Week 2 Exercises

Ray Chandonnet

November 5, 2022

Please complete all exercises below. You may use stringr, lubridate, or the forcats library.

```
#add libraries
library(stringr)
library(forcats)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

Exercise 1

Read the sales_pipe.txt file into an R data frame as sales.

```
# Given the challenges of using the original sales_pipe.txt file on a mac, I
# created a different version using UTF-16LE which I called sales_pipe_mac.txt.  That
# alternative sample file is in my Github if you want to review it. This successfuly reads
# in using read_delim with the right file encoding specified. Doing this allowed me to still
# do exercises 2 & 3, whereas if I had imported the CSV file or the the XLSX file, the first
# column name was already fixed and the dates formatted properly in the source file.  For
# some reason, when I read in this file there is an extra column at the end with blank
# values and column header "X", so my code deletes that last column

sales <- read.delim(file="Data/sales_pipe_mac.txt",
                    stringsAsFactors=FALSE,
                    fileEncoding = "UTF-16LE",
                    header=TRUE,
                    sep="|",
                    fill=TRUE)

# Below I show the first row in Sales to show the phantom last column "X"
sales[1,]
```

```
##   ï..Row.ID       Order.ID Order.Date        Ship.Date    Ship.Mode
## 1         1 CA-2016-152156    11/8/16 November 11 2016 Second Class
##   Customer.ID Customer.Name  Segment       Country      City    State
## 1    CG-12520   Claire Gute Consumer United States Henderson Kentucky
##   Postal.Code Region      Product.ID  Category Sub.Category
## 1       42420  South FUR-BO-10001798 Furniture    Bookcases
##                         Product.Name  Sales Quantity Discount  Profit X
## 1 Bush Somerset Collection Bookcase 261.96        2        0 41.9136
```

```
length(sales)
```

```
## [1] 22
```

```
# This code finds the last column and deletes it
length(sales)
```

```
## [1] 22
```

```
sales <- sales[,-length(sales)]
length(sales)
```

```
## [1] 21
```

```
# Showing that that column is now gone
sales[1,]
```

```
##   ï..Row.ID       Order.ID Order.Date        Ship.Date    Ship.Mode
## 1         1 CA-2016-152156    11/8/16 November 11 2016 Second Class
##   Customer.ID Customer.Name  Segment       Country      City    State
## 1    CG-12520   Claire Gute Consumer United States Henderson Kentucky
##   Postal.Code Region      Product.ID  Category Sub.Category
## 1       42420  South FUR-BO-10001798 Furniture    Bookcases
##                         Product.Name  Sales Quantity Discount  Profit
## 1 Bush Somerset Collection Bookcase 261.96        2        0 41.9136
```

Exercise 2

You can extract a vector of columns names from a data frame using the colnames() function. Notice the first column has some odd characters. Change the column name for the FIRST column in the sales date frame to Row.ID.

Note: You will need to assign the first element of colnames to a single character..

Ray Note: I must confess I don't know what the above Note means - it seems I can successfully just rename the first element in the colnames() vector as per below

```
colnames(sales)[1] <- "Row.ID"
sales[1,]
```

```
##   Row.ID       Order.ID Order.Date        Ship.Date    Ship.Mode
## 1      1 CA-2016-152156    11/8/16 November 11 2016 Second Class
##   Customer.ID Customer.Name  Segment       Country      City    State
## 1    CG-12520   Claire Gute Consumer United States Henderson Kentucky
##   Postal.Code Region      Product.ID  Category Sub.Category
## 1       42420  South FUR-BO-10001798 Furniture    Bookcases
##                         Product.Name  Sales Quantity Discount  Profit
## 1 Bush Somerset Collection Bookcase 261.96        2        0 41.9136
```

Exercise 3

Convert both Order.ID and Order.Date to date vectors within the sales data frame. What is the number of days between the most recent order and the oldest order? How many years is that? How many weeks?

Note: Use lubridate

