Homework 02 CSCI 036 Solutions

Lucas Welch

Due: Friday, 2022-09-16

Instructions

Please box your answers. For numerical answers, this can be done using something like $\boxed{34}$. For text answers, this can be done using something like $\boxed{\text{My answer}}$. The output of a code chunk is automatically boxed, so no need to do more.

You will need the tidyverse package for this homework.

```
library(tidyverse)
library(dplyr)
```

Consider the following dataset.

```
simple_example <- tibble(
  change = c(-5, 3, 4, -1),
  season = c("Winter", "Summer", "Summer", "Fall")
)
simple_example
```

```
## # A tibble: 4 × 2
##   change season
##    <dbl> <chr>
## 1     -5 Winter
## 2      3 Summer
## 3      4 Summer
## 4     -1 Fall
```

a. Write code to order the observations by `change` in descending order.

b. Write code to only keep the variable `change` .

A.

```
simple_example |>
  arrange(desc(change))
```

```
## # A tibble: 4 × 2
##   change season
##    <dbl> <chr>
## 1      4 Summer
## 2      3 Summer
## 3     -1 Fall
## 4     -5 Winter
```

B.

```
simple_example |>
  select(change)
```

```
## # A tibble: 4 × 1
##   change
##    <dbl>
## 1     -5
## 2      3
## 3      4
## 4     -1
```

Using the `simple_example` from the last problem, do the following.

a. Write code to add a variable `abs_change` that is the absolute value of the `change` value.

b. Write code to add a variable `positive` which has value TRUE if the value of `change` is greater than 0, and FALSE to otherwise.

A.

```
simple_example |>
  mutate(abs_change=abs(change))
```

```
## # A tibble: 4 × 3
##   change season abs_change
##    <dbl> <chr>       <dbl>
## 1     -5 Winter          5
## 2      3 Summer          3
## 3      4 Summer          4
## 4     -1 Fall            1
```

B.

```
simple_example |>
  mutate(positive = (change >0))
```

```
## # A tibble: 4 × 3
##   change season positive
##    <dbl> <chr>  <lgl>
## 1     -5 Winter FALSE
## 2      3 Summer TRUE
## 3      4 Summer TRUE
## 4     -1 Fall   FALSE
```

Consider the `mpg` dataset which is in the ggplot2 package. This package can be loaded with the following code.

```
library(ggplot2)
```

Currently engine displacement in `mpg` is measured in liters. Convert this to cubic centimeters with the mutate command.

```
mpg |>
  select(displ) |>
  mutate(L_to_Cnt3 = displ*1000)
```

```
## # A tibble: 234 × 2
##    displ L_to_Cnt3
##    <dbl>     <dbl>
##  1   1.8      1800
##  2   1.8      1800
##  3   2        2000
##  4   2        2000
##  5   2.8      2800
##  6   2.8      2800
##  7   3.1      3100
##  8   1.8      1800
##  9   1.8      1800
## 10   2        2000
## # … with 224 more rows
```

The median command in R calculates the *sample median* of a dataset. This is the middle value in a vector of values if the length of the vector is odd, and the arithmetic average of the two middle values in a vector of values if the length of the vector is even.

For instance,

```
median(c(3, 7, 17))
```

```
## [1] 7
```

and

```
median(c(3, 7, 10, 17))
```

```
## [1] 8.5
```

illustrates this sample median.

Use this command together with summarize to find the sample median of the `mpg` variable in the `mtcars` dataset built into R.

```
summarize(mtcars, median(c(mpg)))
```

```
##   median(c(mpg))
## 1           19.2
```

Use the summarize command to create a tibble that contains the average mpg, the median mpg, and the average of the `wt` variable that measures the weight of the vehicle in thousands of pounds.

```
summarize(mtcars, mean(c(wt)), mean(c(mpg)), median(c(mpg)))
```

```
##   mean(c(wt)) mean(c(mpg)) median(c(mpg))
## 1     3.21725     20.09062           19.2
```

Given a vector that consists of boolean values, TRUE and FALSE, when you use sum, every TRUE gets turned into a 1, and every FALSE into a 0.

