BACS HW (Week10)

106070038

2021-05-02

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

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

i. How many recommendations does each bundle have?

- Answer: **six**(iOS version)

ii. Use your intuition to recommend five other bundles in our dataset that might have similar usage patterns as this bundle.

- Answer: **HeartStickerPack** -> Similar usage patterns: supersweet, fallinlovewiththefall, hellobaby, valentineStickers, warmncozy(by intuition) Because all of the stickers above are related to "love"

```
# install.packages("data.table")
library(data.table)
# setwd("/Users/weiwei/Desktop/2021Spring_Courses/BACS/HW8")
ac_bundles_dt <- fread("piccollage_accounts_bundles.csv")
ac_bundles_matrix <- as.matrix(ac_bundles_dt[, -1, with=FALSE])
```

b. Let's find similar bundles using geometric models of similarity

i. 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

```
# install.packages("lsa")
# install.packages("SnowballC")
library(SnowballC)
library(lsa)
cos_sim <- cosine(ac_bundles_matrix)
## apply(,1,): by row
cos_sim_add <- apply(cos_sim, 1, mean)
cos_sim_add_rank <- cos_sim_add[order(cos_sim_add, decreasing = TRUE)]
cos_sim_add_rank[1:5]
```

```
##     springrose eastersurprise         bemine     watercolor hipsterholiday
##      0.1578966      0.1459645      0.1383451      0.1375165      0.1368757
```

- Answer: springrose, eastersurprise, bemine, watercolor, hipsterholiday

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.

```
get_top5 <- function (bundle_name,data) {
  reg1 <- data[bundle_name,]
  reg2 <- reg1[order(reg1, decreasing = TRUE)]
  return (reg2[2:6]) ## top1-5, exclude itself(cos_sim==1)
}
```

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

```
get_top5("HeartStickerPack",cos_sim)
```

```
##      StickerLite           Emome WordsStickerPack  HipsterChicSara
##        0.4256352        0.3870007        0.3834636        0.3292921
## BlingStickerPack
##        0.3181781
```

- Answer: **HeartStickerPack** -> Similar usage patterns: StickerLite, Emome, WordsStickerPack, HipsterChicSara, BlingStickerPack(by caculation) Totally not same as what I guess in a-ii.

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

```
bundle_means <- apply(ac_bundles_matrix, 2, mean)
bundle_means_matrix <- t(replicate(nrow(ac_bundles_matrix), bundle_means))
ac_bundles_mc_b <- ac_bundles_matrix - bundle_means_matrix
row.names(ac_bundles_mc_b) <- row.names(ac_bundles_dt)
cor_sim_2 <- cosine(ac_bundles_mc_b)
get_top5("HeartStickerPack", cor_sim_2)
```

```
##      StickerLite WordsStickerPack            Emome BlingStickerPack
##        0.3870573        0.3832913        0.3714926        0.3203997
##   HipsterChicSara
##        0.3071336
```

```
class(ac_bundles_mc_b)
```

```
## [1] "matrix" "array"
```

- Answer: **HeartStickerPack** -> Similar usage patterns: StickerLite, WordsStickerPack, Emome, BlingStickerPack, HipsterChicSara The results are same as (b), however, the order is not same.

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

```
#install.packages("data.table")
library(data.table)
library(lsa)
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_3 <- cosine(ac_bundles_mc_b)
get_top5("HeartStickerPack", cor_sim_3)
```

```
##      StickerLite            Emome BlingStickerPack  HipsterChicSara
##        0.3974934        0.3311735        0.2865498        0.2726981
## WordsStickerPack
##        0.2594191
```

- Answer: **HeartStickerPack** -> Similar usage patterns: StickerLite, Emome, BlingStickerPack, HipsterChicSara, WordsStickerPack The results are same as (b), however, the order is not same.

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?

- Answer: No, because I just guess it by literal meaning, but we should calculate the similarity.

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

- Answer:
  - **Cosine similarity** defined as the angular similarity between two vectors. angle 0 defines match and 90 otherwise.
  - **correlation coefficient** defined as the covariance between two vectors divided by their standard deviations.
  - **adjusted-cosine** is very much similar to pearson similarity except if two are calculated over different set of rated vectors.

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

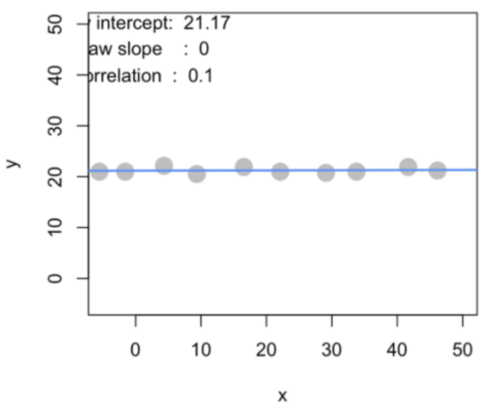
a. Create a horizontal set of random points, with a relatively narrow but flat distribution.

```
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)
}
interactive_regression()
```

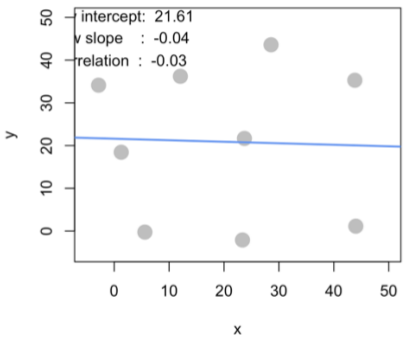
i. What raw slope of x and y would you generally expect?

