Assignment\_2

Saipriya Gourineni

2022-10-25

## Here I am loading dplyr package.  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

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

##Now I am going to load the Online\_Retail.csv file  
getwd()

## [1] "C:/Users/Saipr/OneDrive/Desktop"

setwd("C:/Users/Saipr/OneDrive/Desktop/New folder (2)")  
Online\_Retail <- read.csv("Online\_Retail.csv")

# Here I am setting echo= TRUE

1. Show the breakdown of the number of transactions by countries i.e. how many transactions are in the dataset for each country (consider all records including cancelled transactions). Show this in total number and also in percentage. Show only countries accounting for more than 1% of the total transactions. (5 marks)

#I'm currently categorizing the data frame by nation before averaging transactions by % and count. I am eliminating all the nations that account for less than 1% of all transactions.  
  
Online\_Retail %>%   
 group\_by(Country) %>%  
 summarise(n\_transactions = n(), percent\_total = 100\*(n()/nrow(Online\_Retail))) %>%  
 filter(percent\_total > 1.0) %>%   
 arrange(desc(percent\_total))

## # A tibble: 4 × 3  
## Country n\_transactions percent\_total  
## <chr> <int> <dbl>  
## 1 United Kingdom 495478 91.4   
## 2 Germany 9495 1.75  
## 3 France 8557 1.58  
## 4 EIRE 8196 1.51

1. Create a new variable ‘TransactionValue’ that is the product of the exising ‘Quantity’ and ‘UnitPrice’ variables. Add this variable to the dataframe. (5 marks)

#A new column called "TransactionValue" is now being created, and it is being bound to the original dataframe. Here, I'm displaying the top six rows of a new data frame using the head function.  
Online\_Retail <- cbind(Online\_Retail, TransactionValue = Online\_Retail$Quantity \* Online\_Retail$UnitPrice)  
head(Online\_Retail)

## InvoiceNo StockCode Description Quantity  
## 1 536365 85123A WHITE HANGING HEART T-LIGHT HOLDER 6  
## 2 536365 71053 WHITE METAL LANTERN 6  
## 3 536365 84406B CREAM CUPID HEARTS COAT HANGER 8  
## 4 536365 84029G KNITTED UNION FLAG HOT WATER BOTTLE 6  
## 5 536365 84029E RED WOOLLY HOTTIE WHITE HEART. 6  
## 6 536365 22752 SET 7 BABUSHKA NESTING BOXES 2  
## InvoiceDate UnitPrice CustomerID Country TransactionValue  
## 1 12/1/2010 8:26 2.55 17850 United Kingdom 15.30  
## 2 12/1/2010 8:26 3.39 17850 United Kingdom 20.34  
## 3 12/1/2010 8:26 2.75 17850 United Kingdom 22.00  
## 4 12/1/2010 8:26 3.39 17850 United Kingdom 20.34  
## 5 12/1/2010 8:26 3.39 17850 United Kingdom 20.34  
## 6 12/1/2010 8:26 7.65 17850 United Kingdom 15.30

1. Using the newly created variable, TransactionValue, show the breakdown of transaction values by countries i.e. how much money in total has been spent each country. Show this in total sum of transaction values. Show only countries with total transaction exceeding 130,000 British Pound. (10 marks)

#Here, I'm sorting transactions by nation before totaling them using the "TransactionValue" column's value. Later, I will remove any nations with budgets under 130,000 and sort them by decreasing spending.  
Online\_Retail %>%   
 group\_by(Country) %>%  
 summarise(Total\_Spend = sum(TransactionValue)) %>%  
 filter(Total\_Spend > 130000) %>%   
 arrange(desc(Total\_Spend))

## # A tibble: 6 × 2  
## Country Total\_Spend  
## <chr> <dbl>  
## 1 United Kingdom 8187806.  
## 2 Netherlands 284662.  
## 3 EIRE 263277.  
## 4 Germany 221698.  
## 5 France 197404.  
## 6 Australia 137077.

