HW

110078509

202203

### Question 1

data\_df <- read.csv("piccollage\_accounts\_bundles.csv", row.names = 'account\_id' )

1. Let’s explore to see if any sticker bundles seem intuitively similar:
   1. (recommended) Download PicCollage onto your mobile from the App Store and take a look at the style and content of various bundles in their Sticker Store: how many recommendations does each bundle have? (NOTE: the Android app might not have recommendations).
   * Find a single sticker bundle that is both in our limited data set and also in the app’s Sticker Store
   * Ans: (1). Maroon5V is both in our limited data set and also in the app’s Sticker Store
   * (2). it has 6 others recommended bundles in that page.

* ii.Then, use your intuition to recommend five other bundles in our dataset that might have similar usage patterns as this bundle.
  + Ans:
  + My instinct is to count the item based cosine similarity for ‘sweetmothersday’. Implement as below:

sum(data\_df$sweetmothersday)

## [1] 4

4,which means the sum of usage of bundle 'sweetmothersday' is 4 in total among 24649 rows. It's a sparse dataset. Therefore, in next step, I want to remove the empty column from mommy\_day to decrease the computing burden

# Find the index of non-zero values' index number of colname 'sweetmothersday'  
non\_zero\_index <- which(data\_df$sweetmothersday !=0)  
# it show row 3 and 19104 is not zero  
  
#For other bundles, except the zero-usage in both row of '3' & '19104' of others bundles, list them as below  
mommy\_day <-data\_df[non\_zero\_index, ]  
empty\_columns <- sapply(mommy\_day, function(x) all(is.na(x) | x == 0 | x ==""))  
mommy\_day\_valid <- mommy\_day[, !empty\_columns]

Remove the empty column from mommy\_day to decrease the computing burden By during this , I lower the dim from 2*165 to 2*80.

# Cosine-Sim   
cos\_mom <- cosine(as.matrix(mommy\_day\_valid))  
head(sort(cos\_mom[,'sweetmothersday'], decreasing = TRUE), 6)

## sweetmothersday StickerLite wonderland PhotoboothFest   
## 1.0000000 0.9972783 0.9922779 0.9769732   
## WinterWonderland CutieV   
## 0.9701425 0.9647638

In summary of a-ii, the bundle I recommended based on my instinct with the simpify idea of cosine similarity as above.

1. Let’s find similar bundles using geometric models of similarity:
   1. Let’s create cosine similarity based recommendations for all bundles:
   * (1). Create a matrix or data.frame of the top 5 recommendations for "all bundles

# Cosine-Sim   
cos\_all <- cosine(as.matrix(data\_df))  
  
# transfer matrix coss\_all into df then add a added column as "IndexName"  
ac\_bundles\_df <- as.data.frame(cos\_all) %>% rownames\_to\_column("IndexName")   
  
# Build a empty dataframe with size 6\*165  
new\_df <- data.frame(matrix(ncol = dim(ac\_bundles\_df)[2] -1, nrow = 6))  
  
# Using loop for vector based sorting (can not apply sapply() in this case)  
for (i in c(2:dim(ac\_bundles\_df)[2])){  
 new\_df[i-1] <- ac\_bundles\_df %>%   
 arrange( desc(ac\_bundles\_df[i]))%>%  
 slice\_head(n = 6)%>%   
 pull(colnames(ac\_bundles\_df)[1])  
}  
# Get the colnames & rownames for new dataframe  
nx <- colnames(ac\_bundles\_df)  
colnames(new\_df) <- colnames(ac\_bundles\_df)[2:length(nx)]  
rownames(new\_df) <- c('itself', '1st', '2nd', '3rd', '4th', '5th')   
  
  
# Show the first 2 col  
new\_df[,c(1:2)]

## Maroon5V between  
## itself Maroon5V between  
## 1st OddAnatomy BlingStickerPack  
## 2nd beatsmusic xoxo  
## 3rd xoxo gwen  
## 4th alien OddAnatomy  
## 5th word AccessoriesStickerPack

(2). Create a new function that automates the above functionality: it should take an accounts-bundles matrix as a parameter, and return a data object with the top 5 recommendations for each bundle in our data set, using cos-sim.

top5\_cosine\_all<-function(df){  
 cos\_all <- cosine(as.matrix(df));  
 # only receive cosine matrix as argument  
 # transfer matrix coss\_all into df then add a added column as "IndexName"  
 ac\_bundles\_df <- as.data.frame(cos\_all) %>% rownames\_to\_column("IndexName") ;  
   
