Module13

Joe Vargovich

10/13/2020

Exercise 1 - Table for gender and year combinations

```
#a. Load the data.
Survey <- read.csv('http://www.lock5stat.com/datasets/StudentSurvey.csv', na.strings=c('',' '))

#b. Cleanup and order by the Year variable. We need the year and gender columns.
Survey = Survey %>%
  select(Gender, Year) %>%
  drop_na()

Survey$Year <- factor(Survey$Year, levels = c('FirstYear', 'Sophomore', 'Junior', 'Senior'))
classCounts <- Survey[order(Survey$Year), ]

classCounts = classCounts %>%
  count(Year, Gender)
classCounts
```

```
##        Year Gender  n
## 1 FirstYear      F 43
## 2 FirstYear      M 51
## 3 Sophomore      F 96
## 4 Sophomore      M 99
## 5    Junior      F 18
## 6    Junior      M 17
## 7    Senior      F 10
## 8    Senior      M 26
```

```
classCounts = classCounts %>%
  pivot_wider(names_from=Year, values_from=n) %>%
  mutate(Gender = if_else(Gender=='F', 'Female','Male'))

head(classCounts)
```

```
## # 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 - US Govt Expenditures

```
#a. Download and load the data, skip the first two garbage rows.
expDf = read_excel("hist03z2.xls", skip=2)

#b. Rename function or subfunction to Department
expDf = rename(expDf, `Department` = `Function and Subfunction`)

#c. Remove rows Total, Subtotal, (On-budget), (Off-budget). Remove rows at the bottom.
#First, remove bottom rows.
expDf = expDf %>% filter(row_number() <= n()-4)

#Then filter out terms we do not want in our rows.
expDf = filter(expDf, !str_detect(Department, "(Total|Subtotal|(On-budget)|(Off-budget))"))

#d.Create a new column for ID_number and parse the Department column for it.
#Parse Department and extract ids using str_extract
expDf = expDf %>% mutate(
  Id_number = str_extract(Department,"([0-9][0-9][0-9])")
)

#e. Create Function and Subfunction columns.
#First, copy Department to Subfunction
expDf = expDf %>% mutate(
  Subfunction = Department,
  Function = ifelse(is.na(`2015`), Department, NA)
)
#Fill in NAs
expDf = expDf %>%
  fill(Function)

#f. Remove function rows that did not have data for the 2015 column
expDf = expDf %>%
  na.omit(cols=seq_along(`2015`))

#g. Reshape the data into four columns: Function, Subfunction, Year, and Amount.
reshapedDf = expDf %>%
  pivot_longer(
    `1962`:`2015`,
    names_to = 'Year',
    values_to = 'Amount') %>%
    select(Subfunction:Amount) %>%
    mutate(Function = str_remove_all(Function, '\\d|:'))
head(reshapedDf)
```

```
## # A tibble: 6 x 4
##   Subfunction                          Function                        Year  Amount
##   <chr>                                <chr>                           <chr> <chr>
## 1 053 Atomic energy defense activit~ " Department of Defense-Milit~ 1962  2074
## 2 053 Atomic energy defense activit~ " Department of Defense-Milit~ 1963  2041
## 3 053 Atomic energy defense activit~ " Department of Defense-Milit~ 1964  1902
## 4 053 Atomic energy defense activit~ " Department of Defense-Milit~ 1965  1620
## 5 053 Atomic energy defense activit~ " Department of Defense-Milit~ 1966  1466
## 6 053 Atomic energy defense activit~ " Department of Defense-Milit~ 1967  1277
```

```r
#h, i. Remove rows with ..... values and non-numeric values.
reshapedDf = filter(reshapedDf, !str_detect(Amount, ".....")) %>%
  filter(!str_detect(Year, "(TQ)"))

reshapedDf$Amount = as.numeric(as.character(reshapedDf$Amount))
reshapedDf$Year = as.numeric(as.character(reshapedDf$Year))
head(reshapedDf)
```

```
## # A tibble: 6 x 4
##   Subfunction                          Function                          Year Amount
##   <chr>                                <chr>                            <dbl>  <dbl>
## 1 053 Atomic energy defense activit~ " Department of Defense-Milit~  1962   2074
## 2 053 Atomic energy defense activit~ " Department of Defense-Milit~  1963   2041
## 3 053 Atomic energy defense activit~ " Department of Defense-Milit~  1964   1902
## 4 053 Atomic energy defense activit~ " Department of Defense-Milit~  1965   1620
## 5 053 Atomic energy defense activit~ " Department of Defense-Milit~  1966   1466
## 6 053 Atomic energy defense activit~ " Department of Defense-Milit~  1967   1277
```

```r
#j. Line graph to compare spending for National Defense, Health, Medicare,
# Income Security, and Social Security for each of the years 2001 through 2015
#First, grab only measurements from 2001 to 2015.
finalDf = filter(reshapedDf, Year >= 2001 & Year <= 2015 ) %>%
  filter(str_detect(Function, "Department of Defense-Military|Health|Medicare|Income Security|Social Se
  group_by(Function, Year)
  #Some sort of sum function that I don't know how to use, then the graphing.
```

Exercise 3 - Working with two simple data sets.

