Assignment7-445

Mandi Bluth

2024-11-04

Exercise 1

A common task is to take a set of data that has multiple categorical variables and create a table of the number of cases for each combination. An introductory statistics textbook contains a data set summarizing student surveys from several sections of an intro class. The two variables of interest are `Gender` and `Year` which are the students gender and year in college. *Note: you will need to refer to Chapter 4 and Chapter 7 for some of the operations needed below - this is a great time to review chapter 4!*

a) Download the data set using the following:

```
Survey <- read.csv('https://www.lock5stat.com/datasets2e/StudentSurvey.csv', na.strings=c('',' '))
```

b) Select the specific columns of interest **Year** and **Gender**

```
Survey <- Survey %>% select(Year, Gender)
head(Survey)
```

```
##        Year Gender
## 1    Senior      M
## 2 Sophomore      F
## 3 FirstYear      M
## 4    Junior      M
## 5 Sophomore      F
## 6 Sophomore      F
```

c) Convert the **Year** column to factors and properly order the factors based on common US progression (FirstYear - Sophomore - Junior - Senior)

```
Survey$Year <- factor(Survey$Year, levels = c("FirstYear", "Sophomore", "Junior", "Senior"))
```

d) Convert the **Gender** column to factors and rename them Male/Female.

```
Survey <- Survey %>% mutate(Gender = ifelse(Gender == "M", "Male", "Female"))
head(Survey)
```

```
##        Year Gender
## 1    Senior   Male
## 2 Sophomore Female
## 3 FirstYear   Male
## 4    Junior   Male
## 5 Sophomore Female
## 6 Sophomore Female
```

e) Produce a data set with eight rows and three columns that contains the number of responses for each gender:year combination. *You might want to look at the following functions:* ***`dplyr::count`*** *and* ***`dplyr::drop_na`***.

```
Survey <- Survey %>% drop_na() %>%
  group_by(Year, Gender) %>%
  summarize(Count = n())
```

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

```
Survey
```

```
## # A tibble: 8 x 3
## # Groups:   Year [4]
##   Year      Gender Count
##   <fct>     <chr>  <int>
## 1 FirstYear Female    43
## 2 FirstYear Male      51
## 3 Sophomore Female    96
## 4 Sophomore Male      99
## 5 Junior    Female    18
## 6 Junior    Male      17
## 7 Senior    Female    10
## 8 Senior    Male      26
```

f) Pivot the table in part (e) to produce a table of the number of responses in the following form:

Gender	First Year	Sophomore	Junior	Senior
Female				
Male				

```
Survey <- Survey %>% pivot_wider(names_from =Year, values_from=Count)
Survey
```

```
## # A tibble: 2 x 5
##   Gender FirstYear Sophomore Junior Senior
##   <chr>      <int>     <int>  <int>  <int>
## 1 Female        43        96     18     10
## 2 Male          51        99     17     26
```

Exercise 2

From this book's GitHub there is a .csv file of the daily maximum temperature in Flagstaff at the Pulliam Airport. The link is: https://raw.githubusercontent.com/BuscagliaR/STA_444_v2/master/data-raw/FlagMaxTemp.csv

