STA_445_HW4

Bianca L.

2023-10-26

```
library(tidyverse)
library(stringr)
library(lubridate)
library(mosaicData)
```

Problem 1: Chapter 13 Problem 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 dataset summarizing student surveys from several sections of an intro class. The two variables of interest for us are `Gender` and `Year` which are the students gender and year in college.

a. Download the dataset and correctly order the `Year` variable using the following:

```
Survey <- read.csv('https://www.lock5stat.com/datasets3e/StudentSurvey.csv', na.strings=c('',' '))
```

```
Survey1 <- Survey %>%
  mutate( Year = fct_recode(Year, `Freshman` = 'FirstYear'),
          Year = fct_relevel(Year, 'Freshman', 'Sophomore','Junior', 'Senior') )
```

b.Using some combination of `dplyr` functions, produce a data set with eight rows that contains the number of responses for each gender:year combination. Make sure your table orders the `Year` variable in the correct order of `First Year`, `Sophmore`, `Junior`, and then `Senior`.

```
Survey1 %>%
  group_by(Sex, Year) %>%
  summarize(number = n()) %>%
  drop_na()
```

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

```
## # A tibble: 8 x 3
## # Groups:   Sex [2]
##   Sex   Year      number
##   <chr> <fct>      <int>
## 1 F     Freshman      43
## 2 F     Sophomore     96
## 3 F     Junior        18
## 4 F     Senior        10
## 5 M     Freshman      51
## 6 M     Sophomore     99
## 7 M     Junior        17
## 8 M     Senior        26
```

c. Using `tidyr` commands, produce a table of the number of responses.

```
Survey1 %>%
  group_by(Sex, Year) %>%
  summarize(number = n()) %>%
  drop_na() %>%
  pivot_wider( names_from= Year, values_from=number )
```

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

```
## # A tibble: 2 x 5
## # Groups:   Sex [2]
##   Sex   Freshman Sophomore Junior Senior
##   <chr>    <int>     <int>  <int>  <int>
## 1 F           43        96     18     10
## 2 M           51        99     17     26
```

Problem 2: Chapter 13 Problem 2

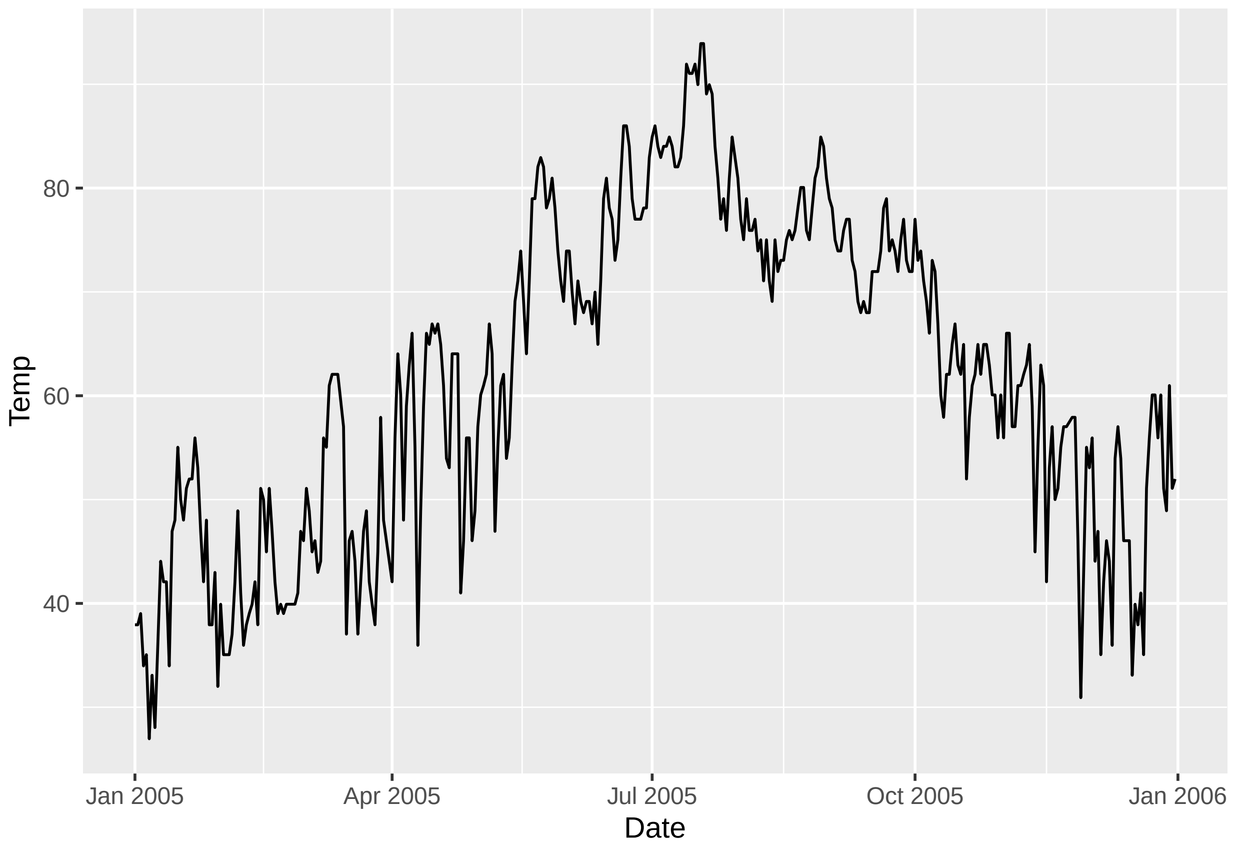
From the book website, there is a .csv file of the daily maximum temperature in Flagstaff at the Pulliam Airport. The direction link is at:

```
temp2005 <- read.csv('https://raw.githubusercontent.com/dereksonderegger/444/master/data-raw/FlagMaxTem
```

