BACS HW10

109062710

4/28/2021

```
# table <- fread("/users/bijonsetyawan/Documents/Git/bacs-hw/hw10/piccollage_accounts_bundles.csv") # macos
table <- fread("/home/johnbjohn/Documents/git_repos/bacs-hw/hw10/piccollage_accounts_bundles.csv")
# linux
matrix  <- as.matrix(table[, -1, with=FALSE])
```

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. (recommended) Download PicCollage onto your mobile from the iOS/Android app store and a look at the style and content of various bundles in their Sticker Store: how many recommendations does each bundle have?

For example for `supercute` , there are 31 recommendations of 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!) 5 other bundles in our data set that might have similar usage patterns as this bundle.

For this case, I will choose `xmasquotes` . Based on my estimation, `xmasquotes` will be related to: 1. `WinterWonderland` 2. `Xmas2012StickerPack` 3. `snowflakes` 4. `chicchristmas` 5. `snowflakeee`

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

```
cosine_similarity_matrix <- cosine(matrix)
diag(cosine_similarity_matrix) <- 100
recommendation_df <- data.frame(stringsAsFactors = F)
for(bundle in row.names(cosine_similarity_matrix)) {
  recommendation_df <-
    rbind(
      recommendation_df,
      names(
        cosine_similarity_matrix[
          bundle,
          order(cosine_similarity_matrix[bundle, ], decreasing = T)
        ]
      )[1:6],
      stringsAsFactors = F
    )
}
rownames(recommendation_df) <- row.names(cosine_similarity_matrix)
recommendation_df <- recommendation_df[, -1]
colnames(recommendation_df) <- c("First", "Second", "Third", "Fourth", "Fifth")

head(recommendation_df, 5)
```

```
##                           First          Second     Third         Fourth
## Maroon5V             OddAnatomy      beatsmusic      xoxo          alien
## between        BlingStickerPack            xoxo      gwen     OddAnatomy
## pellington           springrose           8bit2      mmlm     julyfourth
## StickerLite    HeartStickerPack HipsterChicSara   Mom2013          Emome
## saintvalentine         nashnext      givethanks teenwitch togetherwerise
##                             Fifth
## Maroon5V                     word
## between        AccessoriesStickerPack
## pellington           tropicalparadise
## StickerLite                    Random
## saintvalentine         lovestinks2016
```

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.

```
recommender <- function(matrix) {
  cosine_similarity_matrix <- cosine(matrix)
  diag(cosine_similarity_matrix) <- 100
  recommendation_df <- data.frame(stringsAsFactors = FALSE)
  for(bundle in row.names(cosine_similarity_matrix)) {
    recommendation_df <-
    rbind(
      recommendation_df,
      names(
        cosine_similarity_matrix[
          bundle,
          order(cosine_similarity_matrix[bundle, ], decreasing = T)
        ]
      )[1:6],
      stringsAsFactors = F
    )
  }
  rownames(recommendation_df) <- row.names(cosine_similarity_matrix)
  recommendation_df <- recommendation_df[, -1]
  colnames(recommendation_df) <-
    c("First", "Second", "Third", "Fourth", "Fifth")

  return(recommendation_df)
}

recommendation <- recommender(matrix)
head(recommendation, 5)
```

```
##                           First          Second     Third         Fourth
## Maroon5V             OddAnatomy      beatsmusic      xoxo          alien
## between        BlingStickerPack            xoxo      gwen     OddAnatomy
## pellington           springrose           8bit2      mmlm     julyfourth
## StickerLite    HeartStickerPack HipsterChicSara   Mom2013          Emome
## saintvalentine         nashnext      givethanks teenwitch togetherwerise
##                                Fifth
## Maroon5V                        word
## between        AccessoriesStickerPack
## pellington           tropicalparadise
## StickerLite                    Random
## saintvalentine         lovestinks2016
```

II. Let's create correlation based recommendations.

1. Reuse the function you created above (do not change it; do not use the cor() function)

```
means <- apply(matrix, 2, mean)
col_mean_matrix <- t(replicate(nrow(matrix), means))
col_mean_centered_matrix <- matrix - col_mean_matrix
```

