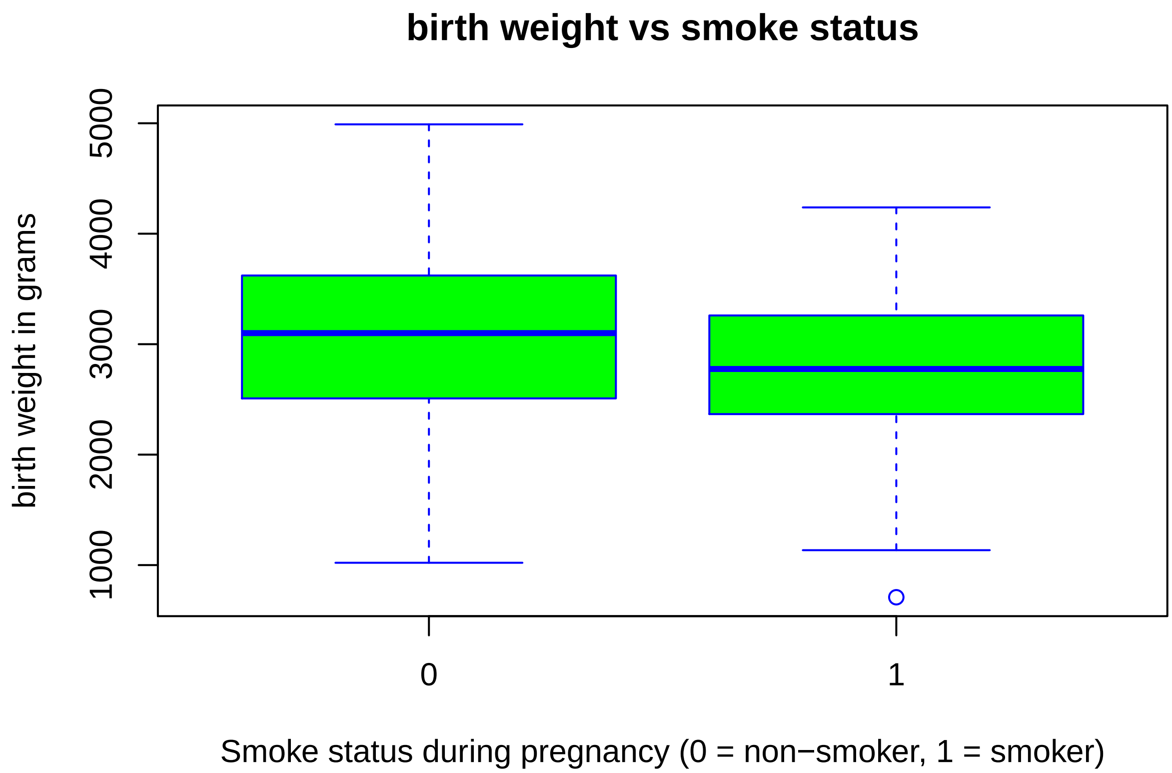
```
plot(bwt ~ age, data = birthwt, xlab = "Mother's age in years", ylab = "Birth weight in grams", pch = 20
```



We see a decreasing trend in the Birth weight with an increase in mother's age. But a increased birth weight at the age of 45 in the dataset looks unusual.

(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.

```
boxplot(bwt ~ smoke, data = birthwt,
        ylab = "birth weight in grams",
        xlab = "Smoke status during pregnancy (0 = non-smoker, 1 = smoker)", main = "birth weight vs sm
        col = "green", border = "blue")
```



Average weight of babies born to smokers looks is less than the averge weight of the babies born to non-smokers. At the same time we need to note that there is high variation in the values.

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.

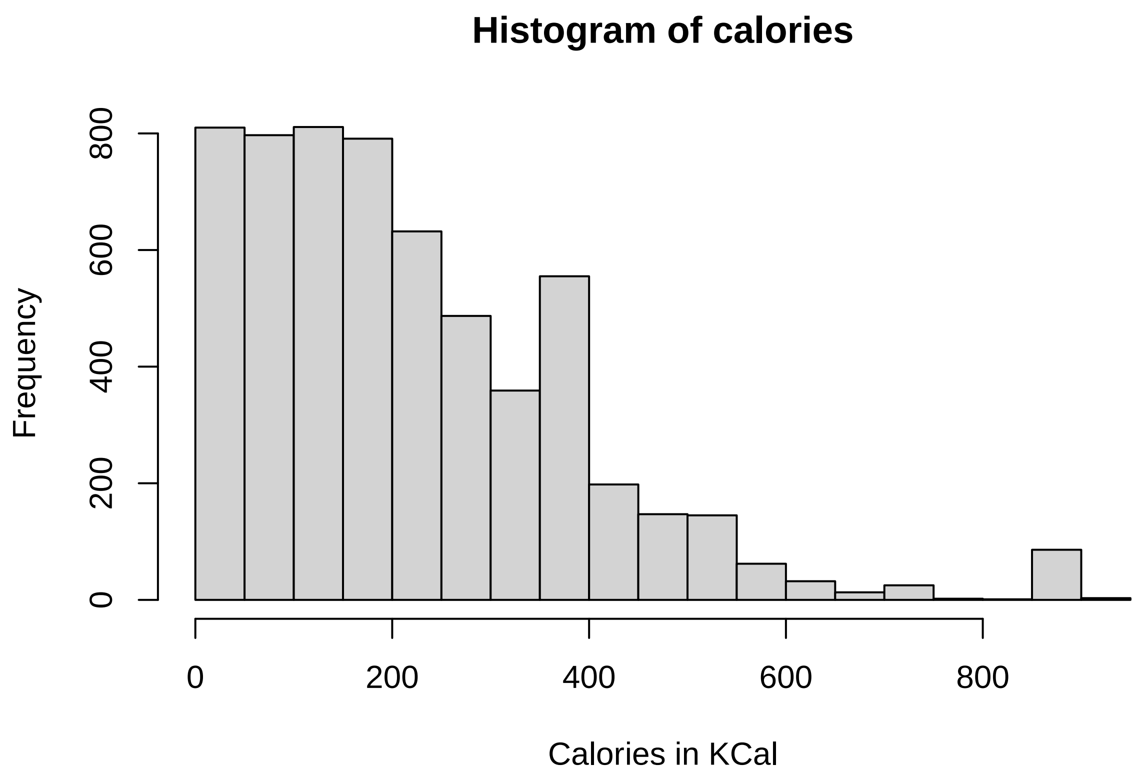
The variables in the dataset are:

