Education

AFM 112 Midterm Prep

Disclosure: This material is for educational purposes only and is intended to supplement course content. Please ensure you review the class materials independently.

Table of Contents

1.0 Load and Read Data **4**
2.0 Basic Commands **4**
3.0 Create New Variables **4**
3.1 Focus on Specific Observations 5
3.1.1 To view the store that generated the largest loss (negative gross profit). 5
4.0 All Code from Chapter 6 **5**
5.0 Summarize() Explanation: **7**
6.0 Mutate() Explanation **8**
6.1 Example of Summary() and Mutate() 8
7.0 Descriptive Statistics **8**
8.0 Determining Outliers **9**
8.1 Code for extreme outliers 9
8.2 To Test for Extreme Outliers 9
8.2.1 For an "IF" statement that automatically performs the test for extreme outliers and returns the outcome. 9
8.2.2 To add a line to include gross profit margin for each store and product type 10
9.0 Aggregate Explanation **10**
10.0 Store and Product Classification Code **10**
11.0 Store Analysis Code **11**
11.1 Use the code below: 11
11.1.1 To find the two best selling days from Store 2 11
12.0 OK Cupid Code Example **12**
12.1 Data Understanding 12
12.2 To find missing values: 12
12.3 Focus on certain data 12
12.3.1 How many users have not reported income (reported income of -1) 12
12.3.2 Create new data set for users that reported income 12
12.3.3 To create a boxplot with results 12
12.3.4 To determine how many outliers 13
12.3.5 Generate summary statistics to understand the income distribution within the outliers: 13
12.3.6 Adding binary variable (reportIncome) 13
13.0 Formula Sheet **14**

1.0 Load and Read Data

- **Control + L** to clear the console.
- "#" to make a note (doesn't run the code).
- **Command + return** to run the line.
- **<-** means "gets".
- Used when assigning values for a variable.
- Use command read_csv to load data.
- **dtL1 <- read_csv("LSL2019_cogs.csv").**
- **dtL1 %>% names()** to view variable names.
- **dtL1 %>% glimpse()** for a detailed understanding of data set.
- <dbl> for variables that are numeric values.
- <chr> for variables that are text based.
- <date> for date-based variables.

2.0 Basic Commands

- "**C**" puts all numbers together into one variable.
- **dtL1 %>% slice_head(n=3)** to show top three observations.
- **dtL1 %>% slice-tail (n=5)** to show bottom five observations.
- **%>%** is "pipe operator".
    - Establishes a sequence of operations.
- **select()** is done to focus on three variables and generate summary/descriptive statistics.
    - **dtL1 %>% select (SalesQuantity, RetailPriceUnit, productCostUnit) %>% summary().**
- To determine how many missing values:
    - **Dt %>% is.na() %>% colSums()**
- To calculate upper whisker for sales quantity:
    - **uw_salesQuantity = quantile(dtL1$SalesQuantity,.75)+1.5*IQR(dtL1$SalesQuantity)**
- To calculate lower whisker for sales quantity:
    - **lw_salesQuantity = quantile(dtL1$SalesQuantity,.25)-1.5*IQR(dtL1$SalesQuantity)**

3.0 Create New Variables

- Use **dtL1 %>% glimpse()** to ensure variables have been added.
- Use **select()** to specify new quantitative variables to generate descriptive statistics.

```
dtL1 %>% select(Revenue, cogs, GrossProfit) %>% summary()
```

3.1 Focus on Specific Observations

- Use command **filter().**
- To look at itemized transaction associated with max order:

dtL2 %>% filter(SalesQuantity == max(SalesQuantity) %>%

select(Store,

SalesDate,

Description,

SalesQuantity,

RetailPriceUnit,

productCostUnit)

3.1.1 To view the store that generated the largest loss (negative gross profit).

- Two conditions required:
  - The first is that the product is Circadia Chardonnay because it had the largest loss.
  - Second is that gross profit is negative.

dtL2 %>% filter(Description == "Circadia Chardonnay" & GrossProfit <0) %>%

select(Store, SalesDate, Description, SalesQuantity, RetailPriceUnit, productCostUnit, Gross Profit)

