PSTAT 10 Worksheet 2

Due 6/26/2024 11:59pm

Problem 1: Basic vector manipulation

1. Recall from lecture my 2023 monthly gas bill in order was given by:

```
gasbill <- c(46, 33, 39, 37, 46, 30, 48, 32, 49, 35, 30, 48)
```

It turns out the charge for December should have been 49 instead of 48. Update the `gasbill` to reflect the true charge. Try not to "cheat" and just type in all the old values again; use the existing `gasbill` vector.

```
gasbill[12] <- 49
gasbill
```

```
##  [1] 46 33 39 37 46 30 48 32 49 35 30 49
```

2. Recreate the following numeric vector. Avoid typing in all of the values manually.

```
## [1] -50 -51 -52 -53 -54 -53 -52 -51 -50
x <- c(-50:-54 , -53:-50)
x
```

```
## [1] -50 -51 -52 -53 -54 -53 -52 -51 -50
```

3. Create a vector from 1 to 10 with increments of 0.05. What is the length of this vector? Hint: Use `seq` with `by` argument.

```
y <- seq(1, 10, 0.05)
y
```

```
##   [1]  1.00  1.05  1.10  1.15  1.20  1.25  1.30  1.35  1.40  1.45  1.50  1.55
##  [13]  1.60  1.65  1.70  1.75  1.80  1.85  1.90  1.95  2.00  2.05  2.10  2.15
##  [25]  2.20  2.25  2.30  2.35  2.40  2.45  2.50  2.55  2.60  2.65  2.70  2.75
##  [37]  2.80  2.85  2.90  2.95  3.00  3.05  3.10  3.15  3.20  3.25  3.30  3.35
##  [49]  3.40  3.45  3.50  3.55  3.60  3.65  3.70  3.75  3.80  3.85  3.90  3.95
##  [61]  4.00  4.05  4.10  4.15  4.20  4.25  4.30  4.35  4.40  4.45  4.50  4.55
##  [73]  4.60  4.65  4.70  4.75  4.80  4.85  4.90  4.95  5.00  5.05  5.10  5.15
##  [85]  5.20  5.25  5.30  5.35  5.40  5.45  5.50  5.55  5.60  5.65  5.70  5.75
##  [97]  5.80  5.85  5.90  5.95  6.00  6.05  6.10  6.15  6.20  6.25  6.30  6.35
## [109]  6.40  6.45  6.50  6.55  6.60  6.65  6.70  6.75  6.80  6.85  6.90  6.95
## [121]  7.00  7.05  7.10  7.15  7.20  7.25  7.30  7.35  7.40  7.45  7.50  7.55
## [133]  7.60  7.65  7.70  7.75  7.80  7.85  7.90  7.95  8.00  8.05  8.10  8.15
## [145]  8.20  8.25  8.30  8.35  8.40  8.45  8.50  8.55  8.60  8.65  8.70  8.75
## [157]  8.80  8.85  8.90  8.95  9.00  9.05  9.10  9.15  9.20  9.25  9.30  9.35
## [169]  9.40  9.45  9.50  9.55  9.60  9.65  9.70  9.75  9.80  9.85  9.90  9.95
## [181] 10.00
```

```
length(y)
```

```
## [1] 181
```

Length of the vector is 181.

4. Create a vector of length 100 from 1 to 10 with uniform increments. What is the increment? Hint: Use `seq` with `length` argument.

```
z <- seq(1, 10, length=100)
z
```

```
##   [1]  1.000000  1.090909  1.181818  1.272727  1.363636  1.454545  1.545455
##   [8]  1.636364  1.727273  1.818182  1.909091  2.000000  2.090909  2.181818
##  [15]  2.272727  2.363636  2.454545  2.545455  2.636364  2.727273  2.818182
##  [22]  2.909091  3.000000  3.090909  3.181818  3.272727  3.363636  3.454545
##  [29]  3.545455  3.636364  3.727273  3.818182  3.909091  4.000000  4.090909
##  [36]  4.181818  4.272727  4.363636  4.454545  4.545455  4.636364  4.727273
##  [43]  4.818182  4.909091  5.000000  5.090909  5.181818  5.272727  5.363636
##  [50]  5.454545  5.545455  5.636364  5.727273  5.818182  5.909091  6.000000
##  [57]  6.090909  6.181818  6.272727  6.363636  6.454545  6.545455  6.636364
##  [64]  6.727273  6.818182  6.909091  7.000000  7.090909  7.181818  7.272727
##  [71]  7.363636  7.454545  7.545455  7.636364  7.727273  7.818182  7.909091
##  [78]  8.000000  8.090909  8.181818  8.272727  8.363636  8.454545  8.545455
##  [85]  8.636364  8.727273  8.818182  8.909091  9.000000  9.090909  9.181818
##  [92]  9.272727  9.363636  9.454545  9.545455  9.636364  9.727273  9.818182
##  [99]  9.909091 10.000000
```

```
length(z)
```

```
## [1] 100
```

```
z[2]-z[1]
```

```
## [1] 0.09090909
```

Increment is 0.09090909.

5. What happens if you try to use `seq` with both the `length` and `by` arguments specified?

```
w <- seq(1, 10, length = 91)
```

If the above chunk is run, we get the error “Error in seq.default(1, 10, by = 1, length = 1) : too many arguments”. For seq, we can only specify either by or length.

Problem 2

Download the file `ws2.csv` from the course website and import it into R. This data set has two variables named `x` and `y`.