4.This is an optional question which carries additional marks (golden questions). In this question, we are dealing with the InvoiceDate variable. The variable is read as a categorical when you read data from the file. Now we need to explicitly instruct R to interpret this as a Date variable. “POSIXlt” and “POSIXct” are two powerful object classes in R to deal with date and time. Click here for more information. First let’s convert ‘InvoiceDate’ into a POSIXlt object:

Temp=strptime(Online\_Retail$InvoiceDate,format=‘%m/%d/%Y %H:%M’,tz=‘GMT’)

#As I use the head command to check the format, it is creating a temporary variable that formats transaction dates as mm/dd/yyyy.  
  
Temp=strptime(Online\_Retail$InvoiceDate,format='%m/%d/%Y %H:%M',tz='GMT')  
head(Temp)

## [1] "2010-12-01 08:26:00 GMT" "2010-12-01 08:26:00 GMT"  
## [3] "2010-12-01 08:26:00 GMT" "2010-12-01 08:26:00 GMT"  
## [5] "2010-12-01 08:26:00 GMT" "2010-12-01 08:26:00 GMT"

Check the variable using, head(Temp). Now, let’s separate date, day of the week and hour components dataframe with names as New\_Invoice\_Date, Invoice\_Day\_Week and New\_Invoice\_Hour:

Online\_Retail$New\_Invoice\_Date <- as.Date(Temp) # echo=TRUE

# Here, I'm applying a date format to the New Invoice Date column using the Temp variable.  
  
Online\_Retail$New\_Invoice\_Date <- as.Date(Temp)

The Date objects have a lot of flexible functions. For example knowing two date values, the object allows you to know the difference between the two dates in terms of the number days. Try this:

Online\_RetailNew\_Invoice\_Date[10]

#This provides an illustration of how dates can be subtracted from one another to return the value differences.  
  
Online\_Retail$New\_Invoice\_Date[20000]- Online\_Retail$New\_Invoice\_Date[10]

## Time difference of 8 days

Also we can convert dates to days of the week. Let’s define a new variable for that

Online\_RetailNew\_Invoice\_Date)

# Now I'm converting dates to days of the week and giving Invoice Day Week a column label.  
  
Online\_Retail$Invoice\_Day\_Week= weekdays(Online\_Retail$New\_Invoice\_Date)

For the Hour, let’s just take the hour (ignore the minute) and convert into a normal numerical value:

Online\_Retail$New\_Invoice\_Hour = as.numeric(format(Temp, “%H”))

# The transaction hour that is associated with New Invoice Hour is now being added to a new column.  
  
Online\_Retail$New\_Invoice\_Hour = as.numeric(format(Temp, "%H"))

Finally, lets define the month as a separate numeric variable too:

Online\_Retail$New\_Invoice\_Month = as.numeric(format(Temp, “%m”))

#The transaction month that is associated with the New Invoice Hour field is now added to a new column.   
  
Online\_Retail$New\_Invoice\_Month = as.numeric(format(Temp, "%m"))

Now answer the following questions:

1. Show the percentage of transactions (by numbers) by days of the week (extra 2 marks)

# The data frame is now grouped by the day of the week, the percentage of transactions (by number) by day is calculated, and the results are returned in descending order of percentages.  
Online\_Retail %>%  
 group\_by(Invoice\_Day\_Week) %>%  
 summarise(percent\_of\_transactions = 100\*(n()/nrow(Online\_Retail))) %>%  
 arrange(desc(percent\_of\_transactions))

## # A tibble: 6 × 2  
## Invoice\_Day\_Week percent\_of\_transactions  
## <chr> <dbl>  
## 1 Thursday 19.2  
## 2 Tuesday 18.8  
## 3 Monday 17.6  
## 4 Wednesday 17.5  
## 5 Friday 15.2  
## 6 Sunday 11.9

