A3

Maharaj Teertha Deb , 40227747

2024-02-15

Section 2.6:

Problem 1:

The following are a sample of observations on incoming solar radiation at a greenhouse: [11.1 , 10.6 , 6.3 , 8.8 , 10.7 , 11.2 , 8.9 , 12.2]

(a) Assign the data to an object called solar.radiation.

a)Answer:

```
solar.radiation <- c(11.1, 10.6, 6.3, 8.8, 10.7, 11.2, 8.9, 12.2)
print(solar.radiation)
```

```
## [1] 11.1 10.6  6.3  8.8 10.7 11.2  8.9 12.2
```

(b) Find the mean, median, range, and variance of the radiation observations.

b)Answer:

```
mean_radiation <- mean(solar.radiation)
median_radiation <- median(solar.radiation)
range_radiation <- diff(range(solar.radiation))
variance_radiation <- var(solar.radiation)

cat("Mean:", mean_radiation, "\n")
```

```
## Mean: 9.975
```

```
cat("Median:", median_radiation, "\n")
```

```
## Median: 10.65
```

```r
cat("Range:", range_radiation, "\n")
```

```
## Range: 5.9
```

```r
cat("Variance:", variance_radiation, "\n")
```

```
## Variance: 3.525
```

(c) Add 10 to each observation of solar.radiation, and assign the result to sr10. Find the mean, median, range, and variance of sr10. Which statistics change, and by how much?

c) Answer:

```r
sr10 <- solar.radiation + 10

mean_sr10 <- mean(sr10)
median_sr10 <- median(sr10)
range_sr10 <- diff(range(sr10))
variance_sr10 <- var(sr10)

cat("Mean:", mean_sr10, "\n")
```

```
## Mean: 19.975
```

```r
cat("Median:", median_sr10, "\n")
```

```
## Median: 20.65
```

```r
cat("Range:", range_sr10, "\n")
```

```
## Range: 5.9
```

```r
cat("Variance:", variance_sr10, "\n")
```

```
## Variance: 3.525
```

```r
# I think The mean, median and both ends of the range are increased by 10. The variance
# remains unchanged.
```

(d) Multiply each observation by -2, and assign the result to srm2. Find the mean, median, range, and variance of srm2. How do the statistics change now?

d)Answer:

```r
srm2 <- solar.radiation * (-2)

mean_srm2 <- mean(srm2)
median_srm2 <- median(srm2)
range_srm2 <- diff(range(srm2))
variance_srm2 <- var(srm2)

cat("Mean:", mean_srm2, "\n")
```

```
## Mean: -19.95
```

```r
cat("Median:", median_srm2, "\n")
```

```
## Median: -21.3
```

```r
cat("Range:", range_srm2, "\n")
```

```
## Range: 11.8
```

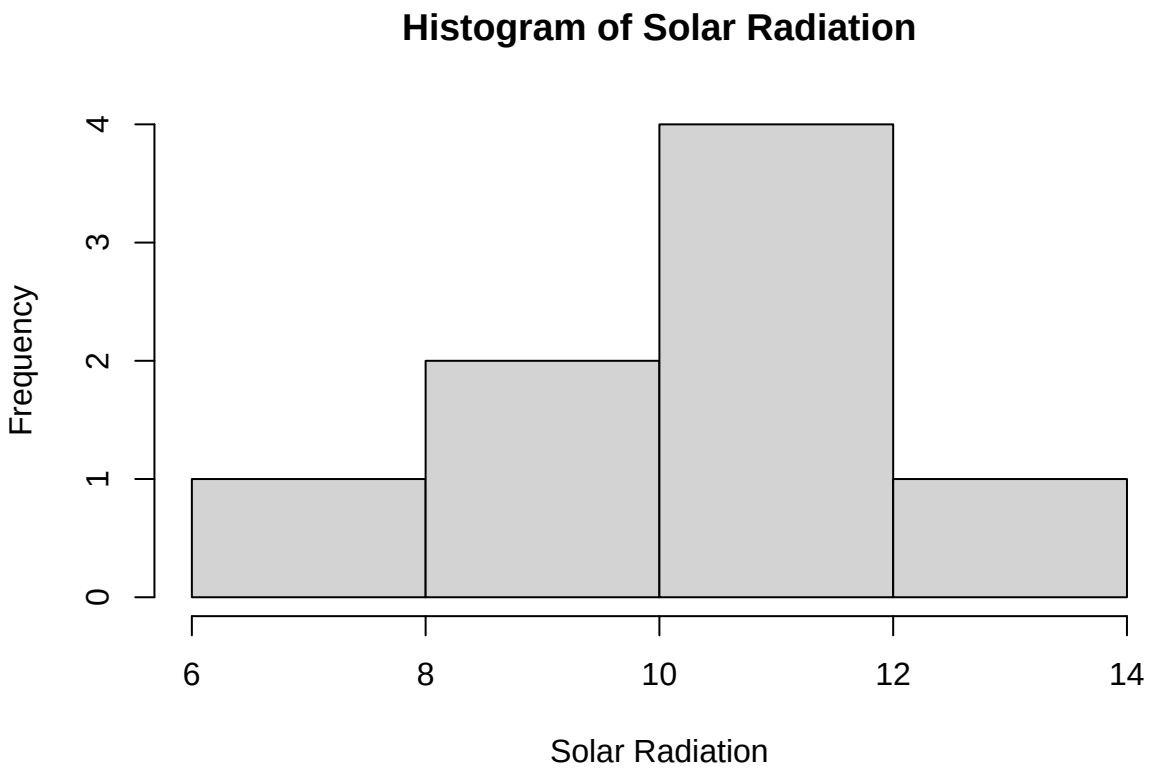
```r
cat("Variance:", variance_srm2, "\n")
```

```
## Variance: 14.1
```

