Assignment 6

Kornel Kovacs

10/17/2019

Exercise 1

1, The number of flights (in the whole year) to each destination

```
dest_counter <-
  flights %>%
  group_by(dest) %>%
  summarize(dest_count = n())

top_n(dest_counter, 10)
```

```
## Selecting by dest_count
```

```
## # A tibble: 10 x 2
##    dest  dest_count
##    <chr>      <int>
##  1 ATL        17215
##  2 BOS        15508
##  3 CLT        14064
##  4 DCA         9705
##  5 FLL        12055
##  6 LAX        16174
##  7 MCO        14082
##  8 MIA        11728
##  9 ORD        17283
## 10 SFO        13331
```

2, The number and list of distinct airports in the US

I think the number of entries should be the same with `faa` and `name` as well, but they are not. No idea why.

```
dist_airports_acc_to_faa <- distinct(filter(airports,
                                            grepl('America', tzone)),
                                     faa)
dist_airports_acc_to_name <- distinct(filter(airports,
                                             grepl('America', tzone)),
                                      name)

top_n(dist_airports_acc_to_faa, 10)
```

```
## Selecting by faa
```

```
## # A tibble: 10 x 1
##    faa
##    <chr>
```

```
##  1 ZRZ
##  2 ZSF
##  3 ZSY
##  4 ZTF
##  5 ZTY
##  6 ZUN
##  7 ZVE
##  8 ZWI
##  9 ZWU
## 10 ZYP
```

```
top_n(dist_airports_acc_to_name, 10)
```

```
## Selecting by name
```

```
## # A tibble: 10 x 1
##    name
##    <chr>
##  1 Youngstown Elser Metro Airport
##  2 Yellowstone Rgnl
##  3 Yeager
##  4 Yolo County Airport
##  5 Zachar Bay Seaplane Base
##  6 Yuba County Airport
##  7 Zamperini Field Airport
##  8 Yellowstone Airport
##  9 Youngstown Warren Rgnl
## 10 Yuma Mcas Yuma Intl
```

```
nrow(dist_airports_acc_to_faa)
```

```
## [1] 1435
```

```
nrow(dist_airports_acc_to_name)
```

```
## [1] 1418
```

3, The number airports that are further south than NYC (Hint: look up longitude and latitude.)

New York City, NY, USA

Latitude and longitude coordinates are: **40.730610, -73.935242**.

We are only looking for airports to the south, therefore we only need longitude.

```
nyc_lon <- -73.935242
south_from_nyc <- filter(airports,
                         lon < nyc_lon)
nrow(south_from_nyc)
```

```
## [1] 1373
```

4, The top 5 carriers that have the lowest average delay times.

For further analysis, we should include the number of flights for each carrier. As it was not defined I worked with arrival delays rather than departure ones as they are more relevant to customers.

```
(airlines_on_time <- flights %>%
  filter(!is.na(arr_delay)) %>%
  group_by(carrier) %>%
  summarize(avg_delay = mean(arr_delay), count = n()) %>%
  arrange(avg_delay) %>%
  head(5))
```

```
## # A tibble: 5 x 3
##   carrier avg_delay count
##   <chr>       <dbl> <int>
## 1 AS          -9.93    709
## 2 HA          -6.92    342
## 3 AA           0.364 31947
## 4 DL           1.64  47658
## 5 VX           1.76   5116
```

Exercise 2

1, **diamonds** dataset, columns: `carat`, `color`

```
(missing_diamonds <- diamonds %>%
  filter(is.na(carat), is.na(color)))
```

```
## # A tibble: 0 x 10
## # ... with 10 variables: carat <dbl>, cut <ord>, color <ord>,
## #   clarity <ord>, depth <dbl>, table <dbl>, price <int>, x <dbl>,
## #   y <dbl>, z <dbl>
```

2, **flights** dataset, columns: `arr_delay`, `dep_delay`

```
(missing_flights <- flights %>%
  filter(is.na(arr_delay), is.na(dep_delay)))
```

```
## # A tibble: 8,255 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
```

```
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 8,245 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

3, **mtcars** dataset, columns: `cyl`, `mpg`

```
(missing_mtcars <- mtcars %>%
  filter(is.na(cyl), is.na(mpg)))
```

```
##  [1] mpg  cyl  disp hp   drat wt   qsec vs   am   gear carb
## <0 rows> (or 0-length row.names)
```

Searching for `NA` values in all columns would require to type the name of all the columns of the DF. It requires a lot of time and exposes us to the pain of typos.

Exercise 3