Shows specific variable, then looks at min:

dtL2 %>% filter (GrossProfit == min(GrossProfit)) %>% select (Store, SalesDate, Description, SalesQuantity, RetailPriceUnit, productCostUnit, Revenue, cogs, GrossProfit)

4.0 All Code from Chapter 6

Set working directory to source file location

Load library

library(tidyverse)

library(lubridate) # needed for extracting weekdays, weeks, months

```
# Load data
    dtL1 <- read_csv("LSL2019_cogs.csv")

# Review/understand data structure
    dtL1 %>% names()
    dtL1 %>% glimpse()
    dtL1 %>% slice_head(n=3)

# Understand data - summary statistics for numeric variables
    dtL1 %>%
    select(SalesQuantity, RetailPriceUnit, productCostUnit) %>%
     summary(uw_salesQuantity = quantile(dtL1$SalesQuantity,.75) +
      1.5*IQR(dtL1$SalesQuantity
    uw_salesQuantity


    lw_salesQuantity = quantile(dtL1$SalesQuantity,.25) - 1.5*IQR(dtL1$SalesQuantity)
    lw_salesQuantity

    dtL1 %>%
      select(productCostUnit) %>%
      summary()

# Data preparation
    dtL1 <- dtL1 %>% mutate(
            ProductPriceUnit=RetailPriceUnit/1.13,
            ProductType=ifelse(ProductPriceUnit>20,"Premium","Regular"),
            Revenue= ProductPriceUnit*SalesQuantity,
            cogs=productCostUnit*SalesQuantity,
```

```
        GrossProfit=Revenue-cogs,
        day=weekdays(SalesDate),
        dayNo=wday(SalesDate)
        )
```

Summary statistics for new variables

- Code below creates a new variable, gross profit margin.

```
dtL1 %>%
        select(Revenue, cogs, GrossProfit) %>%
        summary()

dtL1_ptQ1 <- dtL1 %>%
        mutate(storeGPM=(storeGP/storeRevenue)*100)
        Select(Store, SalesQuantity, Revenue, cogs, GrossProfit) %>%
        group_by (Store) %>%
        summarize()
```

Alternatively, mutate can be used to get a different output.

```
        StoreSalesQ= sum(SalesQuantitiy),
        StoreRevenue= sum(Revenue),
        StoreCOGS= sum(cogs),
        StoreGP= sum(Gross Profit)) %>%
        ungroup()
```

Anytime you group, make sure you ungroup at the bottom.

5.0 Summarize() Explanation:

- Used with **group_by()** to create new aggregate variables for each group.
- Allows for the calculation of the summary (sum) or descriptive statistics (min, max, median, mean, etc) for each group.
- Using **summarize()** without **"group_by"** will generate a summary for the entire data set (The output will have only one line.)

6.0 Mutate() Explanation

- R will generate summary or descriptive statistics for each group and it will add the group level aggregate next to each observation.
    - It will generate as many lines as the original data set.
- R does not treat the results of queries as a new data set, so if we want to have aggregate values across all stores, we need to create a separate query.

6.1 Example of Summary() and Mutate()

```
dtL1 %>%

    select (Store, SalesQuantity, Revenue, cogs, GrossProfit) %>%
    summarize(storeSalesQ= sum(SalesQuantity),

    storeRevenue= sum(Revenue),

    storeCOGS= sum(cogs),

    storeGP= sum(GrossProfit),

    storeGPM=(sum(GrossProfit)/sum(Revenue)*100)) %>%

    ungroup()
```

To include gross profit margin across all stores, add the line below after code **"ungroup()"**

```
mutate(storeGPM=(storeGP/storeRevenue)*100)
```

7.0 Descriptive Statistics

<u>See the code below specifically for the gross profit per transaction for each store:</u>

```
dtL1_ptQ2 <- dtL1 %>%
      select(Store, GrossProfit) %>%
      group_by(Store) %>%
  summarize(minGP = min(GrossProfit), q1GP = quantile(GrossProfit, .25), #formula for
Q1
      maxGP = max(GrossProfit), avgGP = mean(GrossProfit), medianGP=
```

```
median(GrossProfit), q3GP = quantile(GrossProfit, .75), sdGP = sd(GrossProfit),
IQR=IQR(GrossProfit)) %>%
```