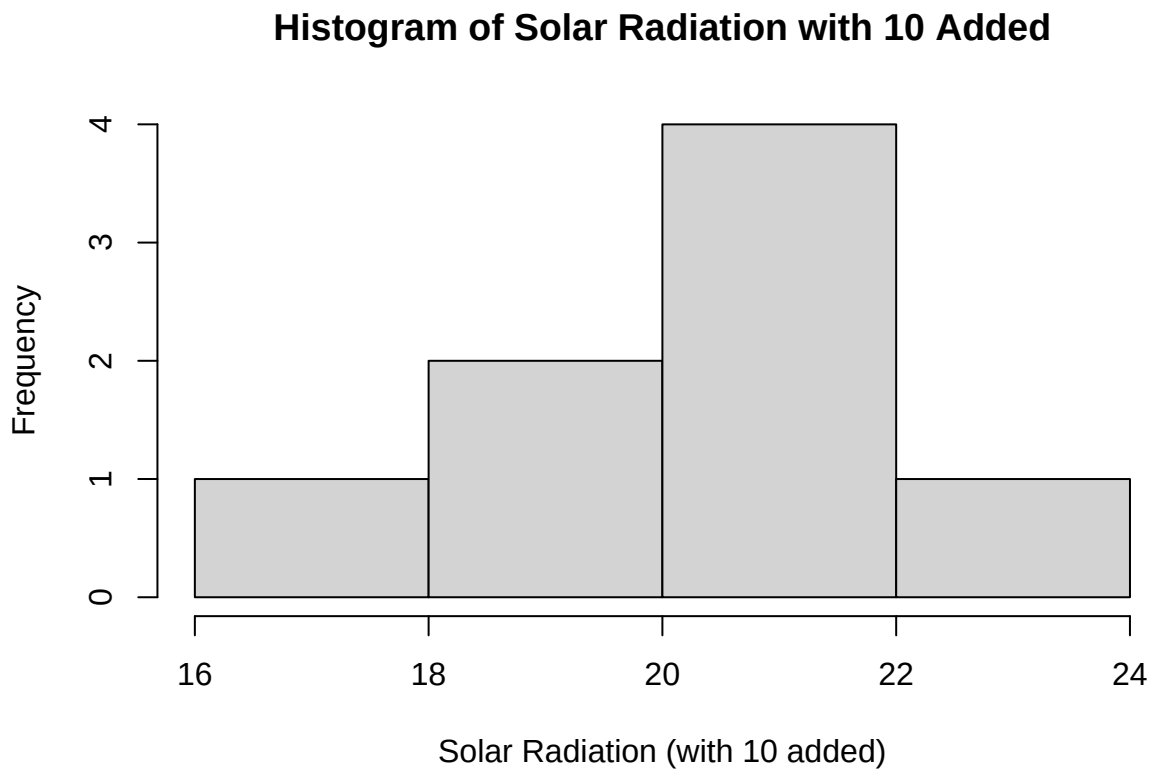
(e) Plot a histogram of the solar.radiation, sr10, and srm2.

