STA035B Homework 2, due: 2/1, 9pm

Spencer Frei

Instructions

Upload a PDF file, named with your UC Davis email ID and homework number (e.g., sfrei_hw2.pdf), to Gradescope (accessible through Canvas). You will give the commands to answer each question in its own code block, which will also produce output that will be automatically embedded in the output file. All code used to answer the question must be supplied, as well as written statements where appropriate.

All code used to produce your results must be shown in your PDF file (e.g., do not use `echo = FALSE` or `include = FALSE` as options anywhere). Rmd files do not need to be submitted, but may be requested by the TA and must be available when the assignment is submitted.

Students may choose to collaborate with each other on the homework, but must clearly indicate with whom they collaborated.

Problem 1

```
library(nycflights13)
```

Consider the `weather` dataset (comes when you load `nycflights13` library), which has columns: "origin", "year", "month", "day", "hour", "temp", "dewp", "humid", "wind_dir", "wind_speed", "wind_gust", "precip", "pressure", "visib", and "time_hour". We show the first few rows and columns below.

```
weather
```

```
# A tibble: 26,115 x 15
   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed
   <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>
 1 EWR     2013     1     1     1  39.0  26.1  59.4      270       10.4
 2 EWR     2013     1     1     2  39.0  27.0  61.6      250        8.06
 3 EWR     2013     1     1     3  39.0  28.0  64.4      240       11.5
 4 EWR     2013     1     1     4  39.9  28.0  62.2      250       12.7
 5 EWR     2013     1     1     5  39.0  28.0  64.4      260       12.7
...
```

(a) Provide code which computes the average total precipitation per origin per month, removing any missing values. That is, you should have for every origin and month pair, the average (across all years in the tibble) amount of precipitation received at that origin in that month. (i.e., we should be able to answer: In a given July at Newark airport, what is the typical total precipitation?) Then filter the resulting tibble so that only those origin-month pairs with the 5 highest average precipitation remain.

Hint: You should first compute what the total precipitation per origin, year, and month is. You now have a tibble which records the total precipitation per origin-month pair for every year; now average across all of the years for which you have observations in that month for the origin.

Answer: if there were multiple years in the tibble, the following code would be the correct way to do it:

```
weather %>%
  group_by(origin, month, year) %>%
  mutate(total_precip = sum(precip, na.rm=TRUE)) %>%
  group_by(origin, month) %>%
  summarize(avg_precip = mean(total_precip, na.rm=TRUE)) %>%
  arrange(by = desc(avg_precip)) %>%
  head(5)
```

```
`summarise()` has grouped output by 'origin'. You can override using the
`.groups` argument.
```

```
# A tibble: 5 x 3
# Groups:   origin [3]
  origin month avg_precip
  <chr>  <int>      <dbl>
1 EWR        6       8.73
2 LGA        6       8.16
3 JFK        6       7.95
4 EWR        5       5.44
5 LGA        5       4.99
```

Since there is only one year in the tibble, this is equivalent to the following code:

```
weather %>%
  group_by(origin, month) %>%
  mutate(total_precip = sum(precip, na.rm=TRUE)) %>%
  summarize(avg_precip = mean(total_precip, na.rm=TRUE)) %>%
```

```
  arrange(by = desc(avg_precip)) %>%
  head(5)
```

```
`summarise()` has grouped output by 'origin'. You can override using the
`.groups` argument.
```

```
# A tibble: 5 x 3
# Groups:   origin [3]
  origin month avg_precip
  <chr>  <int>      <dbl>
1 EWR        6       8.73
2 LGA        6       8.16
3 JFK        6       7.95
4 EWR        5       5.44
5 LGA        5       4.99
```

(b) Do a similar calculation: compute the average total precipitation per origin per **week number** (i.e., Jan 1 - Jan 7 is week 1, Jan 8 - 15 is week 2, etc.) removing any missing values. Then filter the resulting tibble so that only those origin-month pairs with the 5 highest average precipitation remain. Note that `weather` does not have a week number so you need to create this yourself—think of what mathematical operations allow for you to find the week number, and look back at the slides on dates/times.

Hint: similar to the previous part of the problem, you need to calculate the total precipitation per each week for every origin and year. Then average over different years.

Answer: same as before, if there were > 1 year in the tibble, this would be the correct code:

```
weather %>%
  mutate(weeknum = yday(time_hour) %/% 7 + 1) %>%
  group_by(origin, weeknum, year) %>%
  mutate(total_precip = sum(precip, na.rm=TRUE)) %>%
  group_by(origin, weeknum) %>%
  summarize(avg_precip = mean(total_precip, na.rm=TRUE)) %>%
  arrange(by = desc(avg_precip)) %>%
  head(5)
```

```
`summarise()` has grouped output by 'origin'. You can override using the
`.groups` argument.
```

```
# A tibble: 5 x 3
# Groups:   origin [3]
  origin weeknum avg_precip
  <chr>    <dbl>      <dbl>
1 LGA         23       5.25
2 JFK         23       4.79
3 EWR         23       4.5
4 EWR         19       3.55
5 LGA         19       3.31
```

Since there is only one year, the following code is equivalent:

```
weather %>%
  mutate(weeknum = yday(time_hour) %/% 7 + 1) %>%
  group_by(origin, weeknum) %>%
  mutate(total_precip = sum(precip, na.rm=TRUE)) %>%
  summarize(avg_precip = mean(total_precip, na.rm=TRUE)) %>%
  arrange(by = desc(avg_precip)) %>%
  head(5)
```

```
`summarise()` has grouped output by 'origin'. You can override using the
`.groups` argument.
```

```
# A tibble: 5 x 3
# Groups:   origin [3]
  origin weeknum avg_precip
  <chr>    <dbl>      <dbl>
1 LGA         23       5.25
2 JFK         23       4.79
3 EWR         23       4.5
4 EWR         19       3.55
5 LGA         19       3.31
```