a. Create a line graph that gives the daily maximum temperature for 2005.

```
temp2005B <- temp2005 %>% pivot_longer(
    X1:X31,
    names_to  = 'Date',
    values_to = 'Temp') %>%
  select(Year, Month, Date, Temp) %>%
  mutate( Day = str_remove(Date, pattern="X")) %>%
  mutate(Date = ymd(paste(Year, Month, Day))) %>%
  drop_na() %>%
  select(Date, Temp)
```

```
ggplot(data=temp2005B) +
  geom_line(aes(x=Date, y=Temp))
```

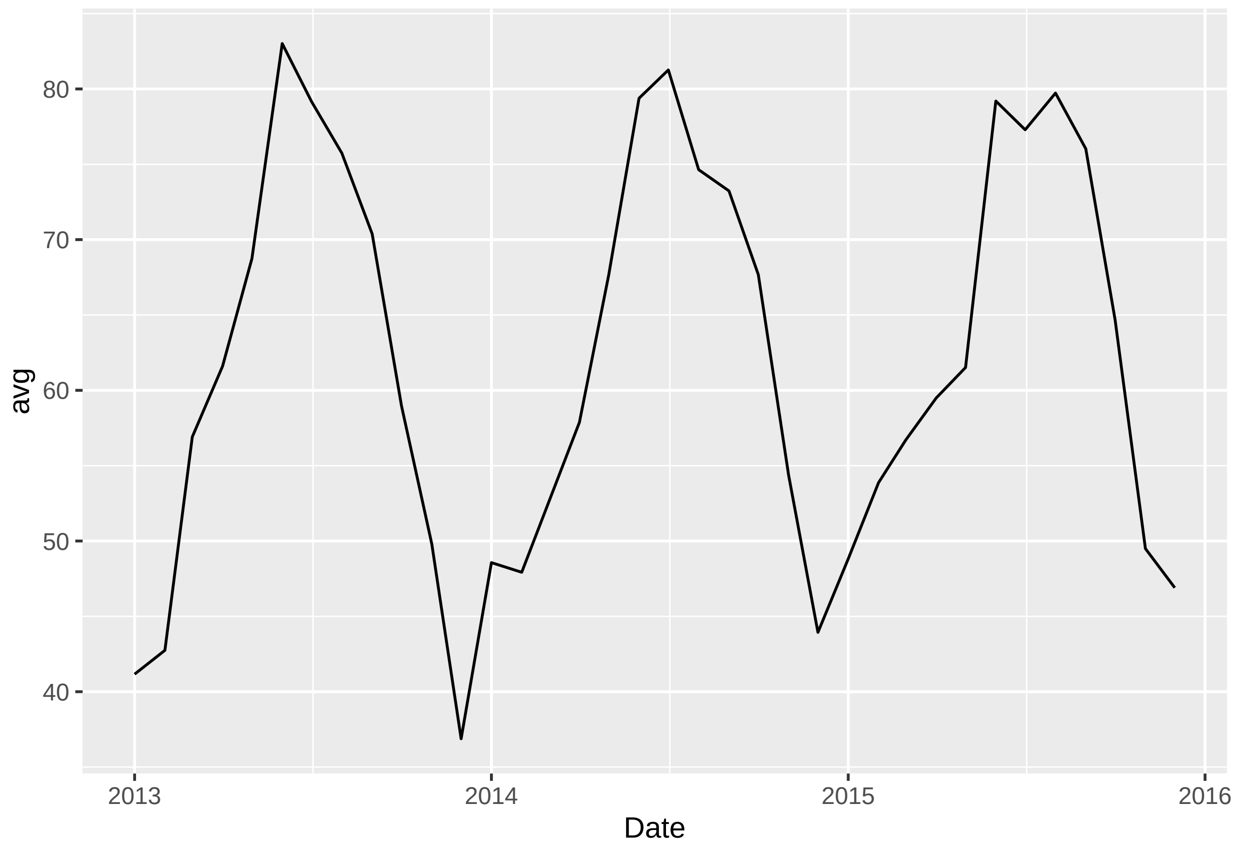


b. Create a line graph that gives the monthly average maximum temperature for 2013 - 2015.

```
temp2013 <- read.csv('https://raw.githubusercontent.com/dereksonderegger/444/master/data-raw/FlagMaxTemp
```

```
temp2013b <- temp2013 %>% pivot_longer(
    X1:X31,
    names_to  = 'Date',
    values_to = 'Temp') %>%
  select(Year, Month, Date, Temp) %>%
  drop_na() %>%
  group_by(Year, Month) %>%
  summarize(avg = mean(Temp)) %>%
  mutate(Date = ym(paste(Year, Month)))
```

```
ggplot(data=temp2013b) +
  geom_line(aes(x=Date, y=avg))
```



Problem 3: Chapter 13 Problem 4

For this problem we will consider two simple data sets.

```
    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. Squish the data frames together to generate a data set with three rows and three columns. Do two ways: first using `cbind` and then using one of the `dplyr join` commands.

```
B <- arrange(B, First.Name)
cbind(A, Pet = B$Pet)
```

```
##      Name             Car    Pet
## 1   Alice       Ford F150 Rabbit
## 2     Bob Tesla Model III    Cat
## 3 Charlie          VW Bug    Dog
```

```
inner_join(A, B, by=c("Name"="First.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 <- rbind(B, c("Alice", "Guinea Pig"))
B
```

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

c. Squish the `A` and `B` data sets together to generate a data set with four rows and three columns. Do this two ways: first using `cbind` and then using one of the `dplyr join` commands. Which was easier to program? Which is more likely to have an error.

```
inner_join(A, B, by=c("Name"="First.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
```

```
A <- rbind(A, c("Alice", "Ford F150"))
A <- arrange(A, Name)
B <- arrange(B, First.Name)
cbind(A, Pet=B$Pet)
```

```
##      Name             Car        Pet
## 1   Alice       Ford F150     Rabbit
## 2   Alice       Ford F150 Guinea Pig
## 3     Bob Tesla Model III        Cat
## 4 Charlie          VW Bug        Dog
```

Using cbind is way more difficult. I was forced to add rows and arrange the tables based on the names before I could actually use cbind.

Problem 4: Chapter 13 Problem 5

Data table joins are extremely common because effective database design almost always involves having multiple tables for different types of objects. To illustrate both the table joins and the usefulness of multiple tables we will develop a set of data frames that will represent a credit card company's customer data base. We will have tables for Customers, Retailers, Cards, and Transactions. Below is code that will create and populate these tables.