```
# First convert both those vectors to dates using the mdy function in lubridate.  Note that
# I am converting Order.Date and Ship. Date, not Order.ID and Order.Date

sales$Order.Date <- lubridate::mdy(sales$Order.Date)
sales$Ship.Date <- lubridate::mdy(sales$Ship.Date)
```

```r
most_recent <- max(sales$Order.Date)  #Find the most recent order date
most_recent
```

```
## [1] "2017-12-30"
```

```r
earliest <- min(sales$Order.Date)  #Find the oldest order date
earliest
```

```
## [1] "2014-01-03"
```

```r
# To calculate elapsed days we can use the difftime base function in R. Since the difftime
# function returns the result in a "difftime" object, I also convert that object to numeric
# in case I want to do some math with it.

elapsed_days_difftime <- difftime(most_recent, earliest, units = "days") #calc elapsed days
#                                                                          as difftime object
elapsed_days_difftime
```

```
## Time difference of 1457 days
```

```r
elapsed_days <- as.numeric(elapsed_days_difftime) #convert to numeric
elapsed_days
```

```
## [1] 1457
```

```r
# Now I use the time_length function of lubridate to calculate elapsed years and months.
# Cannot use the base difftime function for this and specify "years" or "months" as unit
# because months and years are not of standard length

elapsed_years <- lubridate::time_length(elapsed_days_difftime,"years")
elapsed_years
```

```
## [1] 3.991781
```

```r
elapsed_months <- lubridate::time_length(elapsed_days_difftime,"months")
elapsed_months
```

```
## [1] 47.90137
```

Exercise 4

What is the average number of days it takes to ship an order?

```r
# I do this by first creating a new vector "shipping_days" that calculates the elapsed days
# between shipping date and order date, in numeric form, and then calculating the mean of
# that vector.  Note I am printing out the first few rows/columns of the sales data frame
# and the first few rows of the shipping_days vector to prove it works

# create vector of shipping days
shipping_days <- as.numeric(difftime(sales$Ship.Date,sales$Order.Date,units = "days"))
sales[1:3,1:4]
```

```
##   Row.ID       Order.ID Order.Date  Ship.Date
## 1      1 CA-2016-152156 2016-11-08 2016-11-11
## 2      2 CA-2016-152156 2016-11-08 2016-11-11
## 3      3 CA-2016-138688 2016-06-12 2016-06-16
```

```
shipping_days[1:3]
```

```
## [1] 3 3 4
```

```
shipping_days_avg <- mean(shipping_days)
cat("Average # of days to ship :", shipping_days_avg)
```

```
## Average # of days to ship : 3.958175
```

Exercise 5

How many customers have the first name Bill? You will need to split the customer name into first and last name segments and then use a regular expression to match the first name bill. Use the length() function to determine the number of customers with the first name Bill in the sales data.

```
# So I can do this a lot of different ways, and I can also interpret the question a couple
# of different ways.  The first way to interpret the question is to ask literally "How many
# rows contain the first name Bill in Customer.Name".  The second is more subtle, asking
# "how many of our customers are named Bill".  I imagine what you wanted was the first, but
# I decided to also see if I could figure out how to do the second interpretation, which is
# to find the number of UNIQUE customers with first name Bill.

# Method 1 - Finding # of rows where customer first name is Bill by splitting the customer
# name in two.  I like this method if you want to retain a table with subset of matching
# customers

# split first from last (I turn it into a data frame rather than the default matrix output
# for the str_split_fixed function in stringr)
names_frame <- as.data.frame(stringr::str_split_fixed(sales$Customer.Name," ",2))
colnames(names_frame)=c("First.Name","Last.Name") # assign column names for easier notation
names_frame[1:8,]  #show the first few rows to prove it worked
```