1. Show the percentage of transactions (by transaction volume) by days of the week (extra 1 marks)

#The data frame is now grouped by the day of the week, the percentage of transactions (based on transaction values) is calculated by day, and I'm returning the numbers in decreasing order of percentages.  
Online\_Retail %>%  
 group\_by(Invoice\_Day\_Week) %>%  
 summarise(percent\_of\_transactions\_by\_volume = 100\*(sum(TransactionValue)/sum(Online\_Retail$TransactionValue))) %>%  
 arrange(desc(percent\_of\_transactions\_by\_volume))

## # A tibble: 6 × 2  
## Invoice\_Day\_Week percent\_of\_transactions\_by\_volume  
## <chr> <dbl>  
## 1 Thursday 21.7   
## 2 Tuesday 20.2   
## 3 Wednesday 17.8   
## 4 Monday 16.3   
## 5 Friday 15.8   
## 6 Sunday 8.27

1. Show the percentage of transactions (by transaction volume) by month of the year (extra 1 marks)

#In order to return the numbers in descending order of percentages, I am currently grouping the data frame by the month of the year, computing the percentage of transactions (by transaction values) per month, then grouping the data frame by the month of the year..   
  
Online\_Retail %>%  
 group\_by(New\_Invoice\_Month) %>%  
 summarise(percent\_of\_transactions\_by\_volume = 100\*(sum(TransactionValue)/sum(Online\_Retail$TransactionValue))) %>%  
 arrange(desc(percent\_of\_transactions\_by\_volume))

## # A tibble: 12 × 2  
## New\_Invoice\_Month percent\_of\_transactions\_by\_volume  
## <dbl> <dbl>  
## 1 11 15.0   
## 2 12 12.1   
## 3 10 11.0   
## 4 9 10.5   
## 5 5 7.42  
## 6 6 7.09  
## 7 3 7.01  
## 8 8 7.00  
## 9 7 6.99  
## 10 1 5.74  
## 11 2 5.11  
## 12 4 5.06

1. What was the date with the highest number of transactions from Australia? (3 marks)

# In order to return the highest values for the year, I am now constructing a subset of data for Australian transactions, grouping by the date of the invoice.  
  
subset(Online\_Retail, Country == "Australia") %>%  
 group\_by(New\_Invoice\_Date) %>%  
 summarise(n\_transactions = n()) %>%  
 top\_n(3)

## Selecting by n\_transactions

## # A tibble: 3 × 2  
## New\_Invoice\_Date n\_transactions  
## <date> <int>  
## 1 2011-06-15 139  
## 2 2011-07-19 137  
## 3 2011-08-18 97

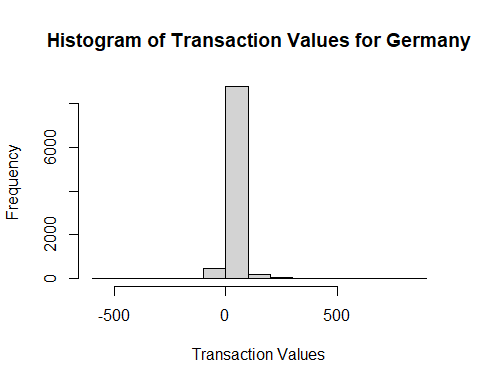
1. The company needs to shut down the website for two consecutive hours for maintenance. What would be the hour of the day to start this so that the distribution is at minimum for the customers? The responsible IT team is available from 7:00 to 20:00 every day(3 marks)

# I'm currently grouping the data frame by transaction hours, averaging the results to produce the transaction percentage by number, and then returning the values in ascending order.  
  