2. But this time give the function an accounts-bundles matrix where each bundle (column) has already been mean-centered in advance.

```
col_mean_centered_recommendation <- recommender(col_mean_centered_matrix)
head(col_mean_centered_recommendation, 5)
```

```
##                           First                  Second            Third
## Maroon5V             OddAnatomy              beatsmusic             xoxo
## between        BlingStickerPack                    xoxo             gwen
## pellington           springrose                  8bit2 tropicalparadise
## StickerLite    HeartStickerPack AnimalFriendsStickerPack          between
## saintvalentine         nashnext              givethanks        teenwitch
##                          Fourth                   Fifth
## Maroon5V                  alien                    word
## between             OddAnatomy AccessoriesStickerPack
## pellington                mmlm             julyfourth
## StickerLite              Emome        HipsterChicSara
## saintvalentine togetherwerise         lovestinks2016
```

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

```
col_mean_centered_recommendation["xmasquotes",]
```

```
##             First     Second       Third        Fourth     Fifth
## xmasquotes wpbear 4thofjuly3 snowflakeee newyearsparty warmncozy
```

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

1. Reuse the function you created above (you should not have to change it)

```
means <- apply(matrix, 1, mean)
row_mean_matrix <- replicate(ncol(matrix), means)
row_mean_centered_matrix <- matrix - row_mean_matrix
```

2. But this time give the function an accounts-bundles matrix where each account (row) has already been mean-centered in advance.

```
row_mean_centered_recommendation <- recommender(row_mean_centered_matrix)
head(row_mean_centered_recommendation, 5)
```

```
##                           First     Second           Third           Fourth
## Maroon5V             OddAnatomy       word            xoxo       beatsmusic
## between        BlingStickerPack       xoxo            gwen      Monsterhigh
## pellington           springrose      8bit2      backtocool tropicalparadise
## StickerLite    HeartStickerPack    Mom2013 HipsterChicSara            Emome
## saintvalentine   togetherwerise givethanks       teenwitch      mrcurlsport
##                         Fifth
## Maroon5V             supercute
## between             OddAnatomy
## pellington          julyfourth
## StickerLite             Random
## saintvalentine          arrows
```

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

```
row_mean_centered_recommendation["xmasquotes",]
```

```
##             First        Second   Third     Fourth         Fifth
## xmasquotes wpbear christmassnow cny2017 frombierun floralwedding
```

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.

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

Close to 0.

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

Since it's only shows a straight line, meaning that there is no correlation between x and y. Thus, the correlation value is always close to 0.

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

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

It's similar to scenario (a) where all data points are scattered across the whole place, thus they offset each other. CLearly, the raw slope is close to 0.

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

Since the data points are all over the place, and they offset each other. Thus, the correlation value will always close to 0.

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

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

As x increases, y also shows an increasing trend. Thus, the raw slope will always close to positive 1.

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

Similarly, as x increases, y also shows an increasing trend. Thus, the correlation value will always close to positive 1.

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

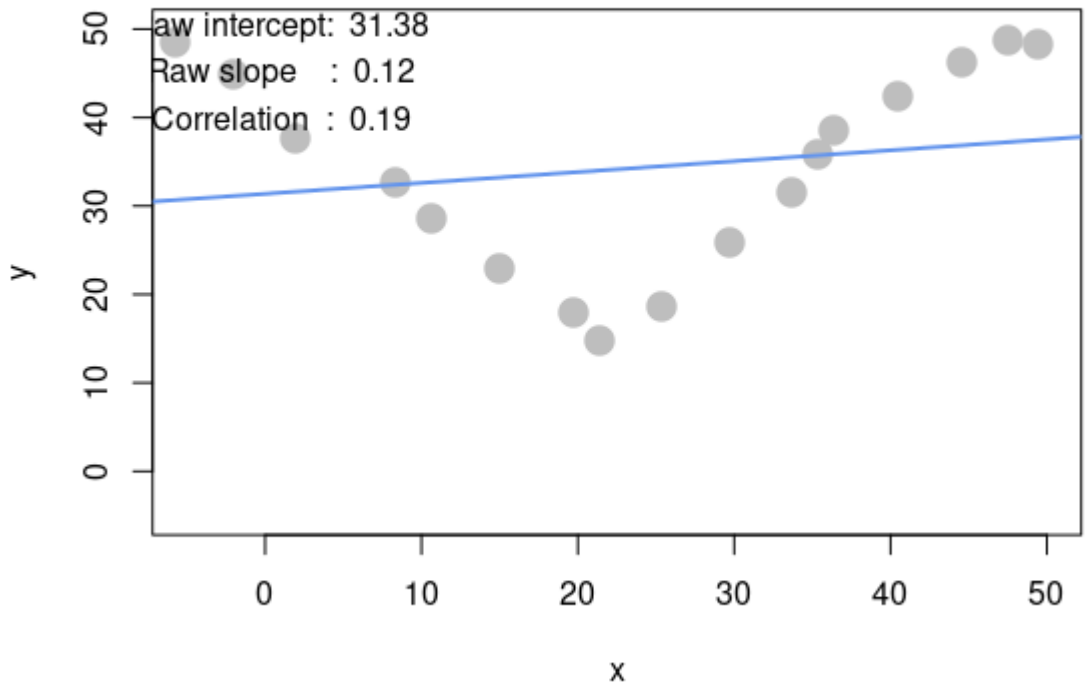
1. What raw slope of x and y would you generally expect? (note that x, y have the same scale)

