Homework 5

Brady Lamson

3/12/2022

PDF PROBLEMS

Problem 1

```
setwd("/home/brady/repos/mth_3270_data_science/module_5/hw_5")
houses <- read.csv("houses-for-sale.txt", header = TRUE, sep = "\t")
translations <- read.csv("house_codes.txt", header = TRUE, sep = "\t")

houses_small <- select(houses, fuel, heat, sewer, construction)

codes <- translations %>%
    tidyr::pivot_wider(
        names_from = system_type,
        values_from = meaning,
        values_fill = "invalid"
    )
```

```
# A
# Join in codes df based on each type of code
# Then select only those code columns to remove the integer columns
houses_small_coded <- houses_small %>%

    dplyr::left_join(
        codes %>%
            dplyr::select(code, fuel_type),
            by = c(fuel = "code")
    ) %>%
    dplyr::left_join(
        codes %>%
            dplyr::select(code, heat_type),
            by = c(heat = "code")
    ) %>%
    dplyr::left_join(
        codes %>%
            dplyr::select(code, sewer_type),
            by = c(sewer = "code")
    ) %>%
    dplyr::left_join(
        codes %>%
```

```
            dplyr::select(code, new_const),
            by = c(construction = "code")
    ) %>%
    dplyr::select(fuel_type, heat_type, sewer_type, new_const)

houses_small_coded %>% head()
```

```
##   fuel_type heat_type sewer_type new_const
## 1  electric  electric    private        no
## 2       gas hot water    private        no
## 3       gas hot water     public        no
## 4       gas   hot air    private        no
## 5       gas   hot air     public       yes
## 6       gas   hot air    private        no
```

```
arrange(summarize(group_by(select(filter(houses_small_coded, new_const == "no"),
fuel_type, heat_type), fuel_type), count = n()), desc(count))
```

```
## # A tibble: 3 x 2
##   fuel_type count
##   <chr>     <int>
## 1 gas        1117
## 2 electric    314
## 3 oil         216
```

This command does the following:

First, it **filters** out only the rows with **NO** new construction. Then, we **select** the fuel_type and heat_type columns, ignoring all the others. After that, we **group by** the type of fuel. Then we **summarize** this data frame by the **count** of each **type** of fuel and we **order** those counts in **descending** order.

```
houses_small_coded %>%
    dplyr::filter(
        new_const == "no"
    ) %>%
    dplyr::select(fuel_type, heat_type) %>%
    dplyr::group_by(fuel_type) %>%
    dplyr::summarise(count = n()) %>%
    dplyr::arrange(dplyr::desc(count))
```

```
## # A tibble: 3 x 2
##   fuel_type count
##   <chr>     <int>
## 1 gas        1117
## 2 electric    314
## 3 oil         216
```

Problem 2

```r
flights <- nycflights13::flights
```

```r
# Group by destination and get total and average minutes of delay
flights %>%
    dplyr::group_by(dest) %>%
    dplyr::summarise(
        total_delay = sum(dep_delay, arr_delay, na.rm = TRUE),
        average_delay = c(dep_delay, arr_delay) %>%
            mean(na.rm = TRUE) %>%
            round(digits = 3)
    ) %>%
    dplyr::arrange(
        dplyr::desc(average_delay)
    )
```

```
## # A tibble: 105 x 3
##    dest  total_delay average_delay
##    <chr>       <dbl>         <dbl>
##  1 CAE          8233          38.7
##  2 TUL         20333          34.3
##  3 OKC         19641          30.6
##  4 JAC          1174          27.3
##  5 TYS         30410          26.3
##  6 BHM         12617          23.3
##  7 DSM         23791          22.6
##  8 MSN         24481          21.9
##  9 RIC        102711          21.9
## 10 CAK         34138          20.3
## # ... with 95 more rows
```

Problem 3

```r
planes <- nycflights13::planes

# Using some hacky tricks we can figure out which column names match automatically
names(flights)[which(names(flights) %in% names(planes))]
```

```
## [1] "year"    "tailnum"
```

