Homework 1 Solutions

Stats 32: Introduction to R for Undergraduates

Due: Thursday, April 11, 2024, 11:59 am PT

Instructions

You may (in fact, are encouraged to) use the Internet (including AI assistants, like ChatGPT) to search up any information to help you with this assignment, though you must cite any external (i.e. non-course related) resources that you use. Similarly, *after attempting this assignment by yourself*, you may collaborate with other students in the course, but you must each write your own code and acknowledge all students with whom you collaborated *for each problem* (you don't need to cite by subpart). However, you may not post on Internet forums (e.g. Stack Exchange) for help with this assignment; doing so is considered an Honor Code violation. You also may not copy verbatim any significant amount of code from the Internet (including AI assistants, like ChatGPT), even with citation. Feeding in the problems directly into AI assistants (or substantively paraphrased version) is also not permitted.

Please provide your code responses to each problem in the `.Rmd` file in the R code chunks directly below each subpart, inserting additional R code chunks if needed. Any text response can go right underneath the corresponding question.

On Gradescope, please submit a single `.pdf` file created by knitting the document with your responses. Problem 0 will provide guidance on how to do this.

Credit is given based on the approach and code, not necessarily the final answer.

Problem 0

Read Sections 2.2-2.6 of R Markdown: The Definitive Guide. It will introduce you to using R markdown. You may skip subsection 2.5.3 if you are not familiar with LaTeX. It is not needed for this class.

Problem 1

(a) [1 point] Load the `nycflights13` package.

Answer:

```r
library(nycflights13)
```

(b) [2 points] Consider the `airports` tibble (data frame). It is automatically loaded into the environment when you complete part (a). How many airports are in the tibble?

Answer: Each row in `airports` is an observation, hence an airport. Thus, the answer is the number of rows in `airports`, or 1,458:

```r
nrow(airports)
```

```
## [1] 1458
```

(c) [2 points] What is the mean altitude of all the airports in the tibble? Give the units.

Answer: We start by looking at the help page for the tibble `airports`:

```
?airports
```

This shows us that the column `alt` refers to altitude in feet. We want to take the mean value of all numbers in this column. We can extract this column via indexing, and then use the built-in `mean()` function from Lecture 2:

```
mean(airports$alt)
```

```
## [1] 1001.416
```

(d) [2 points] How many different time zones are represented among the airports in the tibble? Hint: Look at the help page for `unique()`.

Answer: The help page for `unique()` tells us that this function gives us the unique values in a vector. From the help page for the `airports` tibble we know the time zones are given in the column labeled `tzone`. Thus the following code stores the unique time zones in the tibble:

```
unique_tzones <- unique(airports$tzone)
```

To get the number of unique time zones, we could print out our variable `unique_tzones`, and count the number of entries. Alternatively (better), we could do this programatically using the `length()` function:

```
length(unique_tzones)
```

```
## [1] 10
```

If we inspect `unique_tzones`, there seems to be an `NA` which we may not want to include, so it would actually be better to say the answer is in fact only 9:

```
unique_tzones
```

```
##  [1] "America/New_York"    "America/Chicago"     "America/Los_Angeles"
##  [4] "America/Vancouver"   "America/Phoenix"     "America/Anchorage"
##  [7] "America/Denver"      "Pacific/Honolulu"    "Asia/Chongqing"
## [10] NA
```

(e) [2 points] Select the `name`, `lat`, `lon`, and `alt` of the 3rd through 10th airports in the tibble. The result should be a 8 x 4 tibble.