```
(diamonds_any_missing <- filter_all(diamonds, any_vars(is.na(.))))
```

```
## # A tibble: 0 x 10
## # ... with 10 variables: carat <dbl>, cut <ord>, color <ord>,
## #   clarity <ord>, depth <dbl>, table <dbl>, price <int>, x <dbl>,
## #   y <dbl>, z <dbl>
```

```
(flights_any_missing <- filter_all(flights, any_vars(is.na(.))))
```

```
## # A tibble: 9,430 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1     1525           1530        -5     1934
##  2  2013     1     1     1528           1459        29     2002
##  3  2013     1     1     1740           1745        -5     2158
##  4  2013     1     1     1807           1738        29     2251
##  5  2013     1     1     1939           1840        59       29
##  6  2013     1     1     1952           1930        22     2358
##  7  2013     1     1     2016           1930        46       NA
##  8  2013     1     1       NA           1630        NA       NA
##  9  2013     1     1       NA           1935        NA       NA
## 10  2013     1     1       NA           1500        NA       NA
## # ... with 9,420 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

```
(mtcars_any_missing <- filter_all(mtcars, any_vars(is.na(.))))
```

```
##  [1] mpg  cyl  disp hp   drat wt   qsec vs   am   gear carb
## <0 rows> (or 0-length row.names)
```

Exercise 4

I picked this one to skip. I know myself; if I had done this exercise, I would have spent 80% of the time choosing my favourite dataset instead of visualisation and it would not have been efficient now.

Exercise 5

Calculate the average values of `x` and `y` for each group in the below dataset.

```
(df <- read_csv("1,2,3
         2,-,6
         2,9,-", col_names = c("group_number", "x", "y"),na = "-"))
```

```
## # A tibble: 3 x 3
##   group_number     x     y
##          <dbl> <dbl> <dbl>
## 1            1     2     3
## 2            2    NA     6
## 3            2     9    NA
```

Wrong solution:

```
(avg_df <- df %>%
  group_by(group_number) %>%
  summarise(avg_x = mean(x), avg_y = mean(y)))
```

```
## # A tibble: 2 x 3
##   group_number avg_x avg_y
##          <dbl> <dbl> <dbl>
## 1            1     2     3
## 2            2    NA    NA
```

Good solution:

```
(avg_df <- df %>%
  filter(!is.na(x), !is.na(y)) %>%
  group_by(group_number) %>%
  summarise(avg_x = mean(x), avg_y = mean(y)))
```

```
## # A tibble: 1 x 3
##   group_number avg_x avg_y
##          <dbl> <dbl> <dbl>
## 1            1     2     3
```

Take-away: if a group of values contains at least one `NA` value, the average will be `NA` as well. R will not disregard them automatically, you have to filter out tha `NA` values before calculating means (and probably other summary statistics as well).

Exercise 6

It is about reading our own data to R. Quite helpful.

Reading own CSV file:

```
hotels <- read.csv(file = "..//da_data_repo/hotelbookingdata.csv")
```

"Reading" CSV inputted from command.

```
read_csv("a,b,c
1,2,3
4,5,6")
```

```
## # A tibble: 2 x 3
##       a     b     c
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

Skipping rows:

```
read_csv("The first line of metadata
  The second line of metadata
  x,y,z
  1,2,3", skip = 2)
```

```
## # A tibble: 1 x 3
##       x     y     z
##   <dbl> <dbl> <dbl>
## 1     1     2     3
```

Reading without header:

```
read_csv("1,2,3
         4,5,6", col_names = FALSE)
```

```
## # A tibble: 2 x 3
##      X1    X2    X3
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

Inputting column names separately:

```
read_csv("1,2,3
         4,5,6", col_names = c("x", "y", "z"))
```

```
## # A tibble: 2 x 3
##       x     y     z
##   <dbl> <dbl> <dbl>
## 1     1     2     3
## 2     4     5     6
```

We can tell R what input chars to transform to NA values:

```
read_csv("a,b,c
         1,.,.", na = ".")
```

```
## # A tibble: 1 x 3
##       a b     c
##   <dbl> <lgl> <lgl>
## 1     1 NA    NA
```