Problem 2

- Suppose we have a tibble of the following form:

```
df <- tribble(
  ~name, ~date_of_birth, ~favorite_food,
  "Will", "1995-09-01", "Tacos",
  "Angela", "1993-01-02", "Sushi",
  "Ana", "1994-11-20", "Italian"
)
```

(a) Provide R code which adds the following columns:

- `year`, a number, indicating the year of birth
- `month`, a string, the month (fully spelled out, i.e. January) of birth
- `day`, a number, indicating the day of the month the person was born

The resulting tibble should have 6 columns, the 3 original ones plus these 3 new ones.

```
df %>%
  mutate(parsed_date = as_datetime(date_of_birth),
         year = year(parsed_date),
         month = month(parsed_date),
         day = day(parsed_date)) %>%
  select(-parsed_date)
```

```
# A tibble: 3 x 6
  name   date_of_birth favorite_food  year month   day
  <chr>  <chr>         <chr>         <dbl> <dbl> <int>
1 Will   1995-09-01    Tacos          1995     9     1
2 Angela 1993-01-02    Sushi          1993     1     2
3 Ana    1994-11-20    Italian        1994    11    20
```

(b) Provide R code which adds the following column:

- `ten_years_later`: a date time, indicating a the date corresponding to ten years after the person's birth

The resulting tibble should have 4 columns, the original 3 plus this new one.

```
df %>%
  mutate(parsed_date = as_datetime(date_of_birth),
        ten_years_later = parsed_date + years(10))
```

```
# A tibble: 3 x 5
  name   date_of_birth favorite_food parsed_date         ten_years_later
  <chr>  <chr>         <chr>         <dttm>              <dttm>
1 Will   1995-09-01    Tacos         1995-09-01 00:00:00 2005-09-01 00:00:00
2 Angela 1993-01-02    Sushi         1993-01-02 00:00:00 2003-01-02 00:00:00
3 Ana    1994-11-20    Italian       1994-11-20 00:00:00 2004-11-20 00:00:00
```

(c) Provide R code which adds the following column:

- `age_when_obama_born`: an **integer**, indicating how many years old the person was when Barack Obama was born (August 4, 1961)
  - Be sure it is an integer and there are no decimal places!
  - The resulting tibble should have 4 columns, the original 3 plus this new one. No extra columns!

```
df %>%
  mutate(parsed_date = as_datetime(date_of_birth),
        age_when_obama_born_decimals = (as_date("1961-08-04") %--% parsed_date) / years(1),
        age_when_obama_born = floor(age_when_obama_born_decimals)
```

```
      ) %>%
  select(-parsed_date, -age_when_obama_born_decimals)
```

```
# A tibble: 3 x 4
  name   date_of_birth favorite_food age_when_obama_born
  <chr>  <chr>         <chr>                        <dbl>
1 Will   1995-09-01    Tacos                           34
2 Angela 1993-01-02    Sushi                           31
3 Ana    1994-11-20    Italian                         33
```

Problem 3

- Suppose we have the following tibble:

```
entered_data <- tribble(
  ~id, ~entry,
  0, "Arthur_1985-09-01_Present",
  1, "Zack_1983-01-02_Absent",
  2, "Pat_1984-11-20_Present",
)
```

Provide R code which parses the `entry` column and creates three new columns:

- `name`, a string, indicating the person's name (preceding the first underscore),
- `date_of_birth`, a date time indicating the person's date of birth,
- `day_can_vote`, a date time indicating the day when the person is 18 years old.

The resulting tibble should have 5 columns: `id` and `entry` in addition to these 3 new columns.

```
entered_data %>%
  mutate(name = str_replace(entry, "(.*)_(.*)_(.*)", "\\1"),
         date_of_birth = as_datetime(str_replace(entry, "(.*)_(.*)_(.*)", "\\2"))
  ) %>%
  mutate(day_can_vote = date_of_birth + years(18))
```

```
# A tibble: 3 x 5
     id entry                      name   date_of_birth       day_can_vote
  <dbl> <chr>                      <chr>  <dttm>              <dttm>
1     0 Arthur_1985-09-01_Present Arthur 1985-09-01 00:00:00 2003-09-01 00:00:00
2     1 Zack_1983-01-02_Absent    Zack   1983-01-02 00:00:00 2001-01-02 00:00:00
3     2 Pat_1984-11-20_Present    Pat    1984-11-20 00:00:00 2002-11-20 00:00:00
```

```
entered_data %>%
  separate_wider_regex(
    entry,
    patterns = c(
      name = ".*",
      "_",
      date_of_birth = ".*",
      "_.*"
    )
  ) %>%
  mutate(date_of_birth = as_datetime(date_of_birth),
         day_can_vote = date_of_birth + years(18))
```

```
# A tibble: 3 x 4
     id name   date_of_birth       day_can_vote
  <dbl> <chr>  <dttm>              <dttm>
1     0 Arthur 1985-09-01 00:00:00 2003-09-01 00:00:00
2     1 Zack   1983-01-02 00:00:00 2001-01-02 00:00:00
3     2 Pat    1984-11-20 00:00:00 2002-11-20 00:00:00
```