STAT 412/612 Week 12: Homework

forcats and lubridate

Yunting

4/12/2020

```
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(lubridate))
```

Question 1: Capital Bikeshare Data

1. Load in the data containing trip information from the Capital Bikeshare program. Also load in the station information. Rename variables that have spaces in the names trip data, station data
Note: These data were originally from http://data.codefordc.org/group/transportation.

```
stations <- read_csv(file = "https://dcgerard.github.io/stat_412_612/data/capital_stations.csv")
```

```
## Parsed with column specification:
## cols(
##   id = col_double(),
##   name = col_character(),
##   terminalName = col_double(),
##   lastCommWithServer = col_double(),
##   lat = col_double(),
##   long = col_double(),
##   installed = col_logical(),
##   installDate = col_double(),
##   removalDate = col_double(),
##   temporary = col_logical(),
##   public = col_logical(),
##   capacity = col_double()
## )
```

```
head(stations)
```

```
## # A tibble: 6 x 12
##      id name  terminalName lastCommWithSer...   lat  long installed installDate
##   <dbl> <chr>        <dbl>            <dbl> <dbl> <dbl> <lgl>           <dbl>
## 1     1 20th...        31000    1368370000000  38.9 -77.1 TRUE          1.32e12
## 2     2 18th...        31001    1368370000000  38.9 -77.1 TRUE          1.28e12
## 3     3 20th...        31002    1368370000000  38.9 -77.0 TRUE          1.28e12
## 4     4 15th...        31003    1368370000000  38.9 -77.1 TRUE          1.28e12
```

```
## 5     5 18th...        31004    1368370000000  38.9 -77.1 TRUE          1.28e12
## 6     6 15th...        31005    1368370000000  38.9 -77.1 TRUE          1.28e12
## # ... with 4 more variables: removalDate <dbl>, temporary <lgl>, public <lgl>,
## #   capacity <dbl>
```

```
trips2016 <- read_csv(file = "https://dcgerard.github.io/stat_412_612/data/capital_trips_2016.csv")
```

```
## Parsed with column specification:
## cols(
##   `Duration (ms)` = col_double(),
##   `Start date` = col_character(),
##   `End date` = col_character(),
##   `Start station number` = col_double(),
##   `Start station` = col_character(),
##   `End station number` = col_double(),
##   `End station` = col_character(),
##   `Bike number` = col_character(),
##   `Member Type` = col_character()
## )
```

```
trips2016 %>%
  rename(Duration = "Duration (ms)", Start_date = "Start date",
         End_date = "End date", Start_station_number = "Start station number",
         Start_station = "Start station", End_station_number = "End station number",
         End_station = "End station", Bike_number = "Bike number",
         Member_Type = "Member Type") ->
  trips2016

head(trips2016)
```

```
## # A tibble: 6 x 9
##   Duration Start_date End_date Start_station_n... Start_station End_station_num...
##      <dbl> <chr>      <chr>               <dbl> <chr>                       <dbl>
## 1   301295 3/31/2016... 4/1/201...           31280 11th & S St ...             31506
## 2   557887 3/31/2016... 4/1/201...           31275 New Hampshir...             31114
## 3   555944 3/31/2016... 4/1/201...           31101 14th & V St ...             31221
## 4   766916 3/31/2016... 4/1/201...           31226 34th St & Wi...             31214
## 5   139656 3/31/2016... 3/31/20...           31011 23rd & Cryst...             31009
## 6   967713 3/31/2016... 4/1/201...           31266 11th & M St ...             31600
## # ... with 3 more variables: End_station <chr>, Bike_number <chr>,
## #   Member_Type <chr>
```

2. Parse the date-time information from the trip data. Recall the times are recorded in the **America/New_York** time zone, not the **UTC** time zone. Specify that in your parser.

```
trips2016 %>%
  mutate(Start_date = mdy_hm(Start_date, tz = "America/New_York"),
         End_date = mdy_hm(End_date, tz = "America/New_York")) ->
  trips2016

head(trips2016)
```

```
## # A tibble: 6 x 9
##   Duration Start_date          End_date            Start_station_n...
##      <dbl> <dttm>              <dttm>                          <dbl>
## 1   301295 2016-03-31 23:59:00 2016-04-01 00:04:00            31280
## 2   557887 2016-03-31 23:59:00 2016-04-01 00:08:00            31275
## 3   555944 2016-03-31 23:59:00 2016-04-01 00:08:00            31101
## 4   766916 2016-03-31 23:57:00 2016-04-01 00:09:00            31226
## 5   139656 2016-03-31 23:57:00 2016-03-31 23:59:00            31011
## 6   967713 2016-03-31 23:57:00 2016-04-01 00:13:00            31266
## # ... with 5 more variables: Start_station <chr>, End_station_number <dbl>,
## #   End_station <chr>, Bike_number <chr>, Member_Type <chr>
```