e)Answer:

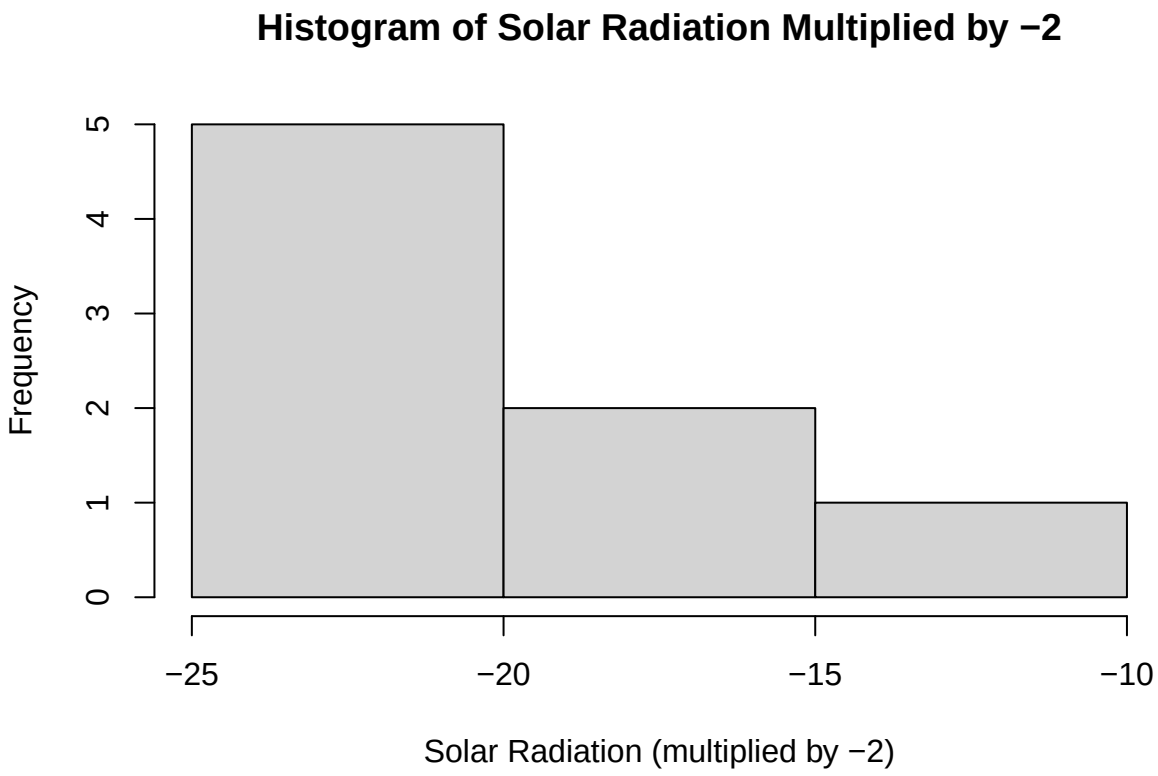
```r
# Original solar radiation
hist(solar.radiation, main = "Histogram of Solar Radiation", xlab = "Solar Radiation")
```




```
# Solar radiation with 10 added
hist(sr10, main = "Histogram of Solar Radiation with 10 Added", xlab =
       "Solar Radiation (with 10 added)")
```




```r
# Solar radiation multiplied by -2
hist(srm2, main = "Histogram of Solar Radiation Multiplied by -2", xlab =
       "Solar Radiation (multiplied by -2)")
```



Problem 2:

Calculate $\sum_{n=1}^{15} \min(2^n, n^3)$. [Hint: the min() function will give the wrong answer.]

Answer:

```
n <- 1:15
sum(pmin(2^n, n^3))
```

```
## [1] 13396
```

Section 2.9:

Problem 2:

Consider the built-in USArrests data frame.