```r
#Given data
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. Join together with both cbind and join commands.
#cbind
bindedData = cbind(A, B)
head(bindedData)
```

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

```r
#left join
joinedData = left_join(A,B, by=c("Name"="First.Name"))
joinedData
```

```
## # 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. Add row for Alice's guinea pig, add to table B!! Not the join!
guineaRow = tibble(First.Name="Alice", Pet="Guinea Pig")
B = rbind(B, guineaRow)
B
```

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

```
#c. Combine the two datasets together with cbind and dplyr join.
#In order to bind the two dataframes, we need to add a row to the B dataset so we can directly column b
placeholderRow = tibble(Name=NA, Car=NA)

biggerA = rbind(A, placeholderRow)
biggerA
```

```
## # A tibble: 4 x 2
##   Name    Car
##   <chr>   <chr>
## 1 Alice   Ford F150
## 2 Bob     Tesla Model III
## 3 Charlie VW Bug
## 4 <NA>    <NA>
```

```
bindedData2 = cbind(biggerA,B)
bindedData2 #This is messier than the join operation.
```

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

```
#Accomplish this with a simple join instead.
joinedData2 = right_join(A,B, by=c("Name"="First.Name"))

joinedData2 #Much cleaner! No NAs necessary!
```

```
## # 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 - Table joins for Customers, Retailers, Cards, and Transactions.

```
Customers <- tribble(
  ~PersonID, ~Name, ~Street, ~City, ~State,
  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. Gives all transactions, amounts, and
head(Customers)
```

```
## # A tibble: 4 x 5
##   PersonID Name               Street             City      State
##      <dbl> <chr>              <chr>              <chr>     <chr>
## 1        1 Derek Sonderegger  231 River Run      Flagstaff AZ
## 2        2 Aubrey Sonderegger 231 River Run      Flagstaff AZ
## 3        3 Robert Buscaglia   754 Forest Heights Flagstaff AZ
## 4        4 Roy St Laurent     845 Elk View       Flagstaff AZ
```

```
#Filter the customers to Derek
custDerek = Customers %>%
  filter(Name == "Derek Sonderegger") %>%
  select(Name)
```

```
#Filter the cards to Derek
cardDerek = Cards %>%
  filter(PersonID == 1) %>%
  select(PersonID,CardID)

#Grab the card ID
derekCardID = cardDerek$CardID[1]

#Filter the transactions based on CardID
transacDerek = Transactions %>%
  filter(CardID == derekCardID)

#Filter the retailers
retailersDerek = Retailers %>%
  filter(RetailID == 1) %>%
  select(RetailID, Name, Street)

#Finally, join these filtered dataframes.
derekStatement = left_join(transacDerek, retailersDerek) %>%
  select(!RetailID)
```

```
## Joining, by = "RetailID"
```

```
derekStatement
```

```
## # A tibble: 3 x 5
##   CardID           DateTime            Amount Name           Street
##   <chr>            <dttm>               <dbl> <chr>          <chr>
## 1 9876768717278723 2019-10-01 08:31:23   5.68 Kickstand Kafe 719 N Humphreys St
## 2 9876768717278723 2019-10-02 08:26:31   5.68 Kickstand Kafe 719 N Humphreys St
## 3 9876768717278723 2019-10-02 08:30:09   9.23 Kickstand Kafe 719 N Humphreys St
```

```
#b. Aubrey lost her credit card on Oct 15 2019
#Close Aubrey's credit card at 4:28:21PM and issue a new credit card in the Cards table.
#Filter the Customers dataframe to get needed info
custAubrey = Customers %>%
  filter(Name == "Aubrey Sonderegger") %>%
  select(Name, PersonID)

#Grab Aubrey's ID
aubreyID = custAubrey$PersonID[1]
aubreyCard = Cards %>%
  filter(PersonID == aubreyID)
#head(aubreyCard)

#Make a new card entry for Aubrey
newCard <- tibble( CardID='6969927542021287', PersonID=2, Issue_DateTime=ymd_hms('2019-10-15 4:28:22'),

#Add the new card
Cards = Cards %>%
  mutate(Exp_DateTime = if_else(PersonID == aubreyID, ymd_hms('2019-10-15 4:28:21'), Exp_DateTime )) %>%
  full_join(.,newCard)
```

```
## Joining, by = c("CardID", "PersonID", "Issue_DateTime", "Exp_DateTime")
```

```r
#c. Generate new transaction for new card at KickStand Kafe.
card = newCard$CardID
retailId = 1
datetime = ymd_hms('2019-10-16 14:30:21')
amount = 4.98
newPurchase = tibble(CardID = card, RetailID=retailId, DateTime=datetime, Amount = amount )

#d. If the card is currently valid, this should return exactly 1 row.
Valid_Cards <- Cards %>%
  filter(CardID == card, Issue_DateTime <= datetime, datetime <= Exp_DateTime)

# If the transaction is valid, insert the transaction into the table
if( nrow(Valid_Cards) == 1){
  Transactions = rbind(Transactions, newPurchase)
}else{
  print('Card Denied')
}

#e. Create statement for Aubrey's cards.
#Filter the customers to Aubrey
custAubrey = Customers %>%
  filter(Name == "Aubrey Sonderegger") %>%
  select(Name)

#Filter the cards to Aubrey
cardsAubrey = Cards %>%
  filter(PersonID == 2) %>%
  select(PersonID,CardID)

#Grab the card ID
aubreyCardIDs = cardsAubrey$CardID[1:2]

#Filter the transactions based on CardID
transacAubrey = Transactions %>%
  filter(CardID == aubreyCardIDs[1:2])

#Finally, join the transactionAubrey and retailers dataframes.
aubreyStatement = left_join(transacAubrey, Retailers) %>%
  select(CardID, DateTime, Amount, Name)
```

```
## Joining, by = "RetailID"
```

```r
aubreyStatement
```

```
## # A tibble: 2 x 4
##   CardID           DateTime            Amount Name
##   <chr>            <dttm>               <dbl> <chr>
## 1 5628927579821287 2019-10-05 18:58:57  68.5  REI
## 2 6969927542021287 2019-10-16 14:30:21   4.98 Kickstand Kafe
```