```
##   First.Name Last.Name
## 1     Claire      Gute
## 2     Claire      Gute
## 3     Darrin  Van Huff
## 4       Sean O'Donnell
## 5       Sean O'Donnell
## 6    Brosina   Hoffman
## 7    Brosina   Hoffman
## 8    Brosina   Hoffman
```

```
first_name_is_Bill <- names_frame$First.Name == "Bill" #create boolean vector of matches
all_the_Bills_method1 <- names_frame[first_name_is_Bill,] #subset the data frame
all_the_Bills_method1
```

```
##      First.Name Last.Name
## 552        Bill Donatelli
## 910        Bill    Eplett
## 911        Bill    Eplett
## 912        Bill    Eplett
## 1081       Bill     Tyler
## 1082       Bill     Tyler
## 1440       Bill Donatelli
## 1441       Bill Donatelli
```

```
## 1643       Bill   Stewart
## 1970       Bill Donatelli
## 2410       Bill    Eplett
## 2448       Bill Donatelli
## 2499       Bill     Tyler
## 2614       Bill  Overfelt
## 3529       Bill Donatelli
## 3792       Bill     Tyler
## 3793       Bill     Tyler
## 4179       Bill Donatelli
## 4251       Bill   Shonely
## 4278       Bill   Shonely
## 4279       Bill   Shonely
## 4280       Bill   Shonely
## 4526       Bill    Eplett
## 4527       Bill    Eplett
## 4528       Bill    Eplett
## 4636       Bill   Stewart
## 4949       Bill Donatelli
## 4950       Bill Donatelli
## 5179       Bill     Tyler
## 5180       Bill     Tyler
## 5181       Bill     Tyler
## 5295       Bill  Overfelt
## 5367       Bill Donatelli
## 5368       Bill Donatelli
## 5369       Bill Donatelli
## 5446       Bill    Eplett
## 5605       Bill   Stewart
## 5968       Bill    Eplett
## 6885       Bill Donatelli
## 7287       Bill   Shonely
## 7288       Bill   Shonely
## 7311       Bill Donatelli
## 7371       Bill     Tyler
## 7539       Bill   Stewart
## 7540       Bill   Stewart
## 7541       Bill   Stewart
## 7542       Bill   Stewart
## 7543       Bill   Stewart
## 7544       Bill   Stewart
## 7545       Bill   Stewart
## 7999       Bill   Shonely
## 8000       Bill   Shonely
## 8246       Bill   Shonely
## 8352       Bill  Overfelt
## 8353       Bill  Overfelt
## 8354       Bill  Overfelt
## 8355       Bill  Overfelt
## 8356       Bill  Overfelt
## 8619       Bill     Tyler
## 8747       Bill  Overfelt
## 8748       Bill  Overfelt
## 8749       Bill  Overfelt
```

```
## 8895       Bill Donatelli
## 8896       Bill Donatelli
## 8897       Bill Donatelli
## 8898       Bill Donatelli
## 9494       Bill  Overfelt
## 9495       Bill  Overfelt
## 9613       Bill Donatelli
## 9618       Bill   Stewart
## 9619       Bill   Stewart
## 9620       Bill   Stewart
```

```
# One comment / question here:  I also could do this subsetting in one line of code like:
# all_the_Bills_method1 <- names_frame[names_frame$First.Name == "Bill",]
# I would be curious to know which you prefer as a professional.   I usually like to make my
# code really readable, so i break out complex operations into their separate components.
# But it also makes my code longer, and I suppose I may be using up more memory by
# explicitly storing the booleans as a vector variable as opposed to passing them as an
# argument to the slicing operation...thoughts?

numBills_method1 <- length(all_the_Bills_method1$First.Name) #Count them
cat("Using Method 1, we identified", numBills_method1,"rows with first name='Bill'")
```

```
## Using Method 1, we identified 72 rows with first name='Bill'
```

```
#
# Method 2 - Finding # of rows where customer first name is Bill without splitting the
# customer name in two, by looking for all Customer.Name that start with "Bill ".  I like
# this method if you just want the answer, because it's more efficient