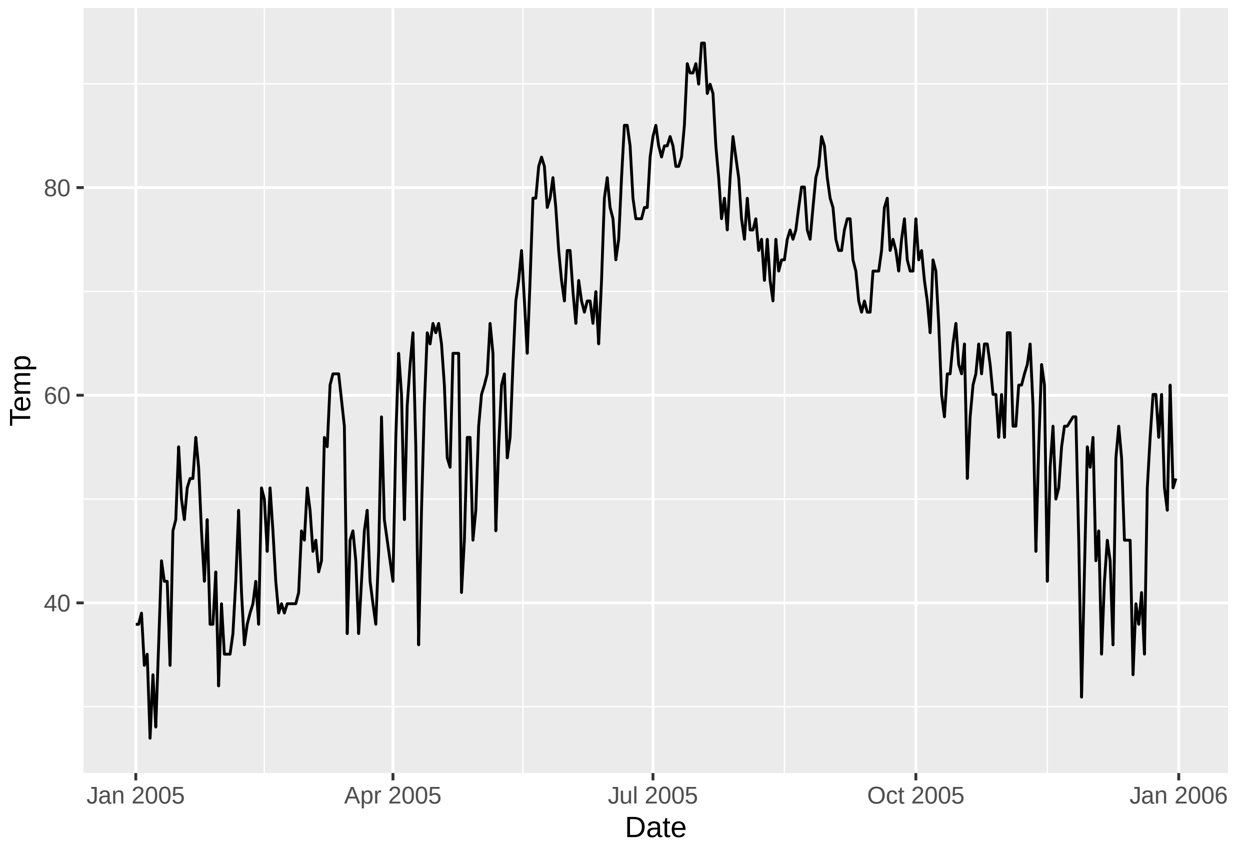
a) Create a line graph that gives the daily maximum temperature for 2005. *Make sure the x-axis is a date and covers the whole year.*

```r
Flag.temp <- read.csv("https://raw.githubusercontent.com/BuscagliaR/STA_444_v2/master/data-raw/FlagMaxT
Flag.temp <- Flag.temp %>% pivot_longer(X1:X31,
            names_to = "Day",
            values_to = "Temp") %>%
            drop_na() %>%
  mutate(Day = str_remove_all(Day, "X")) %>%
  mutate(Date = make_date(year = Year, month = Month, day = Day))

Flag.temp.2005 <- Flag.temp %>%
    filter(Year == 2005)

ggplot(data= Flag.temp.2005 ,
