Week 2 Exercises

Michael Durning

March 24, 2023

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

Place this at the top of your script: library(stringr) library(lubridate) library(forcats)

Exercise 1

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

```
# Changing the file encoding in Notepad++ to UTF-8 BOM did not fix.
# Tried ANSI among other encodings did not fix.
# Replaced the broken | unsupported character with M.
# Continued with UTF-8-BOM Encoding as it was mentioned as a fix in class this week.
# class(sales) [1] "data.frame"

# Wouldn't Knit() without this
setwd("C:/Users/Micha/Desktop/Merrimack/R and Python Programming - Class 1/Module 2/")

sales_txt <- read.delim("Data/sales_pipe_encoding_UTF-8_BOM.txt"
                        ,stringsAsFactors=FALSE
                        ,encoding = "UTF-8-BOM"
                        ,sep ="|"
                        )
```

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.

```
library(stringr)
```

```
## Warning: package 'stringr' was built under R version 4.2.3
```

```
# Assigned first element of colnames to a single character via the .txt file
# due to the inablity to load the file without
# Stores the column names as col_names.

col_names <- data.frame(colnames(sales_txt))

# Renames 1st element of the column name to "Row.ID"
colnames(sales_txt)[1] <- "Row.ID"
```

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

```
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 4.2.3
```

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

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(stringr)

# Convert Order.ID into two columns Year and Order.ID | Used in Question 10
order_id_split <- str_split_fixed(sales_txt$Order.ID, pattern = "-", n = 3)

# Create new column called Year
sales_txt$Year <- as.integer(order_id_split[, 2])

# Combine the Order.ID back into one column
sales_txt$Order.ID <- paste(order_id_split[, 1], order_id_split[, 3], sep = "-")

# Change Order.Date vector to date format of YYYY-MM-DD
sales_txt$Order.Date <- mdy(sales_txt$Order.Date)

# Find the most recent order date
newest_order <- max(sales_txt$Order.Date)

# Find the oldest order date
oldest_order <- min(sales_txt$Order.Date)

# Difference of time between Newest and Oldest order dates
# I use difftime on next problem I just wanted to see if there was another way
time_between <- interval(oldest_order, newest_order)


# Assigns the interval between the oldest and newest
# Order dates for specific timeframe
days_between <- time_length(time_between, "days")
weeks_between <- time_length(time_between, "weeks")
years_between <- time_length(time_between, "years")

paste("Most Recent Order:", newest_order)
```

```
## [1] "Most Recent Order: 2017-12-30"
```

```
paste("Oldest Order:", oldest_order)
```

```
## [1] "Oldest Order: 2014-01-03"
```

```r
paste("Days Between Orders:", days_between)
```

```
## [1] "Days Between Orders: 1457"
```

```r
paste("Weeks Between Order:", round(weeks_between, 2))
```

```
## [1] "Weeks Between Order: 208.14"
```

```r
paste("Years Between Orders:", round(years_between, 2))
```

```
## [1] "Years Between Orders: 3.99"
```

Exercise 4

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

```r
library(lubridate)

# Changes format from Ship.Date column from "Month_name DD YYYY" to "YYYY-MM-DD"
sales_txt$Ship.Date <- mdy(sales_txt$Ship.Date)

# Find the days between shipment for each column between Order.Date and Ship.Date
days_between_shipment <- difftime(sales_txt$Ship.Date, sales_txt$Order.Date, units = "days")

# Calculate the mean of the difference between the days_between_shipment
avg_ship_time_days <- mean(as.numeric(days_between_shipment))

paste(round(avg_ship_time_days, 2))
```

```
## [1] "3.91"
```

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.

```r
library(stringr)

# Split the column Customer.Name into two columns separated by " "
customer_name_column_split <- str_split_fixed(sales_txt$Customer.Name, pattern=' ',n=2)

# Rename and replace the split columns into their repsective
# locations before split
colnames(sales_txt)[8:9] <- c("First.Name", "Last.Name")
sales_txt[,8] <- customer_name_column_split[, 1]
sales_txt[,9] <- customer_name_column_split[, 2]

# Find the number of "Bill"'s in First.Name
num_names_bill <- length(grep("Bill", sales_txt$First.Name))
paste("The number of Bill's in First.Name is:", num_names_bill)
```

```
## [1] "The number of Bill's in First.Name is: 37"
```

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**

```
num_name_table <- length(grep("table", sales_txt$Product.Name))
paste("The number of table's in Product.Name is:", num_name_table)
```

```
## [1] "The number of table's in Product.Name is: 197"
```

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.