# create vector of Booleans where first five characters of Customer.Name are "Bill "  ;
# Note I include the space at the end to avoid accidentally getting first names like Billy,
# Billilyn, etc
first_five_matches_Bill <- stringr::str_starts(sales$Customer.Name,"Bill ")
all_the_Bills_method2 <- sales$Customer.Name[first_five_matches_Bill] #Subset table matches
numBills_method2 <- length(all_the_Bills_method2) #Count them
cat("Using Method 2, we identified", numBills_method2,"rows with first name='Bill'")
```

```
## Using Method 2, we identified 72 rows with first name='Bill'
```

```
#
# Method 3 - In this "bonus" answer, I identify and count all the UNIQUE customers named
# Bill, so I am answering a different question, really "How many of our customers are named
# Bill"  (If there are two different people both named Bill Jones this wouldn't work without
# a unique customer identifier like an SSN to go along with it)

unique_customer_names <- unique(names_frame)  #extract unique customer names from my
#                                              previously-split table
cat("There are",length(unique_customer_names$First.Name),"unique customer names in the table")
```

```
## There are 793 unique customer names in the table
```

```
# subset thelist to just the "Bills"
Unique_Bills <- unique_customer_names[unique_customer_names$First.Name=='Bill',]
Unique_Bills
```

```
##      First.Name Last.Name
## 552        Bill Donatelli
```

```
## 910        Bill    Eplett
## 1081       Bill     Tyler
## 1643       Bill   Stewart
## 2614       Bill  Overfelt
## 4251       Bill   Shonely
```

```
numBills_method3 <- length(Unique_Bills$First.Name) #count them
cat(numBills_method3,"of those customers have the first name 'Bill'")
```

```
## 6 of those customers have the first name 'Bill'
```

Exercise 6

How many mentions of the word 'table' are there in the Product.Name column? **Note you can do this in one line of code**

```
# OK so this function gives me the number of times the string "table" appears, IF that's
# what you wanted
table_mentions1 = sum(stringr::str_count(sales$Product.Name,"table"))
table_mentions1
```

```
## [1] 240
```

```
# This gave us 240 instances of the string "table"  However I see several problems with
# this in a real-world application.  First off, if you're looking for the total sales of
# tables, you don't care whether "table" is capitalized or not.  You want to know how many
# tables you sold.  Since str_count is case sensitive, you would need to adjust for that by
# making everything the same case

table_mentions2 = sum(stringr::str_count(str_to_lower(sales$Product.Name),"table"))
table_mentions2
```

```
## [1] 604
```

```
# That gives us 604 instances of the string "table".  But wait....this includes instances
# where the string "table" isn't a full word, but is embedded in a word like "portable",
# "vegetable" and the like, which may not be what we want.  We may want to look for the WORD
# table.  In this case, we may want to take advantage of R's abilty to include special
# characters in a search string, in this case, using the special character "\\b" which in a
# regexp defines a word boundary.  So we can bound the word "table" with this boundary and
# end up just finding whole word matches for "table", ignoring case, like this:

searchstring <- "\\btable\\b"
table_mentions3 = sum(stringr::str_count(str_to_lower(sales$Product.Name),searchstring))
table_mentions3
```

```
## [1] 230
```

```
# That gives me 230 instances of the WORD table in the product name.  From a business
# perspective, this is probably the closest I can get with this data to answering the
# question "How many tables have we sold".  (A minor flaw in this approach is that if a
# Product.Name contains the word "table" more than once I will over-count. I could solve
# for that too but then I'd be even more off on a tangent.)

# Sorry if this was too long but I feel it's really important to understand what the actual
# business or scientific question is being asked and make sure your code answers that really
# precisely!!
```

Exercise 7

Create a table of counts for each state in the sales data. The counts table should be ordered alphabetically from A to Z.