From this we can see that 'year' and 'tailnum' are our two candidates. **Year** is, based purely on intuition, probably not a good option. Year is tied to the plane in the planes data set, but not the flights data set. The year represents totally different things in each. Thankfully **tailnum** is tied to the tail number in both data sets so we can utilize that. I feel using **inner_join** should work out just fine as that will remove rows without a proper tail number and, by extension, those that lack the manufacturer information we need.

```r
flights %>%
    dplyr::inner_join(
        planes,
        by = 'tailnum'
    ) %>%
    dplyr::group_by(manufacturer) %>%
    dplyr::summarise(count = n()) %>%
    dplyr::arrange(dplyr::desc(count))
```

```
## # A tibble: 35 x 2
##    manufacturer                  count
##    <chr>                         <int>
##  1 BOEING                        82912
##  2 EMBRAER                       66068
##  3 AIRBUS                        47302
##  4 AIRBUS INDUSTRIE              40891
##  5 BOMBARDIER INC                28272
##  6 MCDONNELL DOUGLAS AIRCRAFT CO  8932
##  7 MCDONNELL DOUGLAS              3998
##  8 CANADAIR                       1594
##  9 MCDONNELL DOUGLAS CORPORATION  1259
## 10 CESSNA                          658
## # ... with 25 more rows
```

What we can see from this is that **Boeing** made the most flights with a count of **82912** flights to its name.

Textbook Problems

Chapter 5 Problem 3

- How many planes have a missing date of manufacture?

```
planes %>%
    dplyr::filter(is.na(year)) %>%
    dplyr::summarise(count = n()) %>%
    paste()
```

```
## [1] "70"
```

From this we can say that 70 of the planes in the planes data set are missing a data of manufacture.

- What are the five most common manufactures?

```
# We group by the manufacturer, count up the number for each
# Sort from most common to least and then
# extract the first 5 rows
planes %>%
    dplyr::group_by(manufacturer) %>%
    dplyr::summarise(count = n()) %>%
    dplyr::arrange(
        dplyr::desc(count)
    ) %>%
    # Extract only the top 5 rows
    dplyr::top_n(5)
```

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   manufacturer     count
##   <chr>            <int>
## 1 BOEING            1630
## 2 AIRBUS INDUSTRIE   400
## 3 BOMBARDIER INC     368
## 4 AIRBUS             336
## 5 EMBRAER            299
```

The 5 most common manufacturers are Boeing, airbus industrie, bombardier, airbus and embraer.

Chapter 5 Problem 4

- What is the oldest plane that flew from NYC airports in 2013?

For this we want to combine the planes and flights data sets again. We can combine a smaller version though as we are only concerned with flights done in 2013.

```
flights %>%
    dplyr::filter(year == 2013) %>%
    # Rename year to flight year so we can keep the planes year column
    dplyr::rename(flight_year = year) %>%
    dplyr::left_join(planes, by = 'tailnum') %>%
    dplyr::select(tailnum, year) %>%
    dplyr::filter(year == min(year, na.rm = TRUE))
```

```
## # A tibble: 22 x 2
##    tailnum  year
##    <chr>   <int>
##  1 N381AA   1956
##  2 N381AA   1956
##  3 N381AA   1956
##  4 N381AA   1956
##  5 N381AA   1956
##  6 N381AA   1956
##  7 N381AA   1956
##  8 N381AA   1956
##  9 N381AA   1956
## 10 N381AA   1956
## # ... with 12 more rows
```

This shows us that the oldest plane is N381AA that was created in 1956.

Chapter 6 Problem 2

Rewrite the given command using a single line nested form.