mapping=aes(x=Date , y=Temp ))+
geom_line(size = 0.5)
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



b) Create a line graph that gives the monthly average maximum temperature for 2013 - 2015. *Again the x-axis should be the date and span 3 years.*

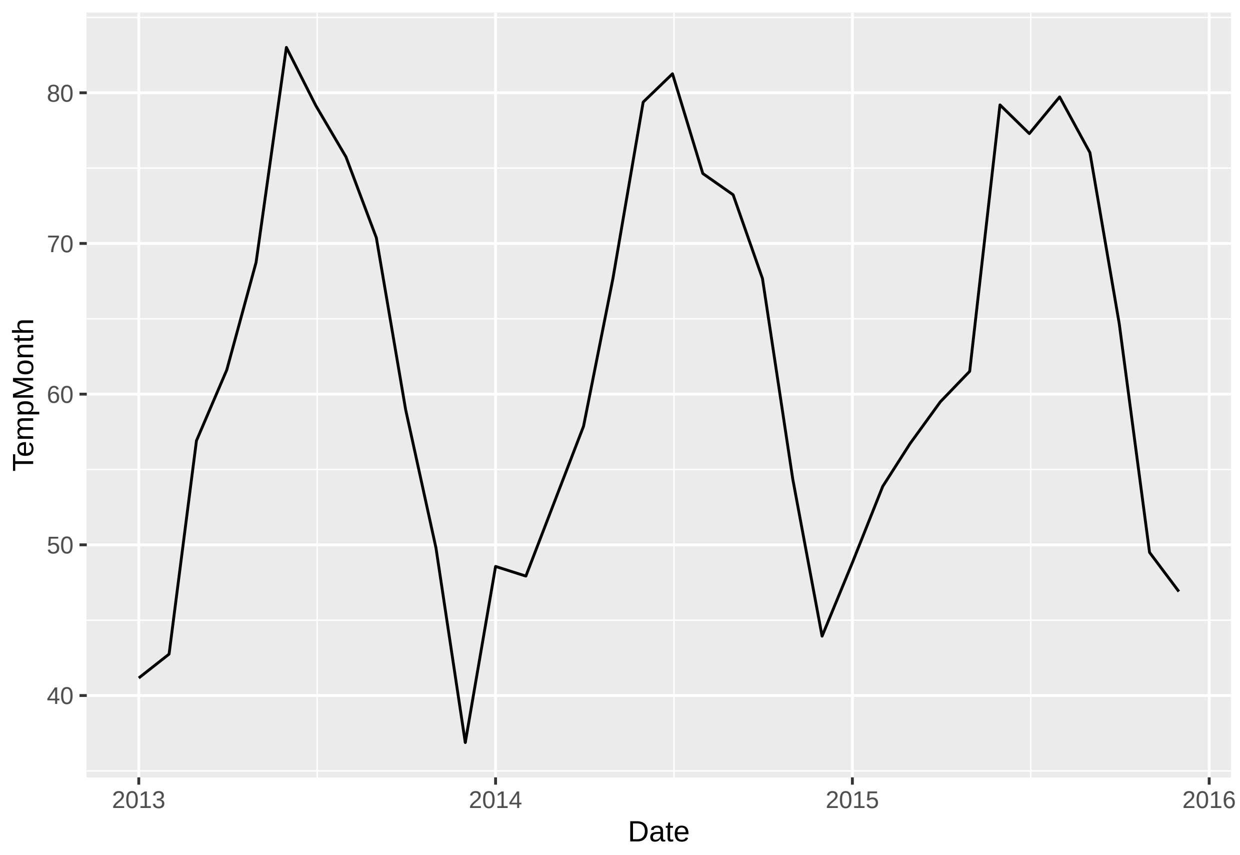
```r
Flag.temp.monthly <- Flag.temp %>%
            filter(Year == 2013 | Year == 2014 | Year == 2015) %>%
  select(Year, Date, Month, Temp) %>%
  group_by(Year, Month) %>%
  summarise(TempMonth = mean(Temp)) %>%
  mutate(Date = make_date(year = Year, month = Month))
```

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