```
    Customers <- tribble(
      ~PersonID, ~Name, ~Street, ~City, ~State,
```

```r
    1, 'Derek Sonderegger',  '231 River Run', 'Flagstaff', 'AZ',
    2, 'Aubrey Sonderegger', '231 River Run', 'Flagstaff', 'AZ',
    3, 'Robert Buscaglia', '754 Forest Heights', 'Flagstaff', 'AZ',
    4, 'Roy St Laurent', '845 Elk View', 'Flagstaff', 'AZ')

  Retailers <- tribble(
    ~RetailID, ~Name, ~Street, ~City, ~State,
    1, 'Kickstand Kafe', '719 N Humphreys St', 'Flagstaff', 'AZ',
    2, 'MartAnnes', '112 E Route 66', 'Flagstaff', 'AZ',
    3, 'REI', '323 S Windsor Ln', 'Flagstaff', 'AZ' )

  Cards <- tribble(
    ~CardID, ~PersonID, ~Issue_DateTime, ~Exp_DateTime,
    '9876768717278723',  1,  '2019-9-20 0:00:00', '2022-9-20 0:00:00',
    '5628927579821287',  2,  '2019-9-20 0:00:00', '2022-9-20 0:00:00',
    '7295825498122734',  3,  '2019-9-28 0:00:00', '2022-9-28 0:00:00',
    '8723768965231926',  4,  '2019-9-30 0:00:00', '2022-9-30 0:00:00' )

  Transactions <- tribble(
    ~CardID, ~RetailID, ~DateTime, ~Amount,
    '9876768717278723', 1, '2019-10-1 8:31:23',    5.68,
    '7295825498122734', 2, '2019-10-1 12:45:45',  25.67,
    '9876768717278723', 1, '2019-10-2 8:26:31',    5.68,
    '9876768717278723', 1, '2019-10-2 8:30:09',    9.23,
    '5628927579821287', 3, '2019-10-5 18:58:57',  68.54,
    '7295825498122734', 2, '2019-10-5 12:39:26',  31.84,
    '8723768965231926', 2, '2019-10-10 19:02:20', 42.83)

  Cards <- Cards %>%
    mutate( Issue_DateTime = lubridate::ymd_hms(Issue_DateTime),
            Exp_DateTime   = lubridate::ymd_hms(Exp_DateTime) )
  Transactions <- Transactions %>%
    mutate( DateTime = lubridate::ymd_hms(DateTime))
```

a. Create a table that gives the credit card statement for Derek. It should give all the transactions, the amounts, and the store name. Write your code as if the only initial information you have is the customer's name. *Hint: Do a bunch of table joins, and then filter for the desired customer name. To be efficient, do the filtering first and then do the table joins.*

```r
A <- inner_join(Customers, Cards)
```

```
## Joining with `by = join_by(PersonID)`
```

```r
B <- inner_join(A, Transactions)
```

```
## Joining with `by = join_by(CardID)`
```

```r
C <- inner_join(B, Retailers, by=c("RetailID"="RetailID"))
```

```r
C %>% filter(Name.x == "Derek Sonderegger") %>% select(Name.x, CardID, DateTime, Amount, Name.y)
```

```
## # A tibble: 3 x 5
##   Name.x            CardID           DateTime            Amount Name.y
##   <chr>             <chr>            <dttm>               <dbl> <chr>
## 1 Derek Sonderegger 9876768717278723 2019-10-01 08:31:23   5.68 Kickstand Kafe
## 2 Derek Sonderegger 9876768717278723 2019-10-02 08:26:31   5.68 Kickstand Kafe
## 3 Derek Sonderegger 9876768717278723 2019-10-02 08:30:09   9.23 Kickstand Kafe
```

b. Aubrey has lost her credit card on Oct 15, 2019. Close her credit card at 4:28:21 PM and issue her a new credit card in the Cards table. Hint: Using the Aubrey's name, get necessary CardID and PersonID and save those as cardID and personID. Then update the Cards table row that corresponds to the cardID so that the expiration date is set to the time that the card is closed. Then insert a new row with the personID for Aubrey and a new CardID number that you make up.

```
personID <- Customers %>% filter(Name=='Aubrey Sonderegger') %>% pull(PersonID)

personID
```

```
## [1] 2
```

```
#personID <- personID[[1]]

#personID

#cardID <- Cards %>% filter(PersonID==personID) %>% select(CardID)

#cardID <- cardID[[1]]
```