Assignment 4

BIOSTAT 213: Introduction to Computing in the R Software Environment

Due: Wednesday, November 27, 2019

Collaboration with other students in this course is permitted on this assignment. However, you must write all work—including R code—in your own voice, as much as possible, and acknowledge your collaborators as thoroughly as possible.

The assignment is due online, in the Collaborative Learning Environment, by 11:59 pm, Wednesday, November 27, 2019. You must submit a Word-compatible document or a PDF file.

For each of the below, share your code as well as the output (if there is any output). If your answer to a question is unclear from your R output, or difficult to identify, include an answer in text form (in addition to the code and output).

1. As you may have noticed, the `round()` function eliminates any trailing 0s. For example:

```
> round(9.403,2)
[1] 9.4
```

   One way around this is to use the `format()` function:

```
> format(9.403,nsmall=2,digits=2)
[1] "9.40"
> round(9.403,2)
[1] 9.4
```

   Write a rounding function that accepts 2 arguments: the number that should be rounded and the number of digits. Here's an example function:

```
> Round(3.403,2)
[1] "3.40"
> Round(pi,4)
[1] "3.1416"
> Round(-90.0004,3)
[1] "-90.000"
```

   Include some tests of the function in your response. You may use the inputs above if you wish.

   (Reminder: as a general rule in R and other statistical software, rounding functions should only be used when presenting final numbers. In other words, do not round in the "middle" of your work. It will lead to imprecision.)

2. Write a function that will implement a two-sample t-test, using `t.test()`. The function should output a formatted version of the results, showing just the difference in means and the associated 95% confidence interval. Note that you should take the difference by subtracting the mean in the second group from the mean in the first group. Optionally, you may consider including the custom rounding function from above. Here's an example function:

```
> difference(y=ii$bmxbmi,x=ii$gender)
[1] "1.23 (95% CI:0.79, 1.67)"
> difference(y=ii$bpxsy1,x=ii$hs)
[1] "4.30 (95% CI:3.22, 5.37)"
```

   Include some tests of the function in your response. You may use the inputs above if you wish.

To be clear: you can access these numbers from `t.test()`. The group means are accessible via `$estimate` while the confidence interval is accessible via `$conf.int`.

3. Write a function that will perform a bivariate test on categorical data. The function should accept 3 arguments: 2 vectors of data and an option that indicates whether the function should use `chisq.test()` or `fisher.test()`. Here's an example function:

```
> bivariate.categorical(ii$gender,ii$hs)

    Pearson's Chi-squared test with Yates' continuity correction

data:  table(yy, xx)
X-squared = 131.05, df = 1, p-value < 2.2e-16
> bivariate.categorical(ii$gender,ii$hs,fisher=TRUE)

    Fisher's Exact Test for Count Data

data:  table(yy, xx)
p-value < 2.2e-16
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.4241661 0.5473432
sample estimates:
odds ratio
 0.4819085
```

Use the `presidencies.csv` data file for the following problems.

4. Use `tapply()` to find the total duration, in days, that each party (`party`) has held office.
5. Use `sapply()` to find the object type of each variable.

Use the `nhanes.csv` data file for the following problems.

6. Use `tapply()` to find the median weight (`bmxwt`) of each age group (`agegroup`), but do this only for the males (`gender`). One way to include the stratification by males is to use `with()` and the `with()` function's `data` argument. For example, using different variables:

```
with(data=subset(ii,hs=='No'),tapply(bpxsy1,bpcat,mean,na.rm=TRUE))
    Elevated Hypertension       Normal
    123.6530     136.7109     108.6307
```

7. Use `apply()` to find the median weight (`bmxwt`) and height (`bmxht`) of all individuals.
8. Use `apply()` to find the median weight (`bmxwt`) and height (`bmxht`) of all females (`gender`). Hint: you can use indexing or `subset()` in combination with the `apply()` function's `X` argument.
9. The `tapply()` function can be a useful way to summarize tall data. Use `reshape()` to create a tall version of the NHANES data, for the repeated measurements of diastolic blood pressure. You can follow the example beginning on slide 57 of Module 7 to do this. Order the reshaped data according to `seqn`, and then subset to just the first 24 rows. Now, use `tapply()` to find the mean diastolic blood pressure of each individual. You should obtain 6 values (each value represents the mean diastolic blood pressure of a unique individual).