The rest of this problem uses

```
x <- c(TRUE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE, TRUE, FALSE)
```

a. Try applying sum to this vector `x`.

b. Try applying mean to this vector `x`.

c. Try applying max to this vector `x`.

d. Try applying min to this vector `x`.

a.

```
sum(c(x==1, 0))
```

```
## [1] 6
```

b.

```
mean(c(x==1, 0))
```

```
## [1] 0.5454545
```

c.

```
max(c(x==1, 0))
```

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

d.

```
min(c(x==1, 0))
```

```
## [1] 0
```

Consider the variable `flights` in the package nycflights13. When `arr_delay` is zero or negative, say that a particular flight is on time.

a. Use mutate to add a new boolean that indicates whether or not a flight is on-time.

b. What percentage of the flights were on time?

a.

```
library("nycflights13")
flights |>
  mutate(flights, on_time=(arr_delay<0))
```

```
## # A tibble: 336,776 × 20
##     year month   day dep_time sched_de…¹ dep_d…² arr_t…³ sched…⁴ arr_d…⁵ carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # … with 336,766 more rows, 10 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, on_time <lgl>, and abbreviated variable
## #   names ¹sched_dep_time, ²dep_delay, ³arr_time, ⁴sched_arr_time, ⁵arr_delay
```

b.

```
flights |>
  summarize(on_time=(arr_delay<0)) |>
  filter(on_time == "TRUE")|>
  sum(c(TRUE))
```

```
## [1] 188934
```

```
percentage <- (188934/336776)*100
percentage
```

```
## [1] 56.10079
```

The `InsectSprays` dataset gives the counts of insects found in different fields sprayed with one of five different insecticides.

a. Find the maximum number of insects in all 72 fields with summarize.

b. Find the average number of insects in all 72 fields with summarize.

c. Find the ratio between the max number and the average number of insects over all 72 fields with summarize.

a.

```
InsectSprays |>
  summarize(max_insects = max(count))
```

```
##   max_insects
## 1          26
```

b.

```
InsectSprays |>
 summarize(avg_insects = mean(count))
```

```
##   avg_insects
## 1         9.5
```

c.

```
InsectSprays |>
  summarize(max_ratio = (26/72))
```

```
##   max_ratio
## 1 0.3611111
```

```
InsectSprays |>
 summarize(avg_ratio = (9.5/72))
```

```
##   avg_ratio
## 1 0.1319444
```

The dataset `rivers` contains the lengths of 141 rivers in the United States. This can be viewed as a tibble using the following code.

```
us_rivers <- tibble(length = rivers)
```

Use arrange and slice to find the length of the 45th longest river.

```
us_rivers |>
  select(length) |>
  arrange(desc(length))|>
  slice(45)
```

```
## # A tibble: 1 × 1
##   length
##    <dbl>
## 1    600
```

The data set `uspop` is a *time series* data type that holds the results of the United States Census from 1790 to 1970. You can convert it to a tibble using the tibble function.

```
us_census <- tibble(uspop)
```

a. Add to the tibble a `year` variable that runs from 1790 to 1970 skipping by ten years.

b. Add to the part a tibble a variable `log_pop` that shows the natural logarithm of the population.

a.

```
us_census |>
  mutate(year = seq(1790, 1970, by = 10))
```

```
## # A tibble: 19 × 2
##     uspop  year
##     <dbl> <dbl>
##  1   3.93  1790
##  2   5.31  1800
##  3   7.24  1810
##  4   9.64  1820
##  5  12.9   1830
##  6  17.1   1840
##  7  23.2   1850
##  8  31.4   1860
##  9  39.8   1870
## 10  50.2   1880
## 11  62.9   1890
## 12  76     1900
## 13  92     1910
## 14 106.    1920
## 15 123.    1930
## 16 132.    1940
## 17 151.    1950
## 18 179.    1960
## 19 203.    1970
```

b.

```
us_census |>
  mutate(year = seq(1790, 1970, by = 10)) |>
  mutate(log_pop = log(uspop))
```

```
## # A tibble: 19 × 3
##     uspop  year log_pop
##     <dbl> <dbl>   <dbl>
##  1   3.93  1790    1.37
##  2   5.31  1800    1.67
##  3   7.24  1810    1.98
##  4   9.64  1820    2.27
##  5  12.9   1830    2.56
##  6  17.1   1840    2.84
##  7  23.2   1850    3.14
##  8  31.4   1860    3.45
##  9  39.8   1870    3.68
## 10  50.2   1880    3.92
## 11  62.9   1890    4.14
## 12  76     1900    4.33
## 13  92     1910    4.52
## 14 106.    1920    4.66
## 15 123.    1930    4.81
## 16 132.    1940    4.88
## 17 151.    1950    5.02
## 18 179.    1960    5.19
## 19 203.    1970    5.31
```