Online\_Retail %>%  
 group\_by(New\_Invoice\_Hour) %>%  
 summarise(percent\_of\_transactions = 100\*(n()/nrow(Online\_Retail))) %>%  
 arrange(percent\_of\_transactions)

## # A tibble: 15 × 2  
## New\_Invoice\_Hour percent\_of\_transactions  
## <dbl> <dbl>  
## 1 6 0.00757  
## 2 7 0.0707   
## 3 20 0.161   
## 4 19 0.684   
## 5 18 1.47   
## 6 8 1.64   
## 7 17 5.26   
## 8 9 6.34   
## 9 10 9.05   
## 10 16 10.1   
## 11 11 10.6   
## 12 14 12.5   
## 13 13 13.3   
## 14 15 14.3   
## 15 12 14.5

1. Plot the histogram of transaction values from Germany. Use the hist() function to plot. (5 marks) # echo=TRUE

# Making a new variable for Germany and visualizing the transaction values on a histogram are the next steps.  
  
Germany\_Transactions <- subset(Online\_Retail, Country == "Germany")  
hist(Germany\_Transactions$TransactionValue, main = "Histogram of Transaction Values for Germany", xlab = "Transaction Values", ylab = "Frequency")



1. Which customer had the highest number of transactions? Which customer is most valuable (i.e. highest total sum of transactions)? (10 marks)

# Here, I'm categorizing the data by client before summarizing it based on count and displaying the top three values in decreasing value.  
Online\_Retail %>%  
 group\_by(CustomerID) %>%  
 summarise(n\_transactions = n()) %>%  
 top\_n(3) %>%  
 arrange(desc(n\_transactions))

## Selecting by n\_transactions

## # A tibble: 3 × 2  
## CustomerID n\_transactions  
## <int> <int>  
## 1 NA 135080  
## 2 17841 7983  
## 3 14911 5903

# Here, I've grouped the data by customers before summarizing it based on transaction values and returning the top three values that are shown in descending value order.  
  
Online\_Retail %>%  
 group\_by(CustomerID) %>%  
 summarise(transaction\_sum = sum(TransactionValue)) %>%  
 top\_n(3) %>%  
 arrange(desc(transaction\_sum))

## Selecting by transaction\_sum

## # A tibble: 3 × 2  
## CustomerID transaction\_sum  
## <int> <dbl>  
## 1 NA 1447682.  
## 2 14646 279489.  
## 3 18102 256438.

1. Calculate the percentage of missing values for each variable in the dataset (5 marks). Hint colMeans():

# Here, I'm computing the proportion of variables in the data frame with missing values.  
  
colMeans(is.na(Online\_Retail))

## InvoiceNo StockCode Description Quantity   
## 0.0000000 0.0000000 0.0000000 0.0000000   
## InvoiceDate UnitPrice CustomerID Country   
## 0.0000000 0.0000000 0.2492669 0.0000000   
## TransactionValue New\_Invoice\_Date Invoice\_Day\_Week New\_Invoice\_Hour   
## 0.0000000 0.0000000 0.0000000 0.0000000   
## New\_Invoice\_Month   
## 0.0000000

8.What are the number of transactions with missing CustomerID records by countries? (10 marks)

# Here, I summarise by total count, group by nation, and filter out numbers that are not NA.  
  
Online\_Retail %>%  
 filter(is.na(Online\_Retail$CustomerID)) %>%  
 group\_by(Country) %>%  
 summarise(n\_missing\_ID = n()) %>%  
 arrange(desc(n\_missing\_ID))

## # A tibble: 9 × 2  
## Country n\_missing\_ID  
## <chr> <int>  
## 1 United Kingdom 133600  
## 2 EIRE 711  
## 3 Hong Kong 288  
## 4 Unspecified 202  
## 5 Switzerland 125  
## 6 France 66  
## 7 Israel 47  
## 8 Portugal 39  
## 9 Bahrain 2

9.On average, how often the costumers comeback to the website for their next shopping? (i.e. what is the average number of days between consecutive shopping) (Optional/Golden question: 18 additional marks!) Hint: 1. A close approximation is also acceptable and you may find diff() function useful.