3. Calculate the average number of trips for each weekday (Sunday, Monday, Tuesday . . .) given the day has trips. There are several days with no trips.
   - Save the resulting days of week and corresponding average number of trips as a data frame called **sumdf** and print it out.

```
library(lubridate)
trips2016 %>%
  separate(Start_date, into = c("Start_d", "Start_t"), sep = " ") %>%
  group_by(Start_d)%>%
  summarize(trips = n()) %>%
  mutate(wday = wday(Start_d, label = TRUE)) %>%
  group_by(wday) %>%
  summarize(mean_num_trips = mean(trips)) -> sumdf

print(sumdf)
```

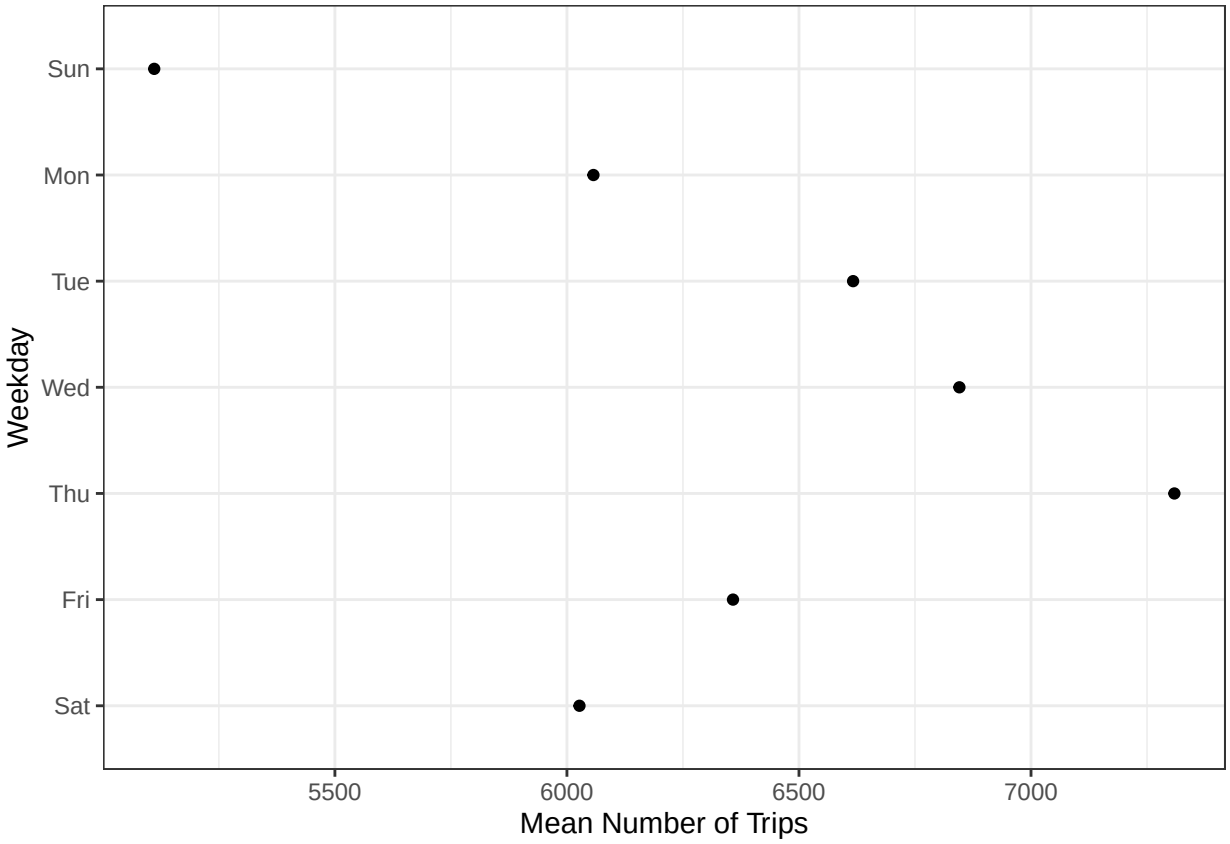
```
## # A tibble: 7 x 2
##   wday  mean_num_trips
##   <ord>          <dbl>
## 1 Sun            5111.
## 2 Mon            6057.
## 3 Tue            6617.
## 4 Wed            6846.
## 5 Thu            7309.
## 6 Fri            6358.
## 7 Sat            6027
```

```
# Another Method: using yday()
# mutate(Start_date = yday(Start_date)) %>%
# count(Start_date)
```