(a) Determine the number of rows and columns for this data frame.

a) Answer:

```
data("USArrests")
# nrow returns the number of rows in the dataset.
cat("Nober of rows in USArrests: " , nrow(USArrests))
```

```
## Nober of rows in USArrests:  50
```

(b) Calculate the median of each column of this data frame.

b) Answer:

```
medians <- sapply(USArrests, median) # the in built function will go through
# column in this dataset and return it's median value.

# Median of Each Column

print(medians)
```

```
##   Murder  Assault UrbanPop     Rape
##     7.25   159.00    66.00    20.10
```

(c) Find the average per capita murder rate (Murder) in regions where the percent age of the population living in urban areas (UrbanPop) exceeds 77%. Compare this with the average per capita murder rate where urban area population is less than 50%.

c) Answer:

```
# Calculate the average per capita murder rate where UrbanPop > 77%
urban_high <- subset(USArrests, UrbanPop > 77)
murder_avg_high <- mean(urban_high$Murder)

# Calculate the average per capita murder rate where UrbanPop < 50%
urban_low <- subset(USArrests, UrbanPop < 50)
murder_avg_low <- mean(urban_low$Murder)

# Calculate the number of observations where UrbanPop > 77% and where UrbanPop < 50%
n_high <- nrow(urban_high)
n_low <- nrow(urban_low)

# Calculate the difference in average per capita murder rates
difference <- murder_avg_high - murder_avg_low

# Print the comparison and difference
cat("Average per capita murder rate where UrbanPop > 77%:", murder_avg_high, "\n")
```

```
## Average per capita murder rate where UrbanPop > 77%: 8.5
```

```
cat("Average per capita murder rate where UrbanPop < 50%:", murder_avg_low, "\n")
```

```
## Average per capita murder rate where UrbanPop < 50%: 8.25
```

```
cat("Difference in average per capita murder rates:", difference, "\n")
```

```
## Difference in average per capita murder rates: 0.25
```

(d) Construct a new data frame consisting of a random sample of 12 of the records of the USArrests data frame,where the records have been sampled without replacement.

d) Answer:

```
set.seed(123)  # Setting seed for reproducibility
sample_data <- USArrests[sample(1:nrow(USArrests), 12, replace = FALSE), ]
print("Random sample of 12 records:")
```

```
## [1] "Random sample of 12 records:"
```

```
print(sample_data)
```

```
##               Murder Assault UrbanPop Rape
## New Mexico      11.4     285       70 32.1
## Iowa             2.2      56       57 11.3
## Indiana          7.2     113       65 21.0
## Arizona          8.1     294       80 31.0
## Tennessee       13.2     188       59 26.9
## Texas           12.7     201       80 25.5
## Oregon           4.9     159       67 29.3
## West Virginia    5.7      81       39  9.3
## Missouri         9.0     178       70 28.2
## Montana          6.0     109       53 16.4
## Nebraska         4.3     102       62 16.5
## California       9.0     276       91 40.6
```

Chapter 2:

Problem 3:

Consider the built-in data frame chickwts.

(a) Create a subset of the data frame called chickwts300p which contains all observations for which the weight exceeds 300.

a) Answer:

```r
data("chickwts")

#chickwts300p which contains all observations for which the weight exceeds 300"


chickwts300p <- subset(chickwts, weight > 300)
print(chickwts300p)
```

```
##    weight      feed
## 11    309   linseed
## 26    327   soybean
## 27    329   soybean
## 31    316   soybean
## 37    423 sunflower
## 38    340 sunflower
## 39    392 sunflower
## 40    339 sunflower
## 41    341 sunflower
## 43    320 sunflower
## 45    334 sunflower
## 46    322 sunflower
## 48    318 sunflower
## 49    325  meatmeal
## 51    303  meatmeal
## 52    315  meatmeal
## 53    380  meatmeal
## 58    344  meatmeal
## 60    368    casein
## 61    390    casein
## 62    379    casein
## 64    404    casein
## 65    318    casein
## 66    352    casein
## 67    359    casein
## 71    332    casein
```

(b) Create another subset called chickwtsLinseed which contains all observations for which the chicks were fed linseed.

b)Answer:

```r
# another subset called chickwtsLinseed which contains all observations for which the
# chicks were fed linseed.

chickwtsLinseed <- subset(chickwts, feed == "linseed")
print(chickwtsLinseed)
```

```
##    weight    feed
## 11    309 linseed
## 12    229 linseed
```

```
## 13    181 linseed
## 14    141 linseed
## 15    260 linseed
## 16    203 linseed
## 17    148 linseed
## 18    169 linseed
## 19    213 linseed
## 20    257 linseed
## 21    244 linseed
## 22    271 linseed
```

(c) Calculate the average weight of the chicks which were fed linseed.

c) Answer:

```r
# the average weight of the chicks which were fed linseed.

average_weight_linseed <- mean(chickwtsLinseed$weight)
print(average_weight_linseed)
```

```
## [1] 218.75
```