#Exercise 5 - nycflights13

```r
data('flights', package='nycflights13')
write.csv(flights, 'flights.csv')
data('airports', package='nycflights13')
```

```r
write.csv(airports, 'airports.csv')
data('airlines', package='nycflights13')
write.csv(airlines, 'airlines.csv')

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
## # ... with 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/New_Y~
## 2 06A   Moton Field Municipal Airp~  32.5 -85.7   264    -6 A     America/Chica~
## 3 06C   Schaumburg Regional          42.0 -88.1   801    -6 A     America/Chica~
## 4 06N   Randall Airport              41.4 -74.4   523    -5 A     America/New_Y~
## 5 09J   Jekyll Island Airport        31.1 -81.4    11    -5 A     America/New_Y~
## 6 0A9   Elizabethton Municipal Air~  36.4 -82.2  1593    -5 A     America/New_Y~
```

```r
head(airlines)
```

```
## # A tibble: 6 x 2
##   carrier name
##   <chr>   <chr>
## 1 9E      Endeavor Air Inc.
## 2 AA      American Airlines Inc.
## 3 AS      Alaska Airlines Inc.
## 4 B6      JetBlue Airways
## 5 DL      Delta Air Lines Inc.
## 6 EV      ExpressJet Airlines Inc.
```

```r
#Filter to Virign America and the date/time, and then join with selected columns.
virginAmerica = flights %>%
  filter(str_detect(carrier, "VX")) %>%
  filter(month == 2 & day == 14) %>% #Filter to the date we want!
  select(carrier, dest, dep_time, air_time) %>%
  left_join(., airports, by=c("dest"="faa"))

head(virginAmerica)
```

```
## # A tibble: 6 x 11
##   carrier dest  dep_time air_time name       lat   lon   alt    tz dst   tzone
##   <chr>   <chr>    <int>    <dbl> <chr>    <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 VX      LAX        706      347 Los Ang~  33.9 -118.   126    -8 A     Americ~
```

```
## 2 VX      SFO        732      344 San Fra~  37.6 -122.    13    -8 A     Americ~
## 3 VX      LAX        909      341 Los Ang~  33.9 -118.   126    -8 A     Americ~
## 4 VX      LAS        934      307 Mc Carr~  36.1 -115.  2141    -8 A     Americ~
## 5 VX      SFO       1029      351 San Fra~  37.6 -122.    13    -8 A     Americ~
## 6 VX      LAX       1317      349 Los Ang~  33.9 -118.   126    -8 A     Americ~
```