Make sure to change your `here()` function to point to your `ws2.csv` file.

```
library(here)
```

```
## here() starts at G:/Documents/School/2023-2024/Summer 2024/PSTAT 10/PSTAT10-Worksheet-2
```

```
ws2_df <- read.csv(here("week1_files", "ws2.csv"))
summary(ws2_df)
```

```
##        x               y         
##  Min.   : 2.00   Min.   :  1.00  
##  1st Qu.:25.75   1st Qu.: 26.00  
##  Median :49.50   Median : 53.50  
##  Mean   :49.11   Mean   : 52.93  
##  3rd Qu.:70.00   3rd Qu.: 78.00  
##  Max.   :99.00   Max.   :100.00
```

Remember the variables in a data frame are accessed by name with the dollar sign (and that the result is a vector).

1. Determine the lengths of x and y.

```
length(ws2_df$x)
```

```
## [1] 100
```

```
length(ws2_df$y)
```

```
## [1] 100
```

The lengths of both x and y are 100.

2. What is the 40th element of x and the 80th element of y?

```
ws2_df$x[40]
```

```
## [1] 30
```

```
ws2_df$y[80]
```

```
## [1] 42
```

The 40th element of x is 30. The 80th element of y is 42.

3. What is the average of all the values in the data frame, including both x and y?

```
mean(c(ws2_df$x, ws2_df$y))
```

```
## [1] 51.02
```

The average of all the values in the data frame is 51.02.

4. How many elements of x are greater than 70?

```
length(ws2_df$x[ws2_df$x>70])
```

```
## [1] 24
```

24 elements of x are greater than 70.

Let's look at the first 4 elements of x and y:

```
ws2_df$x[1:4]
```

```
## [1] 74 89 78 23
```

```
## [1] 74 89 78 23
ws2_df$y[1:4]
```

```
## [1] 58 26 48 80
```

```
## [1] 58 26 48 80
```

The first three elements of x are greater than or equal to their corresponding element in y: $74 > 58$, $89 > 26$, $78 > 48$. But the fourth element of `x`, 23, is less than the fourth element of `y`, 80.

5. How many elements of x are greater than or equal to the corresponding element in y?

```
length(ws2_df$x[ws2_df$x > ws2_df$y])
```

```
## [1] 45
```

45 elements of x are greater than or equal to the corresponding element in y.

6. What is the proportion of elements of x that are greater than or equal to the corresponding element in y?

```
length(ws2_df$x[ws2_df$x >= ws2_df$y])/length(ws2_df$x)
```

```
## [1] 0.46
```

0.46 of the elements of x are greater than or equal to the corresponding element in y.

7. How many values in x differ from their corresponding value in y by more than 10 in absolute value? Hint: there is an `abs` function.

```
length(ws2_df$x[abs(ws2_df$x - ws2_df$y)])
```

```
## [1] 99
```

99 values in x differ from the corresponding value in y by more than 10 in absolute value.

Problem 3

Create a vector of integers from 1 to 12 inclusive.

```
a <- 1:12
a
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

1. Use the vector to create a 3x4 matrix. Did recycling occur?

```
b <- matrix(a, nrow = 3, ncol = 4)
b
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

Recycling did not occur.

2. Use the vector to create a 4x4 matrix. Did recycling occur?

```
c <- matrix(a, nrow = 4, ncol = 4)
```

```
## Warning in matrix(a, nrow = 4, ncol = 4): data length differs from size of
## matrix: [12 != 4 x 4]
```

```
c
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9    1
## [2,]    2    6   10    2
## [3,]    3    7   11    3
## [4,]    4    8   12    4
```

Recycling did occur.

Problem 4

Use the heights_df data frame from worksheet 1. The height variable is given in centimeters (cm).

1. Write a vectorized function cm_to_inch that takes a numeric centimeter and converts it to inches.

```r
heights_df <- read.csv(here("week1_files", "heights.csv"))

cm_to_inch <- function(cm)
{
  inch <- cm * 0.3937
  return(inch)
}
```

Apply the function to the height vector. First 10 elements are shown below:

```r
head(cm_to_inch(heights_df$height), 10) # the head function gives the first elements
## [1] 62.40 67.08 65.52 71.37 68.25 73.71 60.84 65.13 76.05 64.35
```

2. Write a vectorized function cm_to_ft_inch that converts numerical values given in cm to a feet inch format, rounding to the nearest inch. For example,

```r
cm_to_ft_inch <- function(cm)
{
  inch = cm_to_inch(cm)
  feet = inch %/% 12
  inch = inch %% 12
  return(paste(feet, round(inch), sep = " ", collapse = NULL))
}

cm_to_ft_inch(178)
```

```
## [1] "5 10"
```

```r
## [1] "5 9"
```

You may need the (vectorized) quotient function %/% and the remainder function %%:

```r
# Quotient: 3 goes into 7 two times
7 %/% 3
```

```
## [1] 2
```

```r
## [1] 2
# Remainder: The remainder when 7 is divided by 3 is one
7 %% 3
```

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

```r
## [1] 1
```

Remember you should look things up on StackOverflow if you're stuck with some operations. Apply the function to the height vector.

```r
head(cm_to_ft_inch(heights_df$height), 10)
```

```
##  [1] "5 3" "5 8" "5 6" "6 0" "5 9" "6 2" "5 1" "5 6" "6 5" "5 5"
```

```r
## [1] "5 2" "5 7" "5 6" "5 11" "5 8" "6 2" "5 1" "5 5" "6 4" "5 4"
```