As x decreases, y also shows an decreasing trend. Thus, the raw slope will always close to negative 1.

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

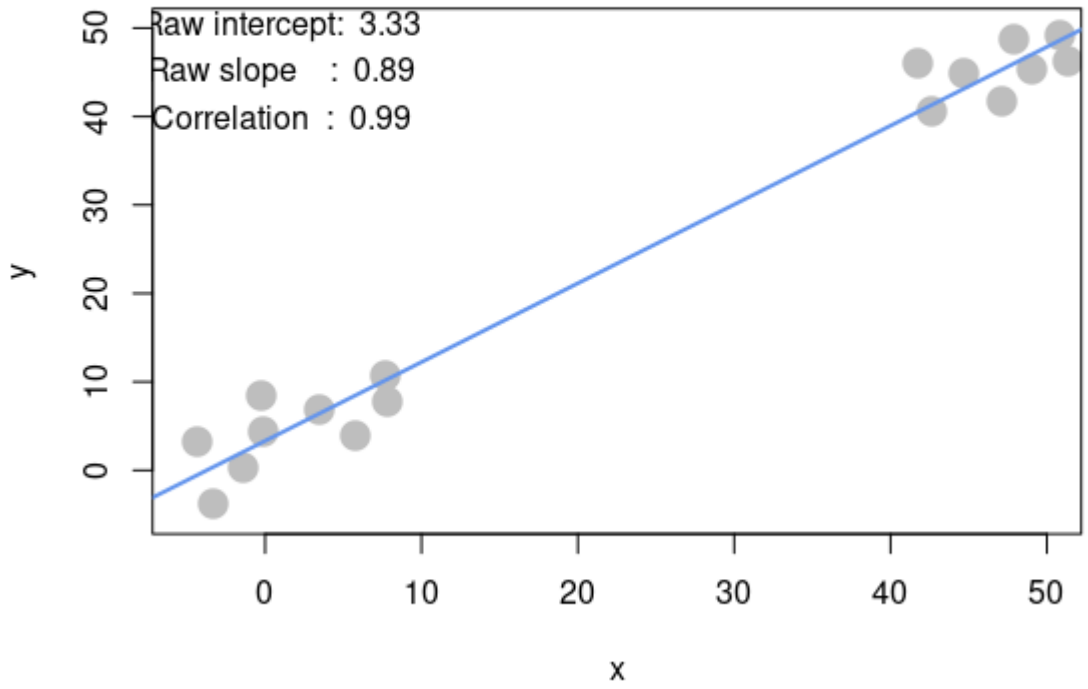
As x decreases, y also shows an decreasing trend. Thus, the raw slope will always close to negative 1.

e. Apart from any of the above scenarios, look for another pattern of data points with no correlation ($r \approx 0$).



