Homework 5

MTH 3270 Data Science

Tobias Boggess

3/9/2022

Chapter 5 Worksheet Problems

Problem 1: Do the following.

Data:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
myURL <- "http://sites.msudenver.edu/ngrevsta/wp-content/
uploads/sites/416/2021/02/houses-for-sale.txt"
Houses <- read.csv(myURL, header = TRUE, sep = "\t")

Houses_small <- select(Houses, fuel, heat, sewer, construction)

myURL <- "http://sites.msudenver.edu/ngrevsta/wp-content/uploads/
sites/416/2021/02/house_codes.txt"
Translations <- read.csv(myURL,
                         header = TRUE,
                         stringsAsFactors = FALSE,
                         sep = "\t")

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

```
Houses_small <- left_join(
  x = Houses_small,
  y = select(codes, code, fuel_type),
  by = c(fuel = "code")
)
```

a) Report R commands that recode the remaining variables (heat, sewer, construction) in Houses_small, then remove the original (integer-valued) variables. Code:

```
Houses_small <-
  left_join(x = Houses_small,
            y = select(codes, code, heat_type),
            by = c(heat = "code"))
Houses_small <-
  left_join(x = Houses_small,
            y = select(codes, code, sewer_type),
            by = c(sewer = "code"))
Houses_small <-
  left_join(x = Houses_small,
            y = select(codes, code, new_const),
            by = c(construction = "code"))

head(select(Houses_small, ends_with("type"), new_const))
```

```
##   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
```

b) Now (using Houses_small obtained in Part a), describe in words what the following command does. Then rewrite it into a more readable version using the pipe operator %>%. Original code:

```
arrange(summarize(group_by(
  select(filter(Houses_small, 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
```

Pipe Operator:

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

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

Problem 2: Using the flights data set (from the "nycflights13" package), for each destination (dest), determine the total minutes of delay and the average minutes of delay. Report your R command(s).

Code:

```
library(nycflights13)
flights_small <- group_by(select(flights, dest, dep_delay, arr_delay), dest)
summarise(.data = flights_small, delay_sum = sum(dep_delay, na.rm = TRUE) + sum(arr_delay, na.rm = TRUE
```

```
## # A tibble: 105 x 3
##    dest  delay_sum delay_mean
##    <chr>     <dbl>      <dbl>
##  1 ABQ        4603       18.1
##  2 ACK        2992       11.3
##  3 ALB       15915       38.0
##  4 ANC          83       10.4
##  5 ATL      401651       23.8
##  6 AUS       46010       19.0
##  7 AVL        4243       16.2
##  8 BDL       10205       24.8
##  9 BGR        9885       27.5
## 10 BHM       12617       46.6
## # ... with 95 more rows
```

Problem 3: Using the flights dataset and answer the following.

a) Which variable would be the key for combining the two data frames using one of the *_join() functions?
The key variable to use when using *_join() is the tailnum variable in the flights data set.

b) Combine the flights and planes data sets using an appropriate *__join() function. Which manufacturer made the most flights in 2013? How many flights did it make? Code:

```
combined_flights <- left_join(x = flights, y = planes, by = 'tailnum')
#View(combined_flights)

grp_combined_flights <- group_by(.data = combined_flights, manufacturer)

sumgrp_comb_flights <- summarise(grp_combined_flights, count = n())
arrange(.data = sumgrp_comb_flights, desc(count))
```

```
## # A tibble: 36 x 2
##    manufacturer                  count
##    <chr>                         <int>
##  1 BOEING                        82912
##  2 EMBRAER                       66068
##  3 <NA>                          52606
##  4 AIRBUS                        47302
##  5 AIRBUS INDUSTRIE              40891
##  6 BOMBARDIER INC                28272
##  7 MCDONNELL DOUGLAS AIRCRAFT CO  8932
##  8 MCDONNELL DOUGLAS              3998
##  9 CANADAIR                       1594
## 10 MCDONNELL DOUGLAS CORPORATION  1259
## # ... with 26 more rows
```

The manufacturer with the most amount of flights made in 2013 was Boeing and it made 82,912 flights in that year.