8.0 Determining Outliers

- Variable xtrmLW is Q1- 3* IQR
- Variable xtrmUW is Q3+3*IQR

8.1 Code for extreme outliers

```
dtL1_ptQ2 <- dtL1_ptQ2 %>%
    mutate(xtrmLW= q1GP-3 * IQR, xtrmUW = q3GP +3 * IQR)
dtL1_ptQ2
dtL2_ptQ2 <- dtL2_ptQ2 %>%
    mutate(xtrmBelow= ifelse(minGP<xtrmLW, 1, 0),
    xtrmAbove= ifelse(maxGP>xtremUW, 1, 0))
```

8.2 To Test for Extreme Outliers

8.2.1 For an "IF" statement that automatically performs the test for extreme outliers and returns the outcome.

Recall that (1) means true and (0) means false.

```
dtL1_ptQ2 %>%
    mutate(xtrmBelow= ifelse(minGP<xtrmLW, 1,0),
        xtrAbove= ifelse(maxGP>xtrmUW, 1, 0)) %>%
    select(Store, minGP, maxGP, xtrmLW, xtrmUW, xtrmBelow, xtrAbove)
```

8.2.2 To add a line to include gross profit margin for each store and product type

```
dtL1_ptQ3a <- dtL1_ptQ3a %>%
     mutate(storeGPM=(storeGP/storeRevenue)*100)
dtL1_ptQ3a
```

9.0 Aggregate Explanation

To generate the aggregate (sum) for each product type and include the gross profit margin for each product type. Use code below:

```
dtL1 %>%
    select(Store, ProductType, SalesQuantity, Revenue, cogs,
        GrossProfit) %>%
    group_by(ProductType) %>%
    summarize(storeSalesQ=sum(SalesQuantity),
            storeRevenue=sum(Revenue),
            storeCOGS=sum(cogs),
        storeGP=sum(GrossProfit)) %>%
    ungroup() %>%
    mutate(storeGPM=(storeGP/storeRevenue)*100)
```

10.0 Store and Product Classification Code

```
dtL1_ptQ3b <- dtL1 %>%
    select(Store, Classification, SalesQuantity, Revenue, cogs,
    GrossProfit) %>%
 group_by(Store, Classification) %>%
    summarize(storeSalesQ=sum(SalesQuantity),
      storeRevenue=sum(Revenue),
      storeCOGS=sum(cogs),
    storeGP=sum(GrossProfit)) %>%
ungroup() %>%
    mutate(storeGPM=(storeGP/storeRevenue)*100)
dtL1_ptQ3b
```

11.0 Store Analysis Code

To show total units sold on each day of the week for each of the stores, and a second query to show the total units sold on each day of the week for each product type.

11.1 Use the code below:

```
dtL1_ptQ4a <- dtL1 %>%
    select(Store, dayNo, day, SalesQuantity) %>%
```

Use **"dayNo"** to show days of the week in order

```
    group_by(Store, dayNo, day) %>%
        summarize(storeSalesQ=sum(SalesQuantity)) %>%
        ungroup()
dtL1_ptQ4a
```

11.1.1 To find the two best selling days from Store 2

```
dtL1_ptQ4a %>% filter(Store==2) %>% slice_max(storeSalesQ, n=2)

dtL1_ptQ4b <- dtL1 %>%
    select(ProductType, dayNo, day, SalesQuantity) %>%
    group_by(ProductType, dayNo, day) %>%
    summarize(storeSalesQ=sum(SalesQuantity)) %>%
    ungroup()
dtL1_ptQ4b
```

12.0 OK Cupid Code Example

12.1 Data Understanding

```
Library(tidyverse)
options(scipen=99)
```

```
dtL1 %>% glimpse()
```

12.2 To find missing values:

```
dtL1 %>% is.na() %>% colSums()
```

12.3 Focus on certain data

```
dtL1 %>%

select(age, height, income) %>%

summary()
```

12.3.1 How many users have not reported income (reported income of -1)

```
dtL1 %>%

mutate(reportINcome=ifelse(income==1, “No”, “Yes”)) %>%

count(reportIncome) %>% #counts how many observations are in the variable reportIncome

mutate(freq=n/sum(n))
```

