HW10

106071035

4/29/2021

Question 1) Let's make an automated recommendation system for the PicCollage mobile app.

```
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.6.2
```

```
ac_bundles_dt <- fread("piccollage_accounts_bundles.csv")
ac_bundles_matrix <- as.matrix(ac_bundles_dt[, -1, with=FALSE])
```

a. Let's explore to see if any sticker bundles seem intuitively similar:

i. (recommended) Download PicCollage onto your mobile from the iOS/Android 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? –>31 or more buddles. In each buddle, there are approximately 24 recommended stickers.

ii. Find a single sticker bundle that is both in our limited data set and also in the app's Sticker Store (e.g., "sweetmothersday"). Then, use your intuition to recommend (guess!) five other bundles in our dataset that might have similar usage patterns as this bundle. –> e.g. I Love Mom, Amazing Mom, Mama and me, To Mom with Lovw, Mommy My Queen! #### b. Let's find similar bundles using geometric models of similarity:

iii. 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

```
library(lsa)
```

```
## Warning: package 'lsa' was built under R version 3.6.2
```

```
## Loading required package: SnowballC
```

```
## Warning: package 'SnowballC' was built under R version 3.6.2
```

```
M <-cosine(ac_bundles_matrix)
Recommendation_Matrix<-matrix(rep(sqrt(length(row(M)))*5), nrow = 165, ncol = 5)
for(i in c(1:sqrt(length(row(M))))){
  Recommendation_Matrix[i,]<-names(sort(M[i,],decreasing=TRUE)[2:6])
  #listt<-c(names(sort(M[i,],decreasing=TRUE)[2:6]))
}
rownames(Recommendation_Matrix) <- c(rownames(M))
colnames(Recommendation_Matrix) <- c("1st","2nd","3rd","4th","5th")
head(Recommendation_Matrix)
```

```
##                 1st                 2nd                 3rd
## Maroon5V        "OddAnatomy"        "beatsmusic"        "xoxo"
## between         "BlingStickerPack"  "xoxo"              "gwen"
## pellington      "springrose"        "8bit2"             "mmlm"
## StickerLite     "HeartStickerPack"  "HipsterChicSara"   "Mom2013"
## saintvalentine  "nashnext"          "givethanks"        "teenwitch"
## HipsterChicSara "Random"            "HeartStickerPack"  "wonderland"
##                 4th              5th
## Maroon5V        "alien"          "word"
```

```
## between         "OddAnatomy"     "AccessoriesStickerPack"
## pellington      "julyfourth"     "tropicalparadise"
## StickerLite     "Emome"          "Random"
## saintvalentine  "togetherwerise" "lovestinks2016"
## HipsterChicSara "Emome"          "StickerLite"

# we show first 6 buddles' top5 recommendations.
```

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 cosine similarity.

```
Recommendation_Top5 <- function(Matrix){
  Recommendation_Matrix_f<-matrix(rep(sqrt(length(row(Matrix)))*5), nrow = 165, ncol = 5)
  for(i in c(1:sqrt(length(row(Matrix))))){
    Recommendation_Matrix_f[i,]<-names(sort(Matrix[i,],decreasing=TRUE)[2:6])
  }
  rownames(Recommendation_Matrix_f) <- c(rownames(Matrix))
  colnames(Recommendation_Matrix_f) <- c("1st","2nd","3rd","4th","5th")
  return(Recommendation_Matrix_f)
}
```

3. What are the top 5 recommendations for the bundle you chose to explore earlier?

```
Recommendation_Top5(M)["sweetmothersday",]

##                1st                2nd                3rd                4th
##             "mmlm"       "julyfourth" "tropicalparadise"        "bestdaddy"
##                5th
##       "justmytype"
```

ii. Let's create correlation based recommendations.

1. Reuse the function you created above (don't change it; don't use the cor() function)
2. But this time give the function an accounts-bundles matrix where each bundle (column) has already been mean-centered in advance.

```
bundle_means <- apply(ac_bundles_matrix, 1, mean)
bundle_means_matrix <- replicate(ncol(ac_bundles_matrix), bundle_means)
ac_bundles_mc_b <- ac_bundles_matrix - bundle_means_matrix
cor_sim <- cosine(ac_bundles_mc_b)
```

3. Now what are the top 5 recommendations for the bundle you chose to explore earlier?

```
Recommendation_Top5(cor_sim)["sweetmothersday",]

##          1st          2nd          3rd          4th          5th
## "justmytype" "julyfourth"   "gudetama"       "mmlm"  "bestdaddy"
```

iii. Let's create adjusted-cosine based recommendations.

