HW_Week9_108020033

Che-Wei, Chang

2023-04-14 helped by 108020024

Question 1)

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

Download the CSV file piccollage_accounts_bundles.csv from Canvas.

You may either use read.csv() and data.frame to load the file as before, or you can optionally try learning how to use data.table – a high performance package for reading, writing, and managing large data sets.

Note: It will take time to fully learn data.table — but here's some code to get you started:

```
library(data.table)
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:

   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.

```
# I choose halloweenparty as a topic bundle that I want to check.
# I think holidayscheers, happyday, wonderland, Halloween2012StickerPack, HalloweenScream2013 might hav
# usage patterns as this bundle.
```

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
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.

```
# import the library
library(lsa)
```

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

```
# create cosine simulation matrix using cosine function in library lsa
cos_simulation_matrix <- cosine(ac_bundles_matrix)

# craete sort_data function to get the top 5 similar about each topic
sort_data <- function(x){
  top5 <- x[order(x, decreasing = T)[2 : 6]]
  attributes(top5)$names
}

# store the result in recommend_matrix
recommend_matrix1 <- t(apply(cos_simulation_matrix, 1, sort_data))

# show the head of it
head(recommend_matrix1)
```

```
##                 [,1]               [,2]               [,3]
## Maroon5V        "OddAnatomy"       "beatsmusic"       "xoxo"
## between         "BlingStickerPack" "xoxo"             "gwen"
## pellington      "springrose"       "8bit2"            "mmlm"
## StickerLite     "HeartStickerPack" "HipsterChicSara"  "Mom2013"
## saintvalentine  "nashnext"         "givethanks"       "teenwitch"
## HipsterChicSara "Random"           "HeartStickerPack" "wonderland"
##                 [,4]             [,