Module 1 Assignment 3: Getting to Know your Home

Ellen Bledsoe

2023-09-11

Assignment Description

Purpose

The goal of this assignment is to get comfortable using the `tidyverse` with 2-dimensional data sets and compare this process to using base R.

Task

Write R code using the `tidyverse` to successfully answer each question below.

Criteria for Success

- Code is within the provided code chunks
- Code chunks run without errors
- Code produces the correct result
  - Code that produces the correct answer will receive full credit
  - Code attempts with logical direction will receive partial credit
- Written answers address the questions in sufficient detail

Due Date

Sept. 19 at 4pm MST

Assignment Questions

For this final assignment for Module 1, you'll be working with another real-world data set–a collection of data from climate stations scattered across Antarctica.

1. In your own words, describe what the `tidyverse` is. Your answer should be between 1-3 sentences.
2. Load in the `tidyverse` package.

```
# load packages
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

3. Load in the data file (called aggregated_station_data.csv) using the `read_csv()` function. Save the data as an object called `weather`.

```
# read in weather station data
weather <- read_csv("../data/aggregated_station_data.csv")
```

```
## Rows: 139160 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): station_id
## dbl (11): year, day, month, running_day, hour, temp, pressure, wind_speed, w...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

4. Take a look at the data in whichever way you would like (e.g., `glimpse()`, `slice()`, `str()`, `head()`, etc.). How many rows and columns are in the data? Type your answers below:

   rows: 139, 160

   columns: 12

5. Create a new data frame that only includes temperatures which are above freezing (AKA greater than 0)

```
filter(weather, temp > 0)
```

```
## # A tibble: 769 x 12
##     year   day month running_day  hour  temp pressure wind_speed wind_direction
##    <dbl> <dbl> <dbl>       <dbl> <dbl> <dbl>    <dbl>      <dbl>          <dbl>
##  1  2018     5     1           5   300   0.2     985.        2.6            8
##  2  2018     7     1           7  1800   0.2     988.        6.5           49.7
##  3  2018     7     1           7  2100   1       988.        8             45
##  4  2018     8     1           8     0   1.4     989.       10.2           44.4
##  5  2018     8     1           8   300   0.5     991.        6            212.
##  6  2018     8     1           8   600   0.3     992.        5.3          226.
##  7  2018    20     1          20     0   1.3     969.       10.7          204.
##  8  2018    20     1          20   300   2.6     968.       14.6          203.
##  9  2018    20     1          20   600   1.9     968        11.5          216.
## 10  2018    20     1          20   900   1.6     967.       15.6          200.
## # i 759 more rows
## # i 3 more variables: humidity <dbl>, delta_t <dbl>, station_id <chr>
```

6. Create a new data frame called `station_temp` that includes *only* the following columns: year, day, month, temp, station_id.

```
station_temp <- select(weather, year:month, temp, station_id)
station_temp
```

```
## # A tibble: 139,160 x 5
##     year   day month  temp station_id
##    <dbl> <dbl> <dbl> <dbl> <chr>
##  1  2018     1     1 -29.5 ag4201801q3h
##  2  2018     1     1 -27.4 ag4201801q3h
##  3  2018     1     1 -25.5 ag4201801q3h
##  4  2018     1     1 -24.9 ag4201801q3h
##  5  2018     1     1 -25   ag4201801q3h
##  6  2018     1     1 -27.5 ag4201801q3h
##  7  2018     1     1 -30.3 ag4201801q3h
##  8  2018     1     1 -30.1 ag4201801q3h
##  9  2018     2     1 -28.8 ag4201801q3h
## 10  2018     2     1 -26.4 ag4201801q3h
## # i 139,150 more rows
```

7. Using the data frame you created in Q5 above (`station_temp`), add a new column to that data frame that converts the temperature column (currently in Celsius) to Fahrenheit. Call the new column `tempF`. (Hint: we did this in class–use that same equation)

```
station_temp %>%
  mutate(tempF = temp*(9/5) + 32)
```

```
## # A tibble: 139,160 x 6
##     year   day month  temp station_id   tempF
##    <dbl> <dbl> <dbl> <dbl> <chr>        <dbl>
##  1  2018     1     1 -29.5 ag4201801q3h -21.1
##  2  2018     1     1 -27.4 ag4201801q3h -17.3
##  3  2018     1     1 -25.5 ag4201801q3h -13.9
##  4  2018     1     1 -24.9 ag4201801q3h -12.8
##  5  2018     1     1 -25   ag4201801q3h -13
##  6  2018     1     1 -27.5 ag4201801q3h -17.5
##  7  2018     1     1 -30.3 ag4201801q3h -22.5
##  8  2018     1     1 -30.1 ag4201801q3h -22.2
##  9  2018     2     1 -28.8 ag4201801q3h -19.8
## 10  2018     2     1 -26.4 ag4201801q3h -15.5
## # i 139,150 more rows
```

8. In *your own words* (either bullet points or sentence form is fine), explain two benefits of using the pipe (`%>%`).

9. Using the *original* data frame (`weather`), find the minimum temperature recorded for each month (in Celsius, the original column called `temp`). (Hint: think about months first (split) and then temperature (apply). You will also want to remove all the NA values.)

```
weather %>%
  group_by(month) %>%
  summarise(min_temp = min(temp, na.rm = TRUE))
```

```
## # A tibble: 12 x 2
##    month min_temp
##    <dbl>    <dbl>
##  1     1    -44.2
##  2     2    -59
##  3     3    -67.9
##  4     4    -72.3
##  5     5    -77.1
##  6     6    -76
##  7     7    -79.5
##  8     8    -80.2
##  9     9    -77.1
## 10    10    -70.8
## 11    11    -59.4
## 12    12    -41.3
```

10. Again, using the *original* data frame, create a data frame with the mean temperature for the month of January for each station.

Some hints:

- take note of how months are represented in the data
- think about using the pipe, how we choose which rows we want, and how we split-apply-combine
- remember to remove the NA values!

```
weather %>%
  filter(month == 1) %>%
  group_by(station_id) %>%
  summarize(mean_temp = mean(temp, na.rm = TRUE))
```

```
## # A tibble: 49 x 2
##    station_id   mean_temp
##    <chr>            <dbl>
##  1 ag4201801q3h    -31.4
##  2 bal201801q3h    -19.1
##  3 brp201801q3h     -6.05
##  4 byd201801q3h    -15.5
##  5 cbd201801q3h     -3.83
##  6 cha201801q3h     -3.04
##  7 d10201801q3h     -3.32
##  8 d47201801q3h    -13.4
##  9 d85201801q3h    -24.2
## 10 dc2201801q3h    -27.4
## # i 39 more rows
```

Bonus! (up to 2 points)

Write code to determine how many unique stations are in the `weather` data set. (Hint: look up the help file for the `distinct()` and the `count()` functions).

```
# number of unique stations
weather %>%
  distinct(station_id) %>%
  count()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1   571
```