r ≈ 0

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



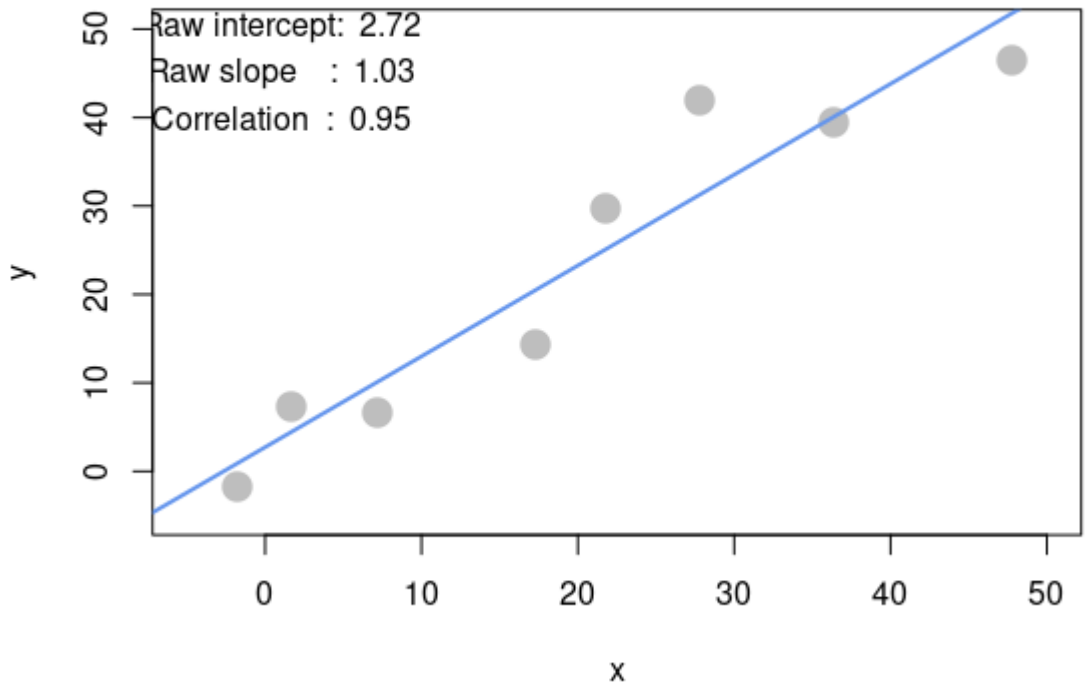
r ≈ 1

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

1. Record data points

```
data_points <-
  data.frame(
    x = c(
      -1.786964, 1.669799, 7.175015, 17.289249,
      21.770239, 27.787568, 36.365463, 47.759979
    ),
    y = c(
      -1.748132, 7.308126, 6.628907, 14.326727,
      29.722366, 41.948315, 39.457844, 46.476445
    )
  )
```

2. Use the `lm()` function to estimate the regression intercept and slope of pts to ensure they are same as the values reported in the simulation plot: `summary( lm( pts$y ~ pts$x ))`



X Y Intercept

```
summary( lm(data_points$y ~ data_points$x))
```

```
##
## Call:
## lm(formula = data_points$y ~ data_points$x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.157 -3.926 -1.629  3.310 10.681
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.7249     3.4536   0.789 0.460146
## data_points$x   1.0272     0.1354   7.587 0.000273 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.181 on 6 degrees of freedom
## Multiple R-squared:  0.9056, Adjusted R-squared:  0.8899
## F-statistic: 57.56 on 1 and 6 DF,  p-value: 0.0002727
```

Yes, the regression intercept and slope of `data_points` are the same.

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

```
cor(data_points)
```

```
##          x        y
## x 1.000000 0.951631
## y 0.951631 1.000000
```

Yes, the correlation value of x and y is the same as shown in the graph.

4. Now, re-estimate the regression using standardized values of both `x` and `y` from `data_points`

```
standardized_data_points <- data.frame(x = scale(data_points$x), y = scale(data_points$y))
summary( lm(standardized_data_points$y ~ standardized_data_points$x))
```

```
##
## Call:
## lm(formula = standardized_data_points$y ~ standardized_data_points$x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33058 -0.21078 -0.08745  0.17772  0.57350
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -2.241e-16  1.173e-01   0.000 1.000000
## standardized_data_points$x  9.516e-01  1.254e-01   7.587 0.000273 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3319 on 6 degrees of freedom
## Multiple R-squared:  0.9056, Adjusted R-squared:  0.8899
## F-statistic: 57.56 on 1 and 6 DF,  p-value: 0.0002727
```

```
cor(standardized_data_points)
```

```
##          x        y
## x 1.000000 0.951631
## y 0.951631 1.000000
```

Even though `(Intercept)` and `standardized_pts$x` values in the summary have changed, the correlation of x and y remain the same shown in the correlation table above.

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

The relationship between correlation and the standardized simple-regression stays the same.