# Here, I'm deleting "NA" CustomerIDs to create a data frame.  
  
Online\_Retail\_NA\_Removed <- na.omit(Online\_Retail)  
  
# Here, I'm deleting cancelled transactions to create a data frame.  
  
Online\_Retail\_NA\_Neg\_Removed <- subset(Online\_Retail\_NA\_Removed, Quantity > 0)  
  
# I'm making a data frame here that only contains the customer ID and the transaction date.  
  
Online\_Retail\_Subset <- Online\_Retail\_NA\_Neg\_Removed[,c("CustomerID","New\_Invoice\_Date")]  
  
# I'm making a data frame in this instance that eliminates numerous invoices from the same customer on the same day.  
  
Online\_Retail\_Subset\_Distinct <- distinct(Online\_Retail\_Subset)  
  
# Here, I've grouped the data set by CustomerID, arranged it by date, and calculated the typical interval between each customer's consecutive transactions. Later, I'm going to remove the CustomerIDs that produce NA values (i.e., have just one distinct transaction) and add up the data to determine the typical interval between shopping excursions for all CustomerIDs.  
  
Online\_Retail\_Subset\_Distinct %>%  
 group\_by(CustomerID) %>%  
 arrange(New\_Invoice\_Date) %>%  
 summarise(avg = mean(diff(New\_Invoice\_Date))) %>%  
 na.omit() %>%  
 summarise(avg\_days\_between\_shopping = mean(avg))

## # A tibble: 1 × 1  
## avg\_days\_between\_shopping  
## <drtn>   
## 1 78.42025 days

10.In the retail sector, it is very important to understand the return rate of the goods purchased by customers. In this example, we can define this quantity, simply, as the ratio of the number of transactions cancelled (regardless of the transaction value) over the total number of transactions. With this definition, what is the return rate for the French customers? (10 marks). Consider the cancelled transactions as those where the ‘Quantity’ variable has a negative value.

# In order to compute the return rate for France, I am now developing two new subsets that calculate the total number of returns and the total number of transactions.  
  
France\_Transactions\_Cancelled <- subset(Online\_Retail, Country == "France" & Quantity < 0)  
France\_Transactions <- subset(Online\_Retail, Country == "France")  
France\_Return\_Rate <- 100\*(nrow(France\_Transactions\_Cancelled) / nrow(France\_Transactions))  
France\_Return\_Rate

## [1] 1.741264

11.What is the product that has generated the highest revenue for the retailer? (i.e. item with the highest total sum of ‘TransactionValue’)(10 marks)

# I've grouped the data in this case by StockCode and item description, and I've then summarized it using transaction values. and I'm bringing back the values in decreasing order of value.  
  
Online\_Retail %>%  
 group\_by(StockCode, Description) %>%  
 summarise(transaction\_sum = sum(TransactionValue)) %>%  
 arrange(desc(transaction\_sum))

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

## # A tibble: 5,752 × 3  
## # Groups: StockCode [4,070]  
## StockCode Description transaction\_sum  
## <chr> <chr> <dbl>  
## 1 DOT "DOTCOM POSTAGE" 206245.  
## 2 22423 "REGENCY CAKESTAND 3 TIER" 164762.  
## 3 47566 "PARTY BUNTING" 98303.  
## 4 85123A "WHITE HANGING HEART T-LIGHT HOLDER" 97716.  
## 5 85099B "JUMBO BAG RED RETROSPOT" 92356.  
## 6 23084 "RABBIT NIGHT LIGHT" 66757.  
## 7 POST "POSTAGE" 66231.  
## 8 22086 "PAPER CHAIN KIT 50'S CHRISTMAS " 63792.  
## 9 84879 "ASSORTED COLOUR BIRD ORNAMENT" 58960.  
## 10 79321 "CHILLI LIGHTS" 53768.  
## # … with 5,742 more rows

12.How many unique customers are represented in the dataset? You can use unique() and length() functions. (5 marks)

# By eliminating the redundant entries, I am now returning the length of the CustomerID vecto.  
  
length(unique(Online\_Retail$CustomerID))

## [1] 4373