- `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?

```
nut_data = read.csv("nutrition-2018.csv")
calories = nut_data$Calories
hist(calories, xlab = "Calories in KCal")
```



Values around 400 and after 800 seems unusual. The distirbution of Calories in right-schewed.

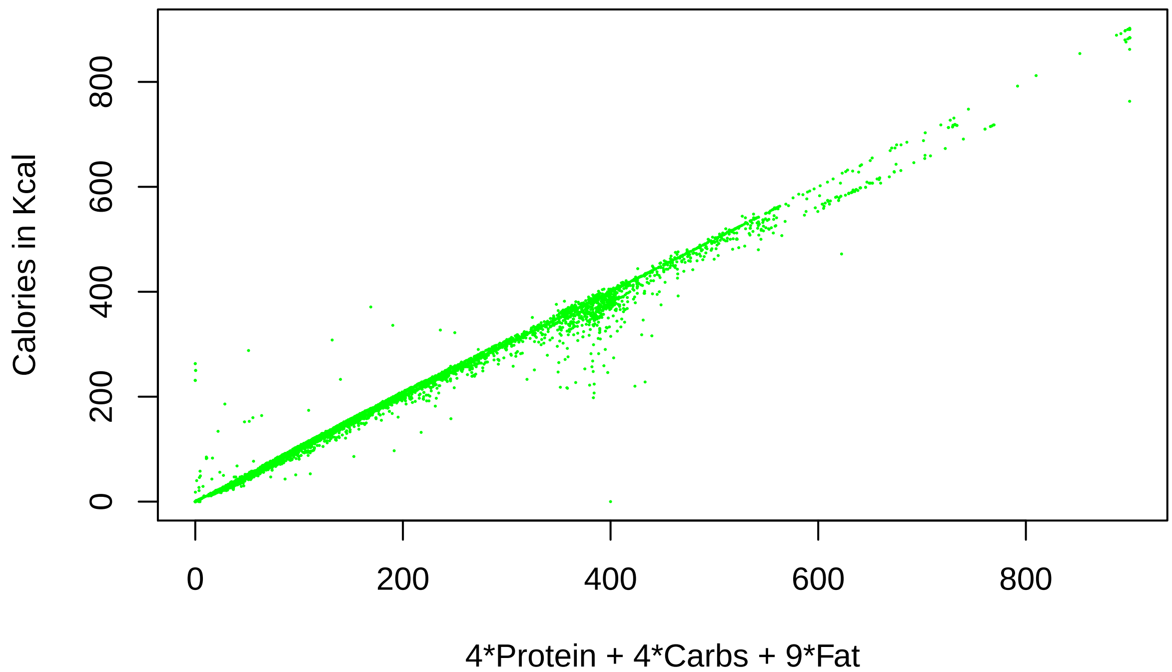
(b) Create a scatter plot of calories (y-axis) vs protein (x-axis). Make the plot presentable. Do you notice any trends? Do you think that knowing only the protein content of a food, you could make a good prediction of the calories in the food?

```
plot(Calories ~ Protein, data = nut_data, main = "Calories vs Protein", xlab = "Protein in grams", ylab
```

As proteins value increases there seems to be a decrease in Calories count. That looks like a trend. Knowing only protein content of a food could help in predicting the calories in the food. But maynot be a good predictor.

(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?

```
plot(Calories ~ I(4*Protein + 4*Carbs + 9*Fat), data = nut_data, xlab = "4*Protein + 4*Carbs + 9*Fat", y
```



Result doesn't look like a straight line. There could be many reasons for not being a straight line. Food is made up of many components other than Proteins/Carbs/Fat. There could be a measurement error.

Exercise 4 (Writing and Using Functions)

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

```r
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:

```r
sum_of_squares = function(x) {
  sum(x^2)
}

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-b))
```

```
## [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:

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

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

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

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

```
## $small
## [1] -3.371739
##
## $large
## [1] 3.229069 3.211199 3.495304
```

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

```
## $small
##  [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
##
## $large
##  [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
```

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

```
## $small
## [1] 0.001703130 0.077464589 0.047054933 0.060877148 0.009629518 0.004321658
## [7] 0.028495955 0.005327612 0.041129370
##
## $large
##  [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`.

```r
mean(list_extreme_values(y, 1.5)$large)
```

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