Lab/HW 2: Basic Data Structures

Your lab/homework must be submitted in with two files: (1) R Markdown format file; (2) a pdf or html file. Other formats will not be accepted. Your responses must be supported by both textual explanations and the code you generate to produce your result.

Part I

1. The normal distribution:

[20 pts]

Explore the normal distribution in R by reading about related functions by `?rnorm`.

[4 pts] 1. Generate $n = 100$ random draws from the standard normal distribution and assign those to the variable `s`.

[3 pts] 2. What are the type and dimensions of the data that the function takes as input here?

[3 pts] 3. What are the type and dimensions of the data that the function returns?

[3 pts] 4. Use `hist()` to plot a histogram of `s`, normalized to be a density (so that the total area of the bars is 1, check `?hist`), make sure that the number of `breaks` (a parameter in the `hist()` function) does justice to your data, as the default value is not good enough at times. Justify your choice (you can provide several figures, prior knowledge or any other convincing argument).

[4 pts] 5. Add a line to the histogram that shows the theoretical normal density:

- First, determine the range you are interested in (the limits of the horizontal axis).
- Generate a sequence that covers the above range and is dense enough to generate a smooth curve, assign it to a variable, say `x`. What is the variable type?
- Find the appropriate function that will calculate for you the values of the density of the normal random variable. Use that to plot the additional line, in red please. What is the input data type for that density function? What is the data type of its output? Does the function operate element by element?

[3 pts] 6. Plot another histogram of the standard normal CDF when applied to `s` (a transformation of a random variable is another random variable). Do you identify this distribution? Hint: start by identifying the range of values that are generated. If you are not sure, try taking a larger `n`.

[4 pts, Extra credit] Provide a proof in LaTeX (embedded in your RMarkdown document) for your assertion about how the above random variable is distributed.

2. Sampling from the truncated normal distribution

[40 pts]

The truncated normal distribution is the probability distribution derived from that of a normally distributed random variable by bounding the random variable from either below or above (or both) (See Wikipedia).

In this exercise we will generate samples from a truncated normal random variable in a very simple way.

Our goal is to sample from a truncated normal distribution that is derived from a standard normal variable that is truncated from below at 0.5.

[5 pts] 1. First, generate `n = 10000` samples from a standard normal random variable, assign those to, say, a variable named `s`.

[5 pts] 2. Then, using a single command, assign to a new variable, say, `trunc_s`, all the values of `s` that are greater than or equal to 0.5. **Explain in words exactly each stage of the computation, which functions are used, what are their input and output data types, etc.**

That is it! we generated a sample of truncated normal values.

[5 pts] 3. Use `hist()` to plot a histogram of `trunc_s`, normalized to be a density, make sure that the number of `breaks` does justice to your data. Justify your choice.

[5 pts] 4. Let us compare the results to the theoretical density. Choose a range for which you want to compute the density, and assign it to a vector. Using only basic operators and functions for the standard normal family (from `dnorm`, `pnorm`, `qnorm`, `rnorm`) and the formula for the density of the truncated normal random variable from Wikipedia that operate on the range vector or other constants (note, we are using `b = Inf` here, what are the other constants?) to compute the density of the truncated normal and add a red line to the histogram that depicts it. For legibility, you may separate the computations to, say, `numerator` and `denominator`. **Carefully describe each part of your computation: which functions operate on which data types? Is recycling used? Is the computation element by element?**

[5 pts] 5. Using the same careful descriptions as in the above, compute the expected value and standard deviation for this truncated normal distribution (based on the theoretical formulas) as well as the average and empirical standard deviation of the sample. Absurdly as it may sound, if you do this correctly, something should go wrong for the theoretical standard deviation. What went wrong? Fix it in a new chunk of code! Are the final values near?

[5 pts] 6. Let us compare the results to the `truncnorm` package. First, install the package and load it. Then, apply `dtruncnorm` (with the correct parameters) to the same range you used before. Add another, blue dashed line to the histogram and check whether or not the lines are similar.

[5 pts] 7. Are the values you computed for the truncated normal density and the ones from the package "the same" (check in at least 3 ways)?

[5 pts] 8. What is the problem/s with this sampling method? Exemplify with code.

[5 pts] Extra credit: compute the *expected length of the sample generated as a function of* `a`. Is it "close" to the actual length that you got? You may sample several times and see how the results look like.

Part II

1. Syntax and class-typing.

[15 pts] [8 pts] 1a. For each of the following commands, either explain why they should be errors, or explain the non-erroneous result.

```
vector1 <- c("5", "12", "7", "32")
max(vector1)
sort(vector1)
sum(vector1)
```

[7 pts] 1b. For the next series of commands, either explain their results, or why they should produce errors.

```
vector2 <- c("5",7,12)
vector2[2] + vector2[3]

list4 <- list(z1="6", z2=42, z3="49", z4=126)
list4[[2]]+list4[[4]]
list4[2]+list4[4]
```

2. Some regression

[25 pts] The following code generates a sequence of `X` values from 1 to 100, each three times. It defines `Y` as a linear function of `X` plus normal noise.

```r
n = 100
X = rep(1:n, each = 3)
Y = 0.5 + 2 * X + rnorm(100 * 3)
```

[6 pts] 2a. Carefully explain each of the above computations: which functions are in it, in which order, what are their respective inputs and outputs, how are the computations performed.

[6 pts] 2b. The following code regresses `Y` on `X` and presents the estimated coefficients and predicted values (expected `Y` values for the same `X` values). i. What are the corresponding data types? ii. Do the results make sense to you (regression-wise)? Why?

```r
reg_Y_X = lm(Y ~ X)
coef(reg_Y_X)
predict(reg_Y_X)
```

[6 pts] 2c. The data type of `reg_Y_X` is `list`, specifically a *named* list. That is, we can address `reg_Y_X[[1]]` according to its name, so `reg_Y_X$coefficients` is the same as `reg_Y_X[[1]]`.

What are the elements that comprise `reg_Y_X` and their respective data types? (there are 12 of those, 2 of them are types that we haven't discussed yet) (please distinguish names types from those without names). You can use functions of the `is.____` family or other methods to prove your claims.

[7 pts] 2d. In light of the above, explain what the `coef` function from 2b does.

[4 pts, Extra credit] 2e. Write a line of code that only involves information stored in `reg_Y_X` (or any of its elements) and provides the same results as the `predict(reg_Y_X)` line from 3b.