```r
head(Flag.temp.monthly)
```

```
## # A tibble: 6 x 4
## # Groups:   Year [1]
##    Year Month TempMonth Date
##   <int> <int>     <dbl> <date>
## 1  2013     1      41.2 2013-01-01
## 2  2013     2      42.7 2013-02-01
## 3  2013     3      56.9 2013-03-01
## 4  2013     4      61.6 2013-04-01
## 5  2013     5      68.7 2013-05-01
## 6  2013     6      83.0 2013-06-01
```

```r
ggplot(data= Flag.temp.monthly ,
mapping=aes(x=Date , y=TempMonth ))+
geom_line(size = 0.5)
```



Exercise 3

For this problem we will consider two simple data sets.

```r
A <- tribble(
  ~Name, ~Car,
  'Alice', 'Ford F150',
  'Bob',   'Tesla Model III',
  'Charlie', 'VW Bug')

B <- tribble(
  ~First.Name, ~Pet,
  'Bob',  'Cat',
  'Charlie', 'Dog',
  'Alice', 'Rabbit')
```

a) Combine the data frames together to generate a data set with three rows and three columns using `join` commands.

```r
B <- B %>% mutate(Name = First.Name) %>% select(Name, Pet)
right_join(A, B, by = join_by(Name))
```

```
## # A tibble: 3 x 3
##   Name    Car             Pet
##   <chr>   <chr>           <chr>
## 1 Alice   Ford F150       Rabbit
## 2 Bob     Tesla Model III Cat
## 3 Charlie VW Bug          Dog
```

b) It turns out that Alice also has a pet guinea pig. Add another row to the `B` data set. Do this using either the base function `rbind`, or either of the `dplyr` functions `add_row` or `bind_rows`.

```
B <- B %>% add_row(Name = "Alice", Pet = "Guinea Pig")
B
```

```
## # A tibble: 4 x 2
##   Name    Pet
##   <chr>   <chr>
## 1 Bob     Cat
## 2 Charlie Dog
## 3 Alice   Rabbit
## 4 Alice   Guinea Pig
```

c) Combine again the `A` and `B` data sets together to generate a data set with four rows and three columns using `join` commands.

Note: You may want to also try using `cbind` to address questions (a) and (c). Leave this as a challenge question and focus on the easier to use `join` functions introduced in this chapter.

```
right_join(A, B, by=join_by(Name))
```

```
## # A tibble: 4 x 3
##   Name    Car             Pet
##   <chr>   <chr>           <chr>
## 1 Alice   Ford F150       Rabbit
## 2 Alice   Ford F150       Guinea Pig
## 3 Bob     Tesla Model III Cat
## 4 Charlie VW Bug          Dog
```

Exercise 4

The package `nycflights13` contains information about all the flights that arrived in or left from New York City in 2013. This package contains five data tables, but there are three data tables we will work with. The data table `flights` gives information about a particular flight, `airports` gives information about a particular airport, and `airlines` gives information about each airline. Create a table of all the flights on February 14th by Virgin America that has columns for the carrier, destination, departure time, and flight duration. Join this table with the airports information for the destination. Notice that because the column for the destination airport code doesn't match up between `flights` and `airports`, you'll have to use the `by=c("TableA.Col"="TableB.Col")` argument where you insert the correct names for `TableA.Col` and `TableB.Col`.

```
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 11 more variables: 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>
```

```r
head(airports)
```

```
## # A tibble: 6 x 8
##   faa   name                             lat   lon   alt    tz dst   tzone
##   <chr> <chr>                          <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport               41.1 -80.6  1044    -5 A     America/Ne~
## 2 06A   Moton Field Municipal Airport   32.5 -85.7   264    -6 A     America/Ch~
## 3 06C   Schaumburg Regional             42.0 -88.1   801    -6 A     America/Ch~
## 4 06N   Randall Airport                 41.4 -74.4   523    -5 A     America/Ne~
## 5 09J   Jekyll Island Airport           31.1 -81.4    11    -5 A     America/Ne~
## 6 0A9   Elizabethton Municipal Airport  36.4 -82.2  1593    -5 A     America/Ne~
```

```r
Table <- right_join(flights, airports, by=c("dest"="faa"))
Table <- Table %>% filter(month == 2 & day == 14 & carrier == "VX") %>%
  select(carrier, dest, dep_time, air_time)
Table
```

```
## # A tibble: 10 x 4
##    carrier dest  dep_time air_time
##    <chr>   <chr>    <int>    <dbl>
##  1 VX      LAX        706      347
##  2 VX      SFO        732      344
##  3 VX      LAX        909      341
##  4 VX      LAS        934      307
##  5 VX      SFO       1029      351
##  6 VX      LAX       1317      349
##  7 VX      LAX       1706      335
##  8 VX      SFO       1746      358
##  9 VX      SFO       1852      355
## 10 VX      LAX       2017      337
```