 # Build a empty dataframe with size 6\*165  
 new\_df <- data.frame(matrix(ncol = dim(ac\_bundles\_df)[2] -1, nrow = 6));  
   
 # Using loop for vector based sorting (can not apply sapply() in this case)  
 for (i in c(2:dim(ac\_bundles\_df)[2])){  
 new\_df[i-1] <- ac\_bundles\_df %>%  
 arrange( desc(ac\_bundles\_df[i]))%>%  
 slice\_head(n = 6)%>%  
 pull(colnames(ac\_bundles\_df)[1])}  
 # Get the colnames & rownames for new dataframe  
 nx <- colnames(ac\_bundles\_df)  
 colnames(new\_df) <- colnames(ac\_bundles\_df)[2:length(nx)]  
 rownames(new\_df) <- c('itself', '1st', '2nd', '3rd', '4th', '5th')  
 return (new\_df)  
}  
  
# Show the first 4 row  
all\_recommendation <- top5\_cosine\_all( df =data\_df)  
all\_recommendation[,c(1:3)]

## Maroon5V between pellington  
## itself Maroon5V between pellington  
## 1st OddAnatomy BlingStickerPack springrose  
## 2nd beatsmusic xoxo X8bit2  
## 3rd xoxo gwen mmlm  
## 4th alien OddAnatomy julyfourth  
## 5th word AccessoriesStickerPack tropicalparadise

(3). What are the top 5 recommendations for the bundle you chose to explore earlier?  
   
 - Ans:   
 I choose all of them in the previous question. List as the answer above above.Therefore, due to my result, my can get any recommendation for the any bundle.  
 However, for better comparison for the following question, I'd choose bundle 'sweetmothersday' as my baseline. And I re-constructed a dummy version of function for a single bundle recommendation.

# Using the result of top5\_cosine\_all for 'sweetmothersday'  
all\_recommendation$sweetmothersday[-1]

## [1] "mmlm" "julyfourth" "tropicalparadise" "bestdaddy"   
## [5] "justmytype"

Moreover, for specific bundle recommendation , here is a simplified function called ‘cosine\_single’

This one is row-based besign , and not powerful than previous one but with more readability.

cosine\_single <- function (name,df , top\_n) {  
 # kindly set data\_df as input, then specific the bundle name as string  
 # output: recommend list according to top\_n  
   
 cos\_df <- df %>% as.matrix() %>% cosine()  
 # get the cosine matrix  
   
 target\_row <- cos\_df[name,]  
 # pick the target row by it's row\_name as a list  
   
 result <- target\_row[order(target\_row, decreasing = TRUE)]  
 # sort it and set it as output  
 return (result[2:(2+top\_n-1)])   
}  
  
cosine\_single( name ="sweetmothersday",df = data\_df, top\_n =5 )

## mmlm julyfourth tropicalparadise bestdaddy   
## 0.9486833 0.9486833 0.9486833 0.9486833   
## justmytype   
## 0.9486833

Both of my code providing the same result as I set bundle as “sweetmothersday”, which prove my functions are valid.

ii. Correlation based recommendations. (minus col-based means)  
  
(1). Reuse the function you created above. (2).But this time give the function an accounts-bundles matrix where each bundle (column) has already been mean-centered in advance.  
  
(3).Now what are the top 5 recommendations for the bundle you chose to explore earlier?

data\_matrix<-as.matrix(data\_df)  
# based on row  
means <- apply(data\_matrix, 2, mean) # length = 165  
# Deduction the mean  
means\_matrix <- t(replicate(nrow(data\_matrix), means));  
col\_normalized\_data\_df <- data\_matrix - means\_matrix  
  
cosine\_single( name ="sweetmothersday",df = col\_normalized\_data\_df, top\_n =5 )

## mmlm julyfourth bestdaddy justmytype gudetama   
## 0.948682 0.948682 0.948682 0.948682 0.948682

iii. Adjusted-cosine based recommendations. (minus row based means)  
  
(1). Reuse the function you created above (you should not have to change it)  
  
(2). But this time give the function an accounts-bundles matrix where each account (row) has already been mean-centered in advance.  
  
(3). What are the top 5 recommendations for the bundle you chose to explore earlier?

data\_matrix<-as.matrix(data\_df)  
# based on row  
means <- apply(data\_matrix, 1, mean) # length = 24649  
# Deduction the mean  
bundle\_means\_matrix <- replicate(ncol(data\_matrix), means);  
  
row\_normalized\_data\_df <- data\_matrix - bundle\_means\_matrix  
  
cosine\_single( name ="sweetmothersday",df = row\_normalized\_data\_df, top\_n =5 )

## justmytype julyfourth gudetama mmlm bestdaddy   
## 0.9984446 0.9984391 0.9984391 0.9961341 0.9961341

#### (not graded) Are the three sets of geometric recommendations similar in nature (theme/keywords) to the recommendations you picked earlier using your intuition alone?