4. Reproduce this plot in R:

```
sumdf %>%
  mutate(wday = ordered(wday, levels= c("Sat", "Fri", "Thu",
                                        "Wed", "Tue", "Mon", "Sun"))) ->
  sumdf1
```

```r
ggplot(sumdf1, aes(x = mean_num_trips, y = wday)) +
geom_point() +
theme_bw()+
xlab("Mean Number of Trips") +
ylab("Weekday")
```



5. In a stunning show of contempt, the IEEE Computer Society decided to add a new weekday called "Fooday" with abbreviation "Foo". Fooday was decided to be the first day of the week (ahead of Sunday).
   On the first Fooday ever, people used Capital Bikeshare in record numbers, yielding 15567 trips. Add Fooday as the first level to the **wday** variable in **sumdf** and add its average number of trips (now 15567 since there has only been one Fooday so far).
   Hint: Create a new data frame that contains the Fooday trips and use **bind_rows()**.

```r
Fooday <- tribble(~wday, ~mean_num_trips,
             ##----/------------
              "Foo", 15567)

# Fooday <- data.frame(wday = "Foo", mean_num_trips = 15567)

wday_vec <- c("Foo","Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat" )

Fooday %>%
  mutate(wday = factor(wday, levels = wday_vec)) -> Fooday2
```

```
sumdf %>%
  mutate(wday = factor(wday, levels = wday_vec, ordered = FALSE )) -> sumdf2

bind_rows(Fooday2, sumdf2) -> Fooday_trips
print(Fooday_trips)
```

```
## # A tibble: 8 x 2
##   wday  mean_num_trips
##   <fct>          <dbl>
## 1 Foo           15567
## 2 Sun            5111.
## 3 Mon            6057.
## 4 Tue            6617.
## 5 Wed            6846.
## 6 Thu            7309.
## 7 Fri            6358.
## 8 Sat            6027
```

6. In another stunning show of contempt, the IEEE Computer Society decided to change the abbreviations from three letters to two letters. Change the levels of wday so that each day uses only two-letter abbreviations. Your final data frame should look like this:

```
Fooday_trips %>%
  mutate(wday = str_sub(wday,1,2),
         wday = parse_factor(wday))->
  ABBR_Fooday_trips

print(ABBR_Fooday_trips)
```

```
## # A tibble: 8 x 2
##   wday  mean_num_trips
##   <fct>          <dbl>
## 1 Fo            15567
## 2 Su             5111.
## 3 Mo             6057.
## 4 Tu             6617.
## 5 We             6846.
## 6 Th             7309.
## 7 Fr             6358.
## 8 Sa             6027
```

7. In the **stations** data frame, it seems that **installDate** is populated by the number of milliseconds since January 1, 1970, 00:00:00 (in the **America/New_York** time zone). Parse this into a date-time and make a histogram of the install dates. It should look something like this:

```
# 1000 milliseconds = 1 seconds
stations <- read_csv(file = "./data/capital_stations.csv")
```

```
## Parsed with column specification:
```

```
## cols(
##   id = col_double(),
##   name = col_character(),
##   terminalName = col_double(),
##   lastCommWithServer = col_double(),
##   lat = col_double(),
##   long = col_double(),
##   installed = col_logical(),
##   installDate = col_double(),
##   removalDate = col_double(),
##   temporary = col_logical(),
##   public = col_logical(),
##   capacity = col_double()
## )
```