```
state_count_table <- aggregate(sales$Row.ID, by=list(sales$State), FUN = length)
colnames(state_count_table) <- c("State","Num.Sales")
state_count_table <- state_count_table[order(state_count_table$State,decreasing = FALSE),]
state_count_table  #sort the table
```

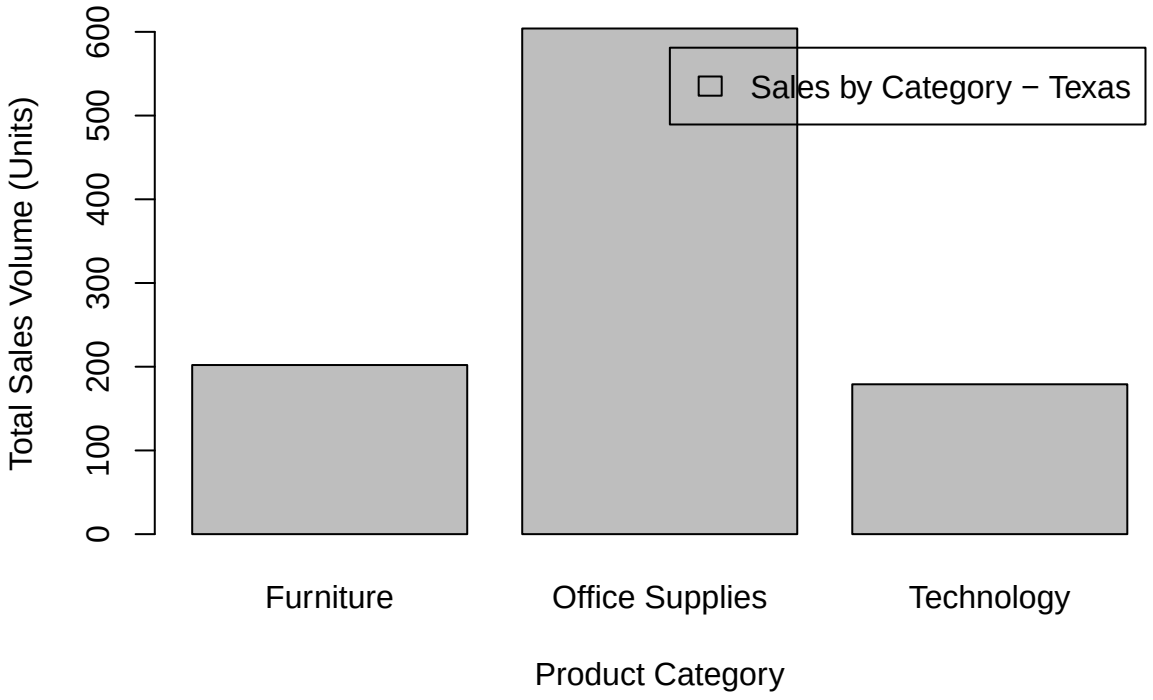
```
##                   State Num.Sales
## 1               Alabama        61
## 2               Arizona       224
## 3              Arkansas        60
## 4            California      2001
## 5              Colorado       182
## 6           Connecticut        82
## 7              Delaware        96
## 8  District of Columbia        10
## 9               Florida       383
## 10              Georgia       184
## 11                Idaho        21
## 12             Illinois       492
## 13              Indiana       149
## 14                 Iowa        30
## 15               Kansas        24
## 16             Kentucky       139
## 17            Louisiana        42
## 18                Maine         8
## 19             Maryland       105
## 20        Massachusetts       135
## 21             Michigan       255
## 22            Minnesota        89
## 23          Mississippi        53
## 24             Missouri        66
## 25              Montana        15
## 26             Nebraska        38
## 27               Nevada        39
## 28        New Hampshire        27
## 29           New Jersey       130
## 30           New Mexico        37
## 31             New York      1128
## 32       North Carolina       249
## 33         North Dakota         7
## 34                 Ohio       469
## 35             Oklahoma        66
## 36               Oregon       124
## 37         Pennsylvania       587
## 38         Rhode Island        56
## 39       South Carolina        42
## 40         South Dakota        12
## 41            Tennessee       183
## 42                Texas       985
## 43                 Utah        53
## 44              Vermont        11
```