- Ans:No.

#### What reasons might explain why your computational geometric recommendation models produce different results from your intuition?

- Ans:  
  
The reason is that , in beginning, I only take the nominator part of cosine-sim into my consideration without the part of the denominator part of the function. Moverover, I did not apply any normalization to my data.

(not graded) What do you think is the conceptual difference in cosine similarity, correlation, and adjusted-cosine?

### Question 2

interactive\_regression <- function() {  
 cat("Click on the plot to create data points; hit [esc] to stop")  
 plot(NA, xlim=c(-5,50), ylim=c(-5,50))  
 points = data.frame()  
 repeat {  
 click\_loc <- locator(1)  
 if (is.null(click\_loc)) break  
 if(nrow(points) == 0 ) {  
 points <- data.frame(x=click\_loc$x, y=click\_loc$y)  
 } else {  
 points <- rbind(points, c(click\_loc$x, click\_loc$y))  
 }  
 plot(points, xlim=c(-5,50), ylim=c(-5,50), pch=19, cex=2, col="gray")  
 if (nrow(points) < 2) next  
   
 model <- lm(points$y ~ points$x)  
 abline(model, lwd=2, col="cornflowerblue")  
 text(1, 50, paste(c("Raw intercept: ", round(model$coefficients[1], 2)), collapse=" "))  
 text(1, 45, paste(c("Raw slope : ", round(model$coefficients[2], 2)), collapse=" "))  
 text(1, 40, paste(c("Correlation : ", round(cor(points$x, points$y), 2)), collapse=" "))  
 }  
   
 return(points)  
}

1. Create a horizontal set of random points, with a relatively narrow but flat distribution.
2. What raw slope of x and y would you generally expect?

- \*Ans:\* slope: 0

1. What is the correlation of x and y that you would generally expect?

- \*Ans:\* correlation: 0

1. Create a completely random set of points to fill the entire plotting area, along both x-axis and y-axis
2. What raw slope of the x and y would you generally expect?

* *Ans:* slope: 0

1. What is the correlation of x and y that you would generally expect?

* *Ans:* correlation: 0

1. Create a diagonal set of random points trending upwards at 45 degrees
2. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)

* *Ans:* slope: 1

1. What is the correlation of x and y that you would generally expect?

* *Ans:* correlation: 1

1. Create a diagonal set of random trending downwards at 45 degrees
2. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)

* *Ans:* slope: -1

1. What is the correlation of x and y that you would generally expect?

* *Ans:*correlation: -1

1. Apart from any of the above scenarios, find another pattern of data points with no correlation (r ≈ 0). (can create a pattern that visually suggests a strong relationship but produces r ≈ 0?)

Done.

No, we can not create a pattern that visually suggests a strong relationship but produces r ≈ 0.

|  |
| --- |
| f. Apart from any of the above scenarios, find another pattern of data points with perfect correlation (r ≈ 1). (can you find a scenario where the pattern visually suggests a different relationship?) |
| Done. If it’s linear, it’s 1. |

1. Let’s see how correlation relates to simple regression, by simulating any linear relationship you wish:
2. Run the simulation and record the points you create: pts <- interactive\_regression() (simulate either a positive or negative relationship)

# pts <- interactive\_regression()

1. Use the lm() function to estimate the regression intercept and slope of pts to ensure they are the same as the values reported in the simulation plot: summary( lm( ptsx ))

# summary( lm( pts$y ~ pts$x ))

1. Estimate the correlation of x and y to see it is the same as reported in the plot: cor(pts)

# cor(pts)

1. Now, standardize the values of both x and y from pts and re-estimate the regression slope

# temp =pts  
# temp$x <-( pts$x - mean\_list[1])/sd(pts$x)  
# temp$y <- (pts$y - mean\_list[2])/sd(pts$y)  
# temp

# lmTemp = lm(x~y, data = temp) #Create the linear regression  
# # summary(lmTemp)  
# slope <- round(lmTemp$coefficients[2], 2)  
# sprintf('slope: %f' , slope)

Ans: The slope is 0.75.

1. What is the relationship between correlation and the standardized simple-regression estimates?

# correlation<- round(cor(temp$x, temp$y), 2)  
#   
# sprintf('correlation: %f' , correlation)  
# sprintf('slope: %f' , slope)

*Ans:* They’re the same.