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



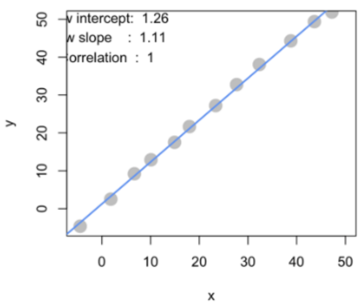
- Answer:
    - Expected Raw slope = 0
    - Expected correlation = 0

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



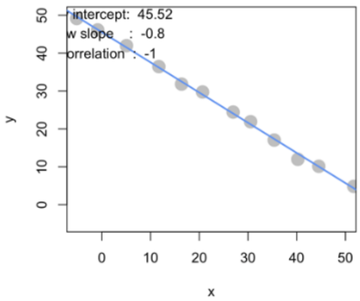
i. What raw slope of the x and y would you generally expect?
ii. What is the correlation of x and y that you would generally expect?
- Answer:
    - Expected Raw slope ≈ 0
    - Expected correlation ≈ 0

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)
ii. What is the correlation of x and y that you would generally expect?
- Answer:
    - Expected Raw slope ≈ 1
    - Expected correlation ≈ 1

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



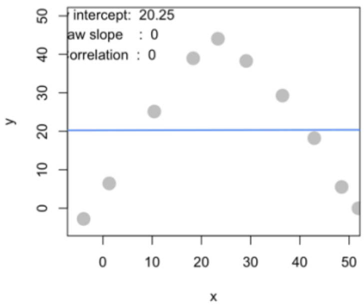
i. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)

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

- Answer:
    - Expected Raw slope = -1
    - Expected correlation = -1

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

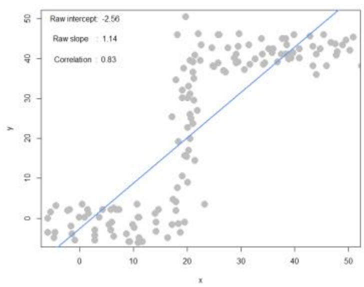
(optionally: can create a pattern that visually suggests a strong relationship but produces r ≈ 0?)



- Answer:
    - Expected correlation = 0

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?)



- Answer:
    - Expected correlation = 1

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()

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

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

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

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

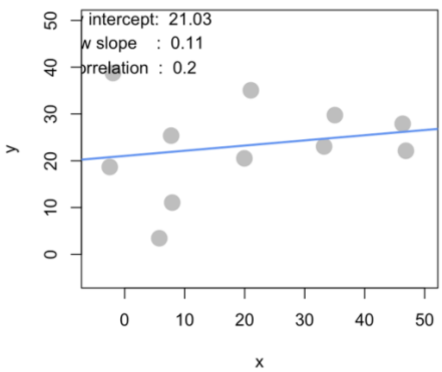
```
pts <- interactive_regression()
pts <- as.data.frame(pts)

linear_model <- lm(y~x, data=pts)
summary(linear_model)

cor(pts)
cor.test(pts$x,pts$y,method="pearson")

x<-(pts[,1]-mean(pts[,1]))/sd(pts[,1])
y<-(pts[,2]-mean(pts[,2]))/sd(pts[,2])
sta<-cbind(x,y)
sta<-as.data.frame(sta)
lm_sta<-lm(y~x, data=sta)
summary(lm_sta)

cor(sta)
cor(pts)
```




```
Call:
lm(formula = y ~ x, data = pts)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.237  -3.411  -1.660   4.153  17.931 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  21.0326     4.7944   4.387  0.00175 **
x             0.1108     0.1814   0.611  0.55658   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.43 on 9 degrees of freedom
Multiple R-squared:  0.03978,	Adjusted R-squared:  -0.06692 
F-statistic: 0.3728 on 1 and 9 DF,  p-value: 0.5566
```

```
> cor(pts)
          x         y
x 1.0000000 0.1994381
y 0.1994381 1.0000000
> cor.test(pts$x,pts$y,method="pearson")

	Pearson's product-moment correlation

data:  pts$x and pts$y
t = 0.61058, df = 9, p-value = 0.5566
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4548548  0.7139032
sample estimates:
      cor
0.1994381
```

```
> summary(lm_sta)

Call:
lm(formula = y ~ x, data = sta)

Residuals:
    Min      1Q  Median      3Q     Max
-1.8065 -0.3379 -0.1645  0.4114  1.7763

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.008e-16  3.114e-01   0.000    1.000
x           1.994e-01  3.266e-01   0.611    0.557

Residual standard error: 1.033 on 9 degrees of freedom
Multiple R-squared:  0.03978,	Adjusted R-squared:  -0.06692
F-statistic: 0.3728 on 1 and 9 DF,  p-value: 0.5566
```

```
> cor(sta)
          x         y
x 1.0000000 0.1994381
y 0.1994381 1.0000000
> cor(pts)
          x         y
x 1.0000000 0.1994381
y 0.1994381 1.0000000
```

- Answer: After the standardization, the correlation will not change.