Homework 3

Instructions

Complete the following problems and submit to Gradescope your work as a file named `hw3.Rmd`.

A file with the wrong name or a bug will prevent the autograder from being able to run your submission. Check that your filename is correct, and check for bugs clearing your memory and then running your file from start to finish. It's also a good idea to check that your file "knits."

Do not use any loops!

Problem 1

For this problem, consider a Normal population with a mean of 73.2 and a standard deviation of 4.9 and consider a 5% significance level. You should generalize as much as possible and hard-code as little as possible.

a

Save the parameter values as objects. Use the names `mu`, `sigma` and `alpha`.

b

Write a function called `my.test()` that generates a sample of a given size (provided via a function argument) from the aforementioned population and uses that sample to conduct a z-test for whether or not the true mean of this population is 73.2. The function should output a `TRUE` if the test rejects the null hypothesis and `FALSE` otherwise.

Your function should **NOT** use a pre-existing R function that conducts the z-test.

Execute your function on a few sample sizes to try it out.

c

Without using any loops, execute your function written in part b $10,000$ times using a sample size of 23. (You should not display all 10,000 outcomes.) What proportion of tests reject the null hypothesis? Assign your answer to a variable called `propReject`

d

Theoretically, what is the proportion resulting from part c expected to be? Assign your answer to the variable called `trueFalsePositiveRate`

e

Write a function called `rep.z()` that will output the proportion described in part c for a given sample size (again, without using any loops).

Be sure to use the same number of simulations as before. Capture the same number-of-simulations parameter as before–don't pass it in as an argument.

Execute your function for the sample sizes 8, 23, and 52.

f

Without using any loops, execute the function written in part e for every sample size from 3 through 52. Store the results as a vector called `myResults`. Again, capture the same number-of-simulations parameter was before. Don't pass it in as an argument.

Note: You should not execute your function in a separate statement for each sample size.

g

What do you notice about the general trend of the resulting proportions? Does sample size appear to have any effect on the results? Are all of the proportions "close" to `trueFalsePositiveRate`, regardless of sample size? If yes, assign `TRUE` to `allClose`. Otherwise, assign `FALSE`.

Problem 2

The data file `nym2021.txt` contains the finishing time and some demographics for select finishers of the 2021 New York City Marathon.

a

Read in the file and save the data in a data frame called `nym2021`. Be sure not to refer to file location that *only* exists on your machine, or else Gradescope will not be able to run your file. Prefer `character` columns to `factor` ones.

Print the first few rows of the data frame using the `head()` function. It should look like this:

```r
head(nym2021)
```

```
##   Age Place DivPlace    DIV DivAge   Time BostonQualifier HomeStateOrCountry
## 1  33  2417      416 M30-34  30-34 207.78               N                 NY
## 2  42  1410      211 M40-44  40-44 196.07               N                IRL
## 3  39   592       17 F35-39  35-39 179.35               Y                 NY
## 4  24  1464       11 F20-24  20-24 197.00               Y                 NY
## 5  54   584       11 M50-54  50-54 179.25               Y                 NY
## 6  36   257        9 F35-39  35-39 169.93               Y                 NY
```

b

Determine the number of finishers' times that are contained in this data set. Store this number as `numFinishers`

c

Determine the number of finishers in the data whose home country is the U.S. Call the answer `numUSAFinishers`

Note: Include U.S. territories as the U.S.

d

Determine the number of finishers representing each country. Use the `table` function to create a `table` object called `numEachCountry`. When counting the american competitors, use the key `"USA"`.

e

Determine the number of countries represented in the data. Store your number under the name `numUniqueCountries`.

f

Determine the age of the youngest and oldest finishers (in that order) given in the data. Store these two numbers in a vector called `oldestAndYoungest`

g

Determine the division of the fastest and slowest finishers given in the data. Store them in the variables called `divFastest` and `divSlowest`, respectively.

h

Create a variable called `numOutside` that holds the number of finishers who finished outside the Top 100 of their division.

i

Determine the divisions of the finishers who finished in the Top 5 of their division. Arrange these divisions in order–female before male, young before old. Store them in the variable `divTopFive`

Note: Each division should be displayed only once.

j

Subset all information for finishers who finished in the Top 50 overall. Store this in a `data.frame` called `top50`

Here's what mine looks like:

```r
head(top50)
```

```
##    Age Place DivPlace    DIV DivAge   Time BostonQualifier HomeStateOrCountry
## 7   35    30        1 F35-39  35-39 146.17               Y                 AZ
## 33  26    28       10 M25-29  25-29 146.07               Y                 NY
## 50  23    39        1 M20-24  20-24 149.22               Y                MEX
## 63  24    33        1 F20-24  20-24 146.30               Y                 MN
## 65  31    44       14 M30-34  30-34 150.47               Y                IRL
## 90  25    43       14 M25-29  25-29 150.40               Y                 AR
```

k

Determine the average age of finishers who did and who did not (in that order) qualify for the Boston Marathon. Store it in a `vector` called `aveAges`.

Note: be careful…a function we talked about in class returns an `array` not a `vector`!