```
## 45            Virginia       224
## 46          Washington       506
## 47       West Virginia         4
## 48           Wisconsin       110
## 49             Wyoming         1
```

Exercise 8

Create an alphabetically ordered barplot for each sales Category in the State of Texas.

```
# Since the instructions don't specificy how to aggregate the data to be plotted, I chose
# to simply count the total # of sales in each category, similar to Exercise 7.
texas_sales <- sales[sales$State=='Texas',] #extract Texas sales into new table
texas_sales_by_category <- aggregate(texas_sales$Row.ID,
                                     by=list(texas_sales$Category),
                                     FUN = length) #Aggregate by category
colnames(texas_sales_by_category) <- c("Category","Num.Sales") #assign column names
texas_sales_by_category <- texas_sales_by_category[order(texas_sales_by_category$Category,
                                                         decreasing = FALSE),] #sort
barplot(texas_sales_by_category$Num.Sales,
        names = texas_sales_by_category$Category,
        legend = "Sales by Category - Texas",
        xlab ="Product Category",
        ylab = "Total Sales Volume (Units)") # plot
```



Exercise 9

Find the average profit by region. **Note: You will need to use the aggregate() function to do this. To understand how the function works type ?aggregate in the console.**

```r
# Aggregate profits by region using mean function, and assign column names
profits_by_region <- setNames(aggregate(sales$Profit,
                                        by=list(sales$Region),
                                        FUN = mean),
                              c("Region","Avg.Profit"))

# Sort results descending by Avg.Proft, for better presentation
profits_by_region <- profits_by_region[order(profits_by_region$Avg.Profit,
                                             decreasing = TRUE),]
profits_by_region
```

```
##    Region Avg.Profit
## 4    West   33.84903
## 2    East   32.13581
## 3   South   28.85767
## 1 Central   17.09271
```

Exercise 10

Find the average profit by order year. **Note: You will need to use the aggregate() function to do this. To understand how the function works type ?aggregate in the console.**

```r
#
#
#
#
## Create a new data frame from sales containingRow ID, the Order Date and Profit, adding a
# fourth column containing the sales year, extracted from the order date(assigning column
# names on the fly)
profit_subset <- setNames(data.frame(sales$Row.ID,
                                     sales$Order.Date,
                                     sales$Profit,
                                     lubridate::year(sales$Order.Date)),
                          c("Row.ID","Order.Date","Profit","Order.Year"))
profit_subset[1:8,]
```

```
##   Row.ID Order.Date    Profit Order.Year
## 1      1 2016-11-08   41.9136       2016
## 2      2 2016-11-08  219.5820       2016
## 3      3 2016-06-12    6.8714       2016
## 4      4 2015-10-11 -383.0310       2015
## 5      5 2015-10-11    2.5164       2015
## 6      6 2014-06-09   14.1694       2014
## 7      7 2014-06-09    1.9656       2014
## 8      8 2014-06-09   90.7152       2014
```

```r
# Aggregate profits by region using mean function, and assign column names
profits_by_year <- setNames(aggregate(profit_subset$Profit,
                                      by=list(profit_subset$Order.Year),
                                      FUN = mean),
                            c("Year","Avg.Profit"))

# Sort results by Year
profits_by_year <- profits_by_year[order(profits_by_year$Year,
```

```
                                                decreasing = FALSE),]
profits_by_year
```

```
##   Year Avg.Profit
## 1 2014   24.85899
## 2 2015   29.31427
## 3 2016   31.61777
## 4 2017   28.21234
```