```
# Create a table of counts for each state and sort by each state
state_levels <- factor(
    sales_txt$State,
    levels = sort(unique(sales_txt$State))
    )

# Create table for each state count
state_counts <- table(state_levels)

# Update the table to now sort alphabetically
sorted_state_counts <- state_counts[sort(names(state_counts))]

# Print the result
print(sorted_state_counts)
```

```
## state_levels
##              Alabama              Arizona             Arkansas
##                   28                  119                   22
##           California             Colorado          Connecticut
##                  993                   90                   50
##             Delaware District of Columbia              Florida
##                   47                    1                  186
##              Georgia                Idaho             Illinois
##                   79                    9                  286
##              Indiana                 Iowa               Kansas
##                   74                   11                   16
##             Kentucky            Louisiana                Maine
##                   64                   18                    4
##             Maryland        Massachusetts             Michigan
##                   63                   71                  142
##            Minnesota          Mississippi             Missouri
##                   41                   27                   37
##              Montana             Nebraska               Nevada
##                    2                   26                   24
##        New Hampshire           New Jersey           New Mexico
##                    9                   58                   11
##             New York       North Carolina         North Dakota
##                  555                  117                    7
##                 Ohio             Oklahoma               Oregon
##                  211                   38                   56
```

```
##         Pennsylvania         Rhode Island       South Carolina
##                  312                   25                   28
##         South Dakota            Tennessee                Texas
##                    9                   88                  460
##                 Utah              Vermont             Virginia
##                   27                   10                   80
##           Washington        West Virginia            Wisconsin
##                  254                    4                   38
##              Wyoming
##                    1
```

Exercise 8

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

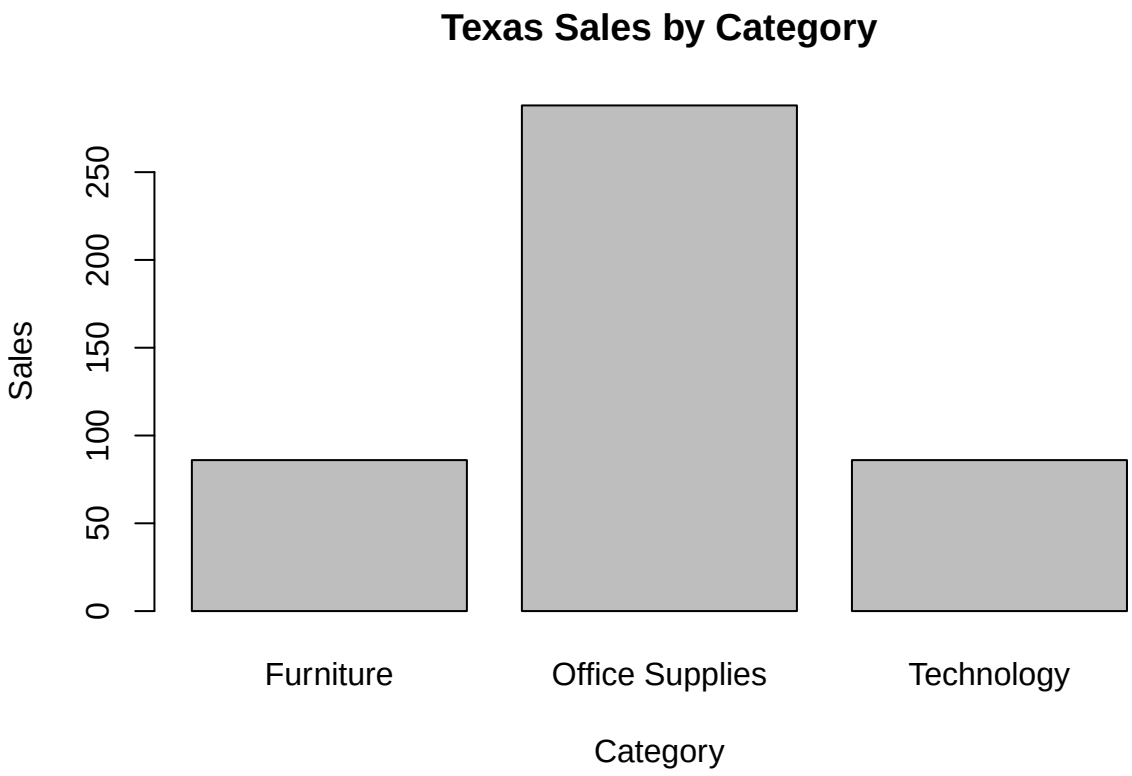
```
# Create a subset for "Texas"
tx_sales <- subset(sales_txt, State == "Texas")

# Create a levels variable for the Category column
category_levels <- factor(
    tx_sales$Category,
    levels = sort(unique(tx_sales$Category))
)

# Create a table for each category count
category_counts <- table(category_levels)

# Sort the table alphabetically
sorted_category_counts <- category_counts[order(names(category_counts))]

# Create an alphabetically ordered barplot
# Faced issue where Y-axis was not going high enough
# Fixed with
barplot(
    sorted_category_counts,
    main = "Texas Sales by Category",
    xlab = "Category",
    ylab = "Sales"
    )
```



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.**

```
# Calculate average profit by region
avg_profit_by_region <- aggregate(
    Profit ~ Region,
    data = sales_txt,
    FUN = mean
    )

print(avg_profit_by_region)
```

```
##    Region   Profit
## 1 Central 20.46822
## 2    East 29.91937
## 3   South 11.27720
## 4    West 32.77000
```

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
# Calculate average profit by order year
avg_profit_by_year <- aggregate(
    Profit ~ Year,
    data = sales_txt,
    FUN = mean
    )

print(avg_profit_by_year)
```

```
##   Year   Profit
## 1 2014 32.24582
## 2 2015 21.58676
## 3 2016 30.10960
## 4 2017 21.31825
```