(d) Calculate the average weight of the chicks which were not fed linseed.

d) Answer:

```r
#the average weight of the chicks which were not fed linseed.

chickwtsNoLinseed <- subset(chickwts, feed != "linseed")
average_weight_no_linseed <- mean(chickwtsNoLinseed$weight)

print(average_weight_no_linseed)
```

```
## [1] 269.9661
```

Problem 6:

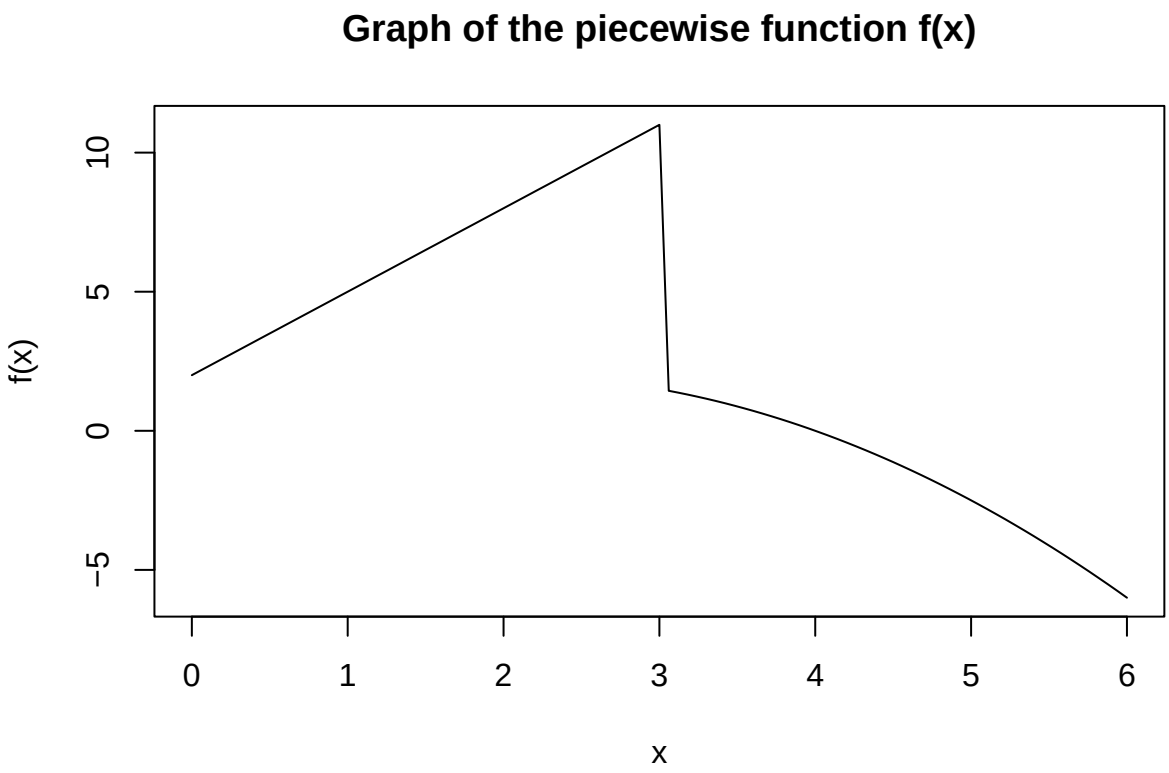
plot the graph of the function $f(x)$:

$$f(x) = \begin{cases} 3x + 2, & \text{if } x \leq 3 \\ 2x - 0.5x^2, & \text{if } x > 3 \end{cases}$$

On the interval $[0, 6]$.

Answe

```r
curve( (3 * x + 2 ) * (x <= 3) + (2 * x - 0.5 * x * x) * (x > 3) ,
       from = 0, to = 6, xlab = "x", ylab = "f(x)",
       main = "Graph of the piecewise function f(x)")
```



Problem 7:

The goal of this exercise is for you to use artificial data to see what the advantages of factors are over numeric and character vectors.