Chapter 5 Book Problems

Problem 3 with c: Answer the following questions about the flights dataset

a) How many planes have a missing date of manufacture? Code:

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

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1    70
```

There are 70 airplanes that don't have a date of manufacturer.

b) What are the five most common manufacturers? Code:

```
planes %>% group_by(manufacturer) %>% summarise(count = n()) %>% arrange(desc(count))
```

```
## # A tibble: 35 x 2
##    manufacturer                  count
##    <chr>                         <int>
##  1 BOEING                         1630
##  2 AIRBUS INDUSTRIE                400
##  3 BOMBARDIER INC                  368
##  4 AIRBUS                          336
##  5 EMBRAER                         299
##  6 MCDONNELL DOUGLAS               120
##  7 MCDONNELL DOUGLAS AIRCRAFT CO   103
##  8 MCDONNELL DOUGLAS CORPORATION    14
##  9 CANADAIR                          9
## 10 CESSNA                            9
## # ... with 25 more rows
```

The five most popular manufacturers are Boeing, Airbus Industrie, Bombardier Incorporation, Airbus, and Embraer.

Problem 4: Answer the following questions about the flights dataset.

a) What is the oldest plane (specified by the tailnum variable) that flew from New York City airports in 2013? Code:

```
combined_flights %>% filter(!is.na(year.y)) %>% summarise(min(year.y))
```

```
## # A tibble: 1 x 1
##   `min(year.y)`
##           <int>
## 1          1956
```

- above code uses the combined data between flights and planes data sets from an earlier question * The oldest plane to fly out of New York in 2013 was manufactured in 1956.

b) How many airplanes that flew from New York City are included in the planes table? Code:

```
combined_flights %>% filter(!is.na(manufacturer)) %>% summarise(count = n())
```

```
## # A tibble: 1 x 1
##    count
##    <int>
## 1 284170
```

```
combined_flights %>% filter(is.na(manufacturer)) %>% summarise(count = n())
```

```
## # A tibble: 1 x 1
##   count
##   <int>
## 1 52606
```

There are a total of 284,170 flights out of New York that are included in the planes data set.

Chapter 6 Problems

Exercise 2: Consider the following pipeline.

Pipeline:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.6     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
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
```

Rewrite this in nested form on a single line. Which set of commands do you prefer and why?

Non-pipeline version:

```r
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
```

The pipeline version is easier to read so I would choose that version of the code. Another thing I like about the pipeline version is I know what the data set thats being used and the parameters are easier to obtain.

Problem 3: Consider the values returned by the as.numeric() and parse_number() functions when applied to the following vectors. Describe the results and their implication.

Code:

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

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

```r
#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 function parse_number(x2) produces an error because when x2 factors x1, it will turn each string into a numeric representation of each string. The function parse_number() will parse a number as long as there are no special characters involved such as $, ',', and 'nearly $'. Given the as.numeric() function returns only one of the values for x1 and all the values of x2 shows the above is true.

Problem 5: Generate the code to convert the following data frame to wide format.

Code:

```
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)
)
my.data %>% pivot_longer(cols = meanL:sdR) %>%
  pivot_wider(names_from = c(sex, name), values_from = value)
```

```
## # A tibble: 2 x 9
##   grp   F_meanL F_sdL F_meanR F_sdR M_meanL M_sdL M_meanR M_sdR
##   <chr>   <dbl> <dbl>   <dbl> <dbl>   <dbl> <dbl>   <dbl> <dbl>
## 1 A        0.22  0.11    0.34  0.09    0.47  0.33    0.57  0.33
## 2 B        0.33  0.11    0.4   0.07    0.55  0.31    0.65  0.27
```

```
#my.data <- my.data %>% pivot_wider(names_from = name, values_from = value)
```

Problem 7: Verify that this code works for this example and generates the correct values of −1, 0, and −2. Describe two problems that might arise if the data set is not sorted in a particular order or if one of the observations is missing for one of the subjects. Provide an alternative approach to generate this variable that is more robust (hint: use pivot_wider).

Code:

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

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
```

One of the problems with the above code is if it is not sorted, the results from the difference formula would produce the wrong result since filter doesn't account for the id value. Another possible issue with the above code is if one of the observations is missing, the difference result would show an NA value in the data. Alternative

```
ds1 <- ds1 %>% pivot_wider(names_from = id, values_from = vals)
ds1
```

```
## # A tibble: 2 x 4
##   group   `1`   `2`   `3`
##   <chr> <dbl> <dbl> <dbl>
## 1 T         4     6     8
## 2 C         5     6    10
```