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.

```
bundle_means_r <- apply(ac_bundles_matrix, 2, mean)
bundle_means_matrix_r <- t(replicate(nrow(ac_bundles_matrix), bundle_means_r))
ac_bundles_mc_b_r <- ac_bundles_matrix - bundle_means_matrix_r
cor_sim_r <- cosine(ac_bundles_mc_b_r)
```

3. What are the top 5 recommendations for the bundle you chose to explore earlier?

```
Recommendation_Top5(cor_sim_r)["sweetmothersday",]
```

```
##          1st           2nd           3rd           4th           5th
##       "mmlm" "julyfourth"   "bestdaddy" "justmytype"    "gudetama"
```

c. (not graded) Are the three sets of geometric recommendations similar in nature (theme/keywords) to the recommendations you picked earlier using your intuition alone? What reasons might explain why your computational geometric recommendation models produce different results from your intuition?

geometric recommendations： care about the keyword #### d. (not graded) What do you think is the conceptual difference in cosine similarity, correlation, and adjusted-cosine? correlation recommendations: care about the "popular similarity", it gathers the "account preference" and make the similar preference recommendations. adjusted-cosine: adjusted the number –> like normalize each buddle to make it comparable.

Question 2) Correlation is at the heart of many data analytic methods so let's explore it further.

a. Create a horizontal set of random points, with a relatively narrow but flat distribution.

i. What raw slope of x and y would you generally expect?
–>0

ii. What is the correlation of x and y that you would generally expect?
–>0

b. Create a completely random set of points to fill the entire plotting area, along both x-axis and y-axis

i. What raw slope of the x and y would you generally expect?
–>0

ii. What is the correlation of x and y that you would generally expect?
–>0

c. Create a diagonal set of random points trending upwards at 45 degrees

i. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)
–>1

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

d. Create a diagonal set of random trending downwards at 45 degrees

i. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)
–>-1

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

e. Apart from any of the above scenarios, find another pattern of data points with no correlation (r ≈ 0).

(optionally: can create a pattern that visually suggests a strong relationship but produces r ≈ 0?)
–>the random points gather like a circle.

f. Apart from any of the above scenarios, find another pattern of data points with perfect correlation (r ≈ 1).

(optionally: can you find a scenario where the pattern visually suggests a different relationship?)
no–> the relationship is 1-1. (The Pearson correlation assumes it's linear)

g. Let's see how correlation relates to simple regression, by simulating any linear relationship you wish:

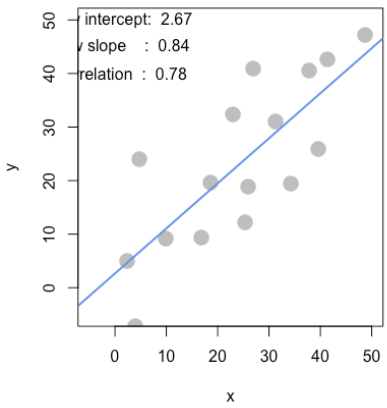
i. Run the simulation and record the points you create: pts <- interactive_regression()

```
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)
}
```

We do the" pts <- interactive_regression() "on the other file.
Becuase this interactive function would cause a problem when the markdown file knitting, in the following section I manually input the"clicking points". Here shown the clicking simulation outcome.



ii. 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:

```
ptsy <- c(9.169420 ,19.628506 ,24.002305 ,40.927007, 31.038417 ,25.903957 ,12.212063 ,32.
369574 ,42.638494, 47.202459 , 4.985786, -7.184787, 9.359585 ,18.867845, 19.438340, 40.54
6677)
```

```
ptsx <- c(9.885718 ,18.598880  ,4.737032 ,26.915989 ,31.272570 ,39.589679 ,25.331778 ,22.
955461, 41.371916 ,48.698893 , 2.360715  ,3.944926, 16.816642 ,25.925857, 34.242966, 37.8
07441)
pts <- rbind(x=ptsx,y=ptsy)
summary( lm( ptsy ~ ptsx ))

##
## Call:
## lm(formula = ptsy ~ ptsx)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1714  -8.1057   0.8241   5.3990  17.3495
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.6686     5.0554   0.528 0.605859
## ptsx          0.8411     0.1804   4.663 0.000366 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.942 on 14 degrees of freedom
## Multiple R-squared:  0.6083, Adjusted R-squared:  0.5803
## F-statistic: 21.74 on 1 and 14 DF,  p-value: 0.0003663
```

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

```
cor(ptsx,ptsy)

## [1] 0.7799185
```

iv. Now, re-estimate the regression using standardized values of both x and y from pts

```
xx<-c((ptsx-mean(ptsx))/sd(ptsx))
yy<-c(((ptsy-mean(ptsy))/sd(ptsy)))
st_regression <- lm(yy ~ xx)
```

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

```
summary(st_regression)

##
## Call:
## lm(formula = yy ~ xx)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8583 -0.5282  0.0537  0.3518  1.1306
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.327e-17  1.620e-01   0.000 1.000000
## xx           7.799e-01  1.673e-01   4.663 0.000366 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6478 on 14 degrees of freedom
## Multiple R-squared:  0.6083, Adjusted R-squared:  0.5803
## F-statistic: 21.74 on 1 and 14 DF,  p-value: 0.0003663
```