(a) Use the sample () function to construct a vector called dieRolls which simulates the results of 1000000 tosses of a 6-sided die.

a)Answer:

```r
dieRolls <- sample(1:6, 1000000, replace = TRUE)
# We can print it, but it takes too much space showing the answers.
# print(dieRolls)
```

(b) Convert dieRolls to a factor called dieRollsFactor. Change the levels of the factor using the code:

levels (diekollsfactor)<- c (“One”, “Two”, “Three”, “Six”)

b)Answer:

```
dieRollsFactor <- factor(dieRolls)
levels(dieRollsFactor) <- c("One", "Two", "Three", "Four", "Five", "Six")

# print(dieRollsFactor) # We can print it, but it takes too much space showing the answers.
```

(c) Create a character version of the vector using

diekollschar <- as.character(dieRollsFactor)

c) Answer:

```
dieRollsChar <- as.character(dieRollsFactor)

#print(dieRollsChar) # We can print it, but it takes too much space showing the answers.
```

(d) Apply the table () function to each of dieRolls, dieRollsFactor and dieRollschar, and compare the results as well as how the information in each of the data sets is displayed.

d)Answer:

```
# Compare results
table_dieRolls <- table(dieRolls)
table_dieRollsFactor <- table(dieRollsFactor)
table_dieRollsChar <- table(dieRollsChar)

print("Table for dieRolls:")
```

```
## [1] "Table for dieRolls:"
```

```
print(table_dieRolls)
```

```
## dieRolls
##      1      2      3      4      5      6
## 166899 166489 166509 167183 166725 166195
```

```
print("Table for dieRollsFactor:")
```

```
## [1] "Table for dieRollsFactor:"
```

```
print(table_dieRollsFactor)
```

```
## dieRollsFactor
##    One    Two  Three   Four   Five    Six
## 166899 166489 166509 167183 166725 166195
```

```r
print("Table for dieRollsChar:")
```

```
## [1] "Table for dieRollsChar:"
```

```r
print(table_dieRollsChar)
```

```
## dieRollsChar
##   Five   Four    One    Six  Three    Two
## 166725 167183 166899 166195 166509 166489
```

(e) Run the code:

system.time (table (dieRolls)) system.time (table (dieRollsFactor)) system. time (table (dieRollsChar)) to compare the length of time required to construct the 3 tables, using each data type. Which table was produced most quickly? Which was the slowest?

e)Answer:

```r
# Compare time to construct tables
time_dieRolls <- system.time(table_dieRolls)
time_dieRollsFactor <- system.time(table_dieRollsFactor)
time_dieRollsChar <- system.time(table_dieRollsChar)

print("Time to construct table for dieRolls:")
```

```
## [1] "Time to construct table for dieRolls:"
```

```r
print(time_dieRolls)
```

```
##    user  system elapsed
##       0       0       0
```

```r
print("Time to construct table for dieRollsFactor:")
```

```
## [1] "Time to construct table for dieRollsFactor:"
```

```r
print(time_dieRollsFactor)
```

```
##    user  system elapsed
##       0       0       0
```

```r
print("Time to construct table for dieRollsChar:")
```

```
## [1] "Time to construct table for dieRollsChar:"
```

```r
print(time_dieRollsChar)
```

```
##    user  system elapsed
##       0       0       0
```

(f) Run the code

dump ("dierolls", "diekolls.R") dump ("dieRollsFactor", "dieRollsFactor.R") dump ("dieRollsChar", "dieRollsChar.R") Investigate the properties of the files dieRolls.R, dieRollsFactor.R, and dieRollsChar:R. Which one requires the smallest amount of memory? Which file takes the most memory?

f)Answer:

```r
# Investigate memory usage
dump("dieRolls", "dieRolls.R")
dump("dieRollsFactor", "dieRollsFactor.R")
dump("dieRollsChar" , "dieRollsChar.R")
```

Now, let's compare the memory usage of each file:

The file dieRolls.R contains the raw die rolls data, which are stored as integers. Since integers require less memory than factors or characters, dieRolls.R requires the smallest amount of memory. and in my current directoty the size is: 511,685 bytes.

The file dieRollsFactor.R contains the die rolls data converted to a factor with custom levels. Factors in R are stored as integers, with a separate character vector for the levels. While factors can be memory-efficient for categorical data, the addition of custom levels may increase memory usage compared to the original integer data. However, the memory usage is still reasonable for this dataset in my current directory it its : 511,814 bytes. which is almost the same as dieRolls.R

The file dieRollsChar.R contains the die rolls data converted to a character vector. Since characters require more memory than the others. R likely requires most memory than dieRolls.R and dieRollsFactor.R. and in my current directory it is: 647,473 bytes.