12.3.2 Create new data set for users that reported income

```
dtL1a <- dtL1 %>% filter(income!=-1) #filter users who have reported income

dtL1a %>%

select(income %>% summary()
```

12.3.3 To create a boxplot with results

```
boxplot(dtL1a$income, horizontal = TRUE, main= “Income”)
```

12.3.4 To determine how many outliers

1. Create a new variable to capture the end of the upper whisker.
2. Use the variable as a filter and generate count on filtered results.

```
endUW <- quantile(dtL1a$income, .75) + 1.5*IQR(dtL1a$income)

dtL1a %>% filter(income>endUW) %>% select(income) %>% count()
```

12.3.5 Generate summary statistics to understand the income distribution within the outliers:

```
dtL1a %>% filter(income>endUW) %>% select(income) %>% summary()
```

12.3.6 Adding binary variable (reportIncome)

- Shows whether a user has reported their income as part of their profile
- Categorical variable (incomeGroup) that captures all different income groups/categories:

```
dtL1 <- dtL1 %>%
```

For users who did not report income

```
mutate(reportIncome= ifelse(income==-1, "No", "Yes"), incomeGroup=case_when(
income==-1 ~ "IG1_doNotReport"
```

For group of users whose income is below Q1

```
income<= 20000 ~ "IG2_belowMedian"
```

For group of users whose income is between Q1 and Q3 (around the median)

```
income<100000 ~ "IG3_aroundMedian"
```

For group of users whose income is above Q3 but below upper whisker

```
income<220000 ~ "IG4_aboveMedian"
```

For users whose income is above end of upper whisker

```
TRUE ~ "IG5_top1Pct"))
```

13.0 Formula Sheet

Code	Description
library()	Load the package
names()	View variable names
glimpse()	Details of data set

	• Shows data points, variables, names of variables, format, and a brief summary
select()	Focus on a subset of variables
summary()	Generate summary/descriptive statistics
dir()	To see the files in the folder
ifelse()	Equivalent to if() in spreadsheets
weekdays()	Name of the day of the week
wday()	Numeric value representing weekday
read_csv()	Read the data set
slice_head(n=)	Look at the top “n” lines
slice_tail(n=)	Look at the bottom “n” lines
slice_max(n=)	For a given variable, select rows with the highest values
slice_min(n=)	For a given variable, select rows with the lowest values
filter()	Specific observations
is.na()	Examines value of variable and returns TRUE=1 if the value is missing and FALSE=0 if the value is not missing
colSum()	Generates the sum of all values in a column
is.na() %>% colSums()	Answers the question “How many missing values are in these variables?”
arrange()	Arrange
distinct()	Look through the data set and give the unique rows
nrow()	How many rows are present in a data frame
mutate()	Create new variables
summarize()	Generate aggregate variables for groups • Typically used with group_by() • Used to calculated summary or descriptive statistics • If "group_by" is not used, summary statistics for the entire data set are presented (only one line)
group_by()	Grouping by a specific variable
options(scipen=)	Do not present scientific notation in the reporting of results

horizontal=TRUE	Graph shown horizontally
quantile(_____, .25)	First quartile
quantile(_____, .75)	Third quartile
case_when()	Acts as a placeholder for multiple nested if statements
pivot_wider()	Pivot table output
count()	Count how many of the reported the variable in the brackets ● Represented by "n"
condProb()	Probability given a condition
split()	Divides the data into groups defined by variables specified
map(summary)	Command summary is applied to each group
set.seed()	Indicates to return the same sample in randomization
write_csv(input, "output")	Create a copy and make a data set out of it
,	Separates one variable from the other
==	Equality
<-	Define a new variable or new data set
%>%	Pipe ● Sequence of operations/actions
FALSE	0
TRUE	1
|	Used to connect two conditions (e.g. connect 2 conditions in a filter)
!=	"not equal"

- **<dbl>:** For variables that take numeric values.
- **<chr>:** Variables which are text-based.
- **<date>:** Date-based variables.