```
mtcars %>%
  filter(cyl == 4) %>%
  select(mpg, cyl)
```

```
##                 mpg cyl
## Datsun 710     22.8   4
## Merc 240D      24.4   4
## Merc 230       22.8   4
## Fiat 128       32.4   4
## Honda Civic    30.4   4
## Toyota Corolla 33.9   4
## Toyota Corona  21.5   4
## Fiat X1-9      27.3   4
## Porsche 914-2  26.0   4
## Lotus Europa   30.4   4
## Volvo 142E     21.4   4
```

```
select(filter(mtcars, cyl == 4), mpg, cyl)
```

```
##                 mpg cyl
## Datsun 710     22.8   4
## Merc 240D      24.4   4
## Merc 230       22.8   4
## Fiat 128       32.4   4
## Honda Civic    30.4   4
## Toyota Corolla 33.9   4
## Toyota Corona  21.5   4
## Fiat X1-9      27.3   4
## Porsche 914-2  26.0   4
## Lotus Europa   30.4   4
## Volvo 142E     21.4   4
```

I definitely prefer the pipe format, the other format becomes unreadable very quickly.

Chapter 6 Problem 3

```r
x1 <- c("1900.45", "$1900.45", "1,900.45", "nearly $2000")
x2 <- as.factor(x1)

readr::parse_number(x1)
```

```
## [1] 1900.45 1900.45 1900.45 2000.00
```

```r
#readr::parse_number(x2)

as.numeric(x1)
```

```
## Warning: NAs introduced by coercion
```

```
## [1] 1900.45      NA      NA      NA
```

```r
as.numeric(x2)
```

```
## [1] 3 1 2 4
```

The first command returns the expected numbers you'd be looking for. The second command throws an error though. Looking in the documentation for readr::parse_number() clears up why. parse_number() takes in a character vector, which a vector of factors is not. We see the opposite problem with as.numeric(). With that one we get a reasonable 1900.45 value followed by 3 NA's. This is because of the not "numeric" parts of the string confusing it, like the commas and dollar signs. Whereas x2 works fine because factors are just integers under the hood.

Chapter 6 Problem 5

```r
my.data <- data.frame(grp = rep(c("A", "B"), each = 2),
                      sex = rep(c("F", "M"), times = 2),
                      meanL = c(0.22, 0.47, 0.33, 0.55),
                      sdL = c(0.11, 0.33, 0.11, 0.31),
                      meanR = c(0.34, 0.57, 0.40, 0.65),
                      sdR = c(0.09, 0.33, 0.07, 0.27))
```

```r
# my.data %>%
#     tidyr::pivot_longer(cols = c(meanL, sdL, meanR, sdR), values_to = 'sex')
#
#
# tidyr::pivot_wider(names_from = grp,
#                         values_from = sex)
```

Chapter 6 Problem 7

```r
ds1 <- data.frame(id = rep(1:3, times = 2),
                  group = rep(c("T", "C"), each = 3),
                  vals = c(4, 6, 8, 5, 6, 10))

ds1
```

```
##   id group vals
## 1  1     T    4
## 2  2     T    6
## 3  3     T    8
## 4  1     C    5
## 5  2     C    6
## 6  3     C   10
```

The big problem with the below approach is that it assumes a consistent ordering for id. If, for instance, the ordering for ID got messed up somehow and was instead "3, 2, 1" for whatever reason this code would provide inaccurate results. This would also break down is a value was to be removed for whatever reason. It is entirely dependent on ordering being consistent which isn't something you can rely on with larger messier data sets.

```r
Treat <- filter(ds1, group == "T")
Control <- filter(ds1, group == "C")
all <- mutate(Treat, diff = Treat$vals - Control$vals)
all
```

```
##   id group vals diff
## 1  1     T    4   -1
## 2  2     T    6    0
## 3  3     T    8   -2
```

```r
ds1 %>%
    tidyr::pivot_wider(
        names_from = group,
        values_from = vals
    ) %>%
    dplyr::mutate(
        diff = T - C
    )
```

```
## # A tibble: 3 x 4
##      id     T     C  diff
##   <int> <dbl> <dbl> <dbl>
## 1     1     4     5    -1
## 2     2     6     6     0
## 3     3     8    10    -2
```