Lab 2: Data Types

INSERT YOUR NAME HERE (INSERT YOUR UW NETID HERE)

Due by 23:59pm on Jan 22, 2024

Total Points: 40

Part 1. Review Questions (2+5+3 pts)

1. Multiply the inverse of a matrix $\begin{bmatrix} 3 & 2 & 1 \\ 4 & 8 & 1 \\ 5 & 9 & 16 \end{bmatrix}$ with itself.

   Also, return those entries that are bigger than 10^{-9}.

```
# Your code here
```

2. Make a list `lst1` with components

- `1:15` under the name `num_vec`;
- `matrix(15:1, ncol = 3)` under the name `mat`;
- `rep(c("a", "x"), each = 3)` under the name `char_vec`;
- `list(x = c(1,2), y="STAT 302")` under the name `sublst`.

Answer the following questions using R:

- Compute the sum of the component `num_vec`;
- What is the element in position [2,3] in the component `mat`?
- What is the third element in the component `char_vec`?
- Use the function `strsplit()` with argument `split = ""` to split the subcomponent `y` in the component `sublst`. What is the data type for the result `strsplit()`?
- Subset the result after `strsplit()` via `[[1]]`. What is the fifth element of this character vector?

```
# Your code here
```

3. Download the `family.txt` shown in Lecture 2 to your laptop. Then, read the file into R using the function `read.delim()`. Then, compute the following statistics in R:

- What is the standard deviation of ages in `family.txt`?
- What is the percentage of males in `family.txt`?
- What is the maximum BMI within all the female individuals?

```
# Your code here
```

Part 2: Normal Distribution (2+5+3+4 pts)

R provides several functions for the normal/Gaussian distribution:

- `dnorm()` computes the density function of a normal distribution;
- `pnorm()` calculates the percentiles (or equivalently, the cumulative distribution function) of a normal distribution;
- `qnorm()` returns the quantiles of a normal distribution;
- `rnorm()` generates the normally distributed random variables.

Use R to answer the following questions:

1. Create and store a vector `norm_vec` with $100,000$ random variables from a Normal distribution with mean 6 and standard deviation 2. Print out the first 7 elements of `norm_vec` using the function `head()`.

```r
set.seed(123)  ## Don't change this line. It makes the result reproducible.
# Your code starts from here
```

2. Plot two histograms, one with the first 100 elements of `norm_vec`, and the other with all the elements of `norm_vec`. Set the argument `freq = FALSE` for both histograms for better comparisons.

- Change the x axis labels for both histograms to "Observations".
- Set their titles as "Histogram of N(6,2) distributed random sample with n=THE CORRECT NUMBER OF SAMPLE POINTS". Remember to change "THE CORRECT NUMBER OF SAMPLE POINTS".
- Answer it by words: Which one looks more symmetric?

```r
# Your code starts from here
```

3. Standardize the vector `norm_vec` to $N(0,1)$ by subtracting its mean and then dividing it by its standard deviation. Name it as `norm_vec_std`. Compute the standard deviation of `norm_vec_std`. Also, what is the percentage of observations in `norm_vec_std` that are greater than 1.644854?

```r
# Your code here
```

4. Apply the function `pnorm()` (without specifying any other arguments) to the vector `norm_vec_std`. Then, compute its mean and variance after applying the function `pnorm()`. Finally, plot its histogram after applying the function `pnorm()` with the argument `freq = FALSE`.

- Describe in words what do you see from the histogram. (Hint: How is the height of each bin compared with others?)

```r
# Your code here
```

Part 3: Binomial Distribution (4pts per question)

The binomial distribution $\text{Bin}(m, p)$ is defined by the number of successes in m independent trials, each have probability p of success. Think of flipping an (unfair) coin m times, where the coin could be biased and has probability p of landing on heads.

Similar to the above normal distribution, R also provides several functions for the binomial distribution:

- `dbinom()` computes the probability mass function of a binomial distribution;
- `pbinom()` calculates the percentiles (or equivalently, the cumulative distribution function) of a binomial distribution;
- `qbinom()` returns the quantiles of a binomial distribution;
- `rbinom()` generates the random variables from a binomial distribution.

1. Initialize a matrix `binom_mat` with 3 columns and 100 rows, whose entries are all `NA`.

- Then, fill in each column with random samples from binomial distributions with $m = 300, p = 0.25$ (first column), $m = 300, p = 0.5$ (second column), and $m = 300, p = 0.75$ (third column), respectively.
- Compute the column means of `binom_mat`.

```r
set.seed(1234)  ## Don't change this line. It makes the result reproducible.
# Your code starts from here
```

2. Compute the means of every 10 elements in the first column of `binom_mat`. There should be 10 mean values in total. Then, output the median of these 10 mean values. Assign it to a variable `MoM`.

- Compared with the mean of the first column of `binom_mat`, is `MoM` closer to the expected mean `75`? (Output a logical TRUE/FALSE using R!)

```r
# Your code here
```

3. Now, change the first element in the first column of `binom_mat` to -100. Then, repeat what we did in Question 2 (i.e., compute the means of every 10 elements in the first column of `binom_mat` and then calculate the median as `MoM2`.)

- Now, compared with the mean of the first column of `binom_mat`, is `MoM2` closer to the expected mean `m*p = 75`? (Output a logical TRUE/FALSE using R!)

```r
# Your code here
```

4. Create a list `binom_lst` with 3 components:

- A vector with 500 elements from a $\text{Bin}(300, 0.75)$ and name it as `binom500`;
- A vector with 1000 elements from a $\text{Bin}(300, 0.75)$ and name it as `binom1000`;
- A vector with 20000 elements from a $\text{Bin}(300, 0.75)$ and name it as `binom20000`.
- Compute the mean of each component of `binom_lst`. Which one is closest to the expected mean `m*p = 225`? Can you explain why?
- Look at the documentation of the functions `qqnorm()` and `qqline()`. Make QQ-plots with diagonal lines for each component of `binom_lst`. Which QQ-plot is most aligned with the diagonal line? Can you explain why?

```r
set.seed(1234)  ## Don't change this line. It makes the result reproducible.
# Your code starts from here
```