Answer: First we recall that tibbles and data frames are named lists under the hood, where each entry of the list is a column and the name of the entry is the name of the column. So we can first select the relevant columns using the syntax for indexing into a named list:

```
airport_columns <- airports[c("name", "lat", "lon", "alt")]
```

Then we recall that data frames can be indexed as a matrix, so we now index into the proper rows with the notation of a matrix:

```
airport_columns[c(3,4,5,6,7,8,9,10),]
```

```
## # A tibble: 8 x 4
##   name                             lat    lon   alt
##   <chr>                          <dbl>  <dbl> <dbl>
## 1 Schaumburg Regional             42.0  -88.1   801
## 2 Randall Airport                 41.4  -74.4   523
## 3 Jekyll Island Airport           31.1  -81.4    11
## 4 Elizabethton Municipal Airport  36.4  -82.2  1593
## 5 Williams County Airport         41.5  -84.5   730
## 6 Finger Lakes Regional Airport   42.9  -76.8   492
## 7 Shoestring Aviation Airfield    39.8  -76.6  1000
## 8 Jefferson County Intl           48.1 -123.    108
```

(f) [2 points] How many airports are in the "America/New_York" time zone?

Answer: Recalling Lab 2, we convert `airports$tzone` to a factor and call the `summary()` function:

```r
summary(as.factor(airports$tzone))
```

```
##   America/Anchorage     America/Chicago      America/Denver America/Los_Angeles
##                 239                 342                 119                 176
##    America/New_York     America/Phoenix   America/Vancouver      Asia/Chongqing
##                 519                  38                   2                   2
##    Pacific/Honolulu                NA's
##                  18                   3
```

We see there are 519 airports in the "America/New_York" time zone.

Note: tidyverse solutions for (e) and (f) are probably cleaner and perfectly acceptable, but not used in our solutions here since this homework was designed to be completable using only the content from the first two lectures and labs.

Problem 2

(a) [4 points] Write a function called `percent_decrease` that takes in two arguments — `num` and `percent` — and returns the value when `num` is decreased by `percent` percent. For instance, if `num` is 10 and `percent` is 20, your function should return 8.

Answer:

```r
percent_decrease <- function(num, percent){
  return(num*(1-percent/100))
}
```

(b) [2 points] Use your function from (a) to calculate the cost of an item of clothing which originally sold for $49.95, but is on sale for 35% off. Use `round()` to print out the price to the nearest **cent**.

Answer: The help menu for `round()` tells us that the second argument specifies how many digits after the decimal point to round to. A cent has two decimal places, so our second argument needs to be 2:

```r
round(percent_decrease(num=39.95, percent=40), 2)
```

```
## [1] 23.97
```

Problem 3

(a) [4 points] Run the following three lines of code:

```r
scores <- c(89, 72, 96, 93, 100, 68, 82, 69, 77, 79, 78, 90)
x <- scores > 90
y <- scores[scores > 90]
```

Explain what kind of data structure `x` is, the dimension(s) of that data structure, and the data type(s) of the objects (variables) within that data structure. Repeat for `y`. Use code to justify your answers.

Answer: We start by examining the structure of `x` using the `str` function from Lab 1:

```r
str(x)
```

```
##  logi [1:12] FALSE FALSE TRUE TRUE TRUE FALSE ...
```

We see `x` is a logical vector of length 12. Next, we examine `y`:

```r
str(y)
```

```
##  num [1:3] 96 93 100
```

This is a numeric vector of length 3.

(b) [2 points] Suppose that `scores` is a vector of exam scores for students in a recent class. In light of this, interpret (in words) the content of the objects `x` and `y` above.

Answer: Based on what we printed in part (a), we see that `x` is a vector of logicals indicating whether the score of each student is greater than 90. Then `y` indexes into the vector of scores (`scores`) with the logical vector `x` and prints out only those entries of `scores` where the corresponding entry of `x` is TRUE. More succinctly, `y` is a vector containing all the scores greater than 90. The construction of `y` is known as **logical indexing**, and is a very important concept in R and similar languages.

(c) [2 points] Describe (in words) what the following line of code does.

```r
scores[-2]
```

```
##  [1]  89  96  93 100  68  82  69  77  79  78  90
```

Answer: We observe that `scores[-2]` prints out all the entries of `scores` *except* the 2nd one. This is called **negative indexing**.