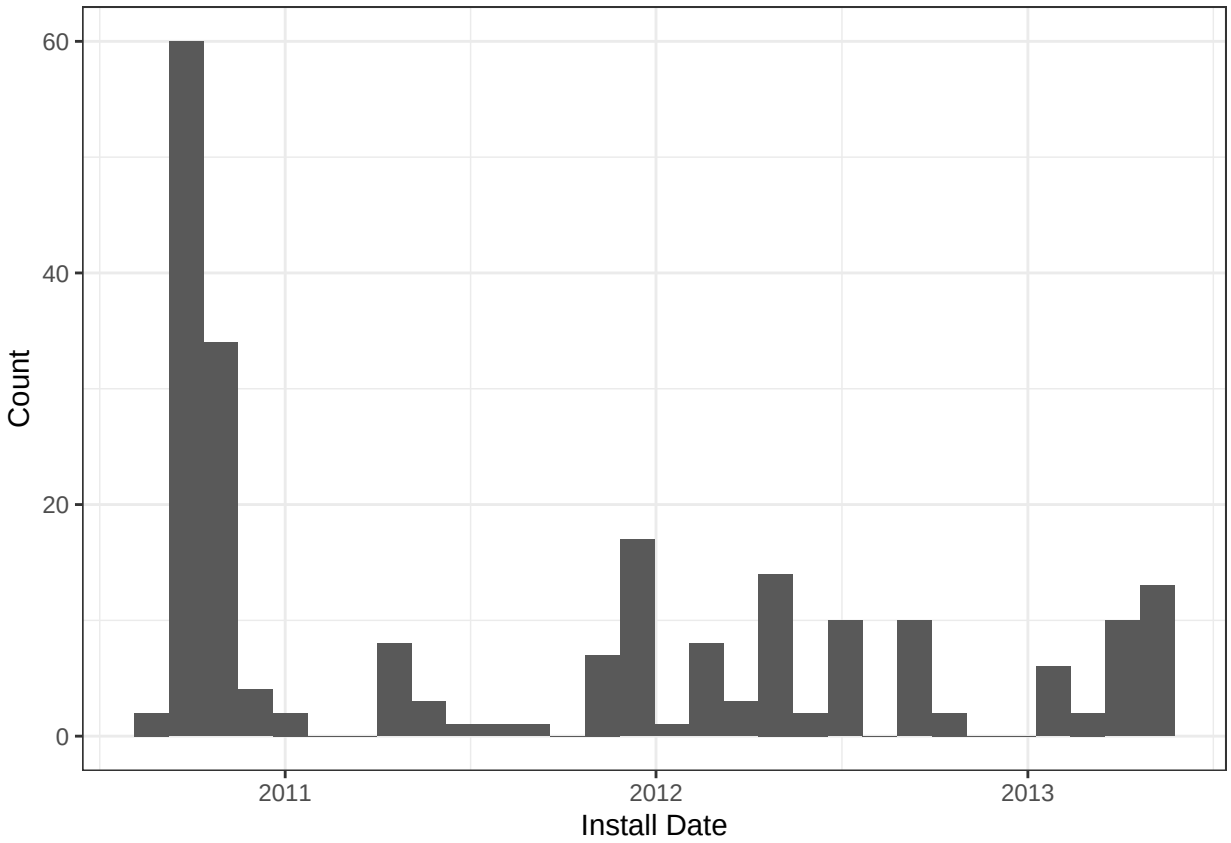
```
sample_n(stations,6)
```

```
## # A tibble: 6 x 12
##      id name  terminalName lastCommWithSer...   lat  long installed installDate
##   <dbl> <chr>        <dbl>            <dbl> <dbl> <dbl> <lgl>           <dbl>
## 1   125 11th...        31262    1368370000000  38.9 -77.0 TRUE          1.36e12
## 2   146 9th ...        31404    1368370000000  38.9 -77.0 TRUE          1.32e12
## 3    86 19th...        31224    1368370000000  38.9 -77.0 TRUE          1.29e12
## 4   145 7th ...        31245    1368370000000  38.9 -77.0 TRUE          1.32e12
## 5   141 18th...        31242    1368370000000  38.9 -77.0 TRUE          1.32e12
## 6    61 Flor...        31503    1368370000000  38.9 -77.0 TRUE          1.29e12
## # ... with 4 more variables: removalDate <dbl>, temporary <lgl>, public <lgl>,
## #   capacity <dbl>
```

```
stations %>%
  mutate(oldtime = ymd_hms("1970-01-01 00:00:00", tz = "America/New_York"),
         installDate = (oldtime + dmilliseconds(installDate))) %>%

  ggplot(aes(x = installDate)) +
  geom_histogram() +
  theme_bw() +
  xlab("Install Date") +
  ylab("Count")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


60
40
20
0
Count
2011
2012
2013
Install Date