Data Science Module 3 Exercises

Brady Lamson

2/14/2022

4: Data Wrangling

4.2:

Extracting Columns with select()

Exercise 1

a) This will pull only the years and days from the flights data set.

b) This pulls all the columns between year and day, inclusive.

c) This pulls all columns *except* year and day.

Exercise 2:

a) This will pull all column names starting with "sched", of which there are 2 columns in this data set. (sched_dep_time and sched_arr_time)

b) Same as (a) but with columns starting with "arr" (arr_time and arr_delay).

c) Pulls all columns starting either with "dep_" or "arr_".

4.4: Filtering Rows with filter()

Exercise 3:

```
# A
dplyr::filter(flights, arr_delay >= 120)
```

```
# B
flights %>%
    dplyr::filter(dest %in% c("IAH", "HOU"))
```

```
# C
flights %>%
    dplyr::filter(carrier %in% c("UA", "AA", "DL"))
```

```
# D
flights %>%
    dplyr::filter(month %in% 7:9)
```

```
# E
flights %>%
    dplyr::filter(dep_time %in% 0:600)
```

```
# F
flights %>%
    dplyr::filter(
        carrier == "UA",
        month == 7,
        arr_delay >= 120
    )
```

4.5: Arranging Rows with arrange()

Exercise 4:

```
# A
flights %>%
    dplyr::arrange(dep_delay, arr_delay)
```

```
# B
flights %>%
    dplyr::arrange(
        dplyr::desc(dep_delay),
        dplyr::desc(arr_delay)
    )
```

```
# C
flights %>%
    dplyr::arrange(dep_time)
```

```
# D
flights %>%
    dplyr::arrange(dplyr::desc(dep_time))
```

```
# E
flights %>%
    dplyr::arrange(distance)
```

```
# F
flights %>%
    dplyr::arrange(dplyr::desc(distance))
```

Exercise 5:

```
x <- data.frame(x1 = c(2, 1, NA, 8, 7, 5, 4),
x2 = c("a", NA, "c", "d", "c", "a", "d"),
stringsAsFactors = FALSE)
```

The following code will sort the data with NAs at the top of the *first* column.

```
x %>%
    dplyr::arrange(is.na(x1))
```

```
##   x1   x2
## 1  2    a
## 2  1 <NA>
## 3  8    d
## 4  7    c
## 5  5    a
## 6  4    d
## 7 NA    c
```

The following code will sort the data with NAs at the bottom of the *first* column.

```
x %>%
    dplyr::arrange(
        dplyr::desc(is.na(x1))
    )
```

```
##   x1   x2
## 1 NA    c
## 2  2    a
## 3  1 <NA>
## 4  8    d
## 5  7    c
## 6  5    a
## 7  4    d
```

4.6: Create New Variables (Columns) with mutate()

Exercise 6:

```
flights %>%
    dplyr::mutate(travel_time = arr_time - dep_time)
```

The arrival and departure time use totally different units than air time. The former are units of time, so 600 is 6:00am, whereas with air time 600 would be 600 minutes. They aren't equivalent.

Exercise 7:

```
flights_small <- select(
    .data = flights,
    year:day,
    ends_with("delay"),
    distance,
    air_time)
```

```
dplyr::mutate(
    .data = flights_small,
    gain = dep_delay - arr_delay,
    hours = air_time / 60,
    gain_per_hour = gain / hours
    )
```

```
## # A tibble: 336,776 x 10
##     year month   day dep_delay arr_delay distance air_time  gain hours
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>
##  1  2013     1     1         2        11     1400      227    -9 3.78
##  2  2013     1     1         4        20     1416      227   -16 3.78
##  3  2013     1     1         2        33     1089      160   -31 2.67
##  4  2013     1     1        -1       -18     1576      183    17 3.05
##  5  2013     1     1        -6       -25      762      116    19 1.93
##  6  2013     1     1        -4        12      719      150   -16 2.5
##  7  2013     1     1        -5        19     1065      158   -24 2.63
##  8  2013     1     1        -3       -14      229       53    11 0.883
##  9  2013     1     1        -3        -8      944      140     5 2.33
## 10  2013     1     1        -2         8      733      138   -10 2.3
## # ... with 336,766 more rows, and 1 more variable: gain_per_hour <dbl>
```

The variable gain_per_hour does get computed!

4.7: Renaming Variables (Columns) with rename()

Exercise 8:

```
z <- data.frame(
    z1 = c(5, 4, 3),
    z2 = c("a", "c", "b"),
    z3 = c(14, 22, 13))
```

```
z %>%
    dplyr::rename(
        new_z1 = z1,
        new_z2 = z2,
        new_z3 = z3
    )
```

```
##   new_z1 new_z2 new_z3
## 1      5      a     14
## 2      4      c     22
## 3      3      b     13
```

4.8: Summarize Data with summarize()

```
not_cancelled <- filter(.data = flights, !is.na(dep_delay), !is.na(arr_delay))
```

Exercise 9:

```
#A

not_cancelled %>%
    dplyr::summarise(
        median_dep_delay = median(dep_delay),
        median_arr_delay = median(arr_delay)
    )
```

```
## # A tibble: 1 x 2
##   median_dep_delay median_arr_delay
##              <dbl>            <dbl>
## 1               -2               -5
```

```
# B
not_cancelled %>%
    dplyr::summarise(
        max_dep_delay = max(dep_delay),
        max_arr_delay = max(arr_delay)
    )
```

```
## # A tibble: 1 x 2
##   max_dep_delay max_arr_delay
##           <dbl>         <dbl>
## 1          1301          1272
```

```
not_cancelled %>%
    dplyr::summarise(
        shortest_dep_delay = min(dep_delay),
        shortest_arr_delay = min(arr_delay)
    )
```

```
## # A tibble: 1 x 2
##   shortest_dep_delay shortest_arr_delay
##                <dbl>              <dbl>
## 1                -43                -86
```

Exercise 10:

a) The following code will give you the total count of rows in the data set.

```
summarize(.data = not_cancelled,
          total_flights = n())
```

```
## # A tibble: 1 x 1
##   total_flights
##           <int>
## 1        327346
```

b) This counts the *number* of flights that have a delay of an hour or more.

```
summarize(.data = not_cancelled,
          hour_arr_delay_total = sum(arr_delay > 60))
```

```
## # A tibble: 1 x 1
##   hour_arr_delay_total
##                  <int>
## 1                27789
```

c) This returns the ratio of flights with a delay of an hour or more compared to the total number of flights in the data set.

```
summarize(.data = not_cancelled,
          hour_arr_delay_proportion = sum(arr_delay > 60) / n())
```

```
## # A tibble: 1 x 1
##   hour_arr_delay_proportion
##                       <dbl>
## 1                    0.0849
```

4.9: Applying summarize() to Groups using group_by()

Exercise 11:

```
# A

by_dest <- group_by(.data = flights, dest)

delay_by_dest <- summarize(
    .data = by_dest,
    mean_arr_delay = mean(arr_delay, na.rm = TRUE)
    )
```

delay_by_dest gives a breakdown of the average delay by the location of travel.

```
# B
by_dest <- group_by(.data = flights, dest)

delay_dist_by_dest <- summarize(
    .data = by_dest,
    mean_dist = mean(distance, na.rm = TRUE),
    mean_arr_delay = mean(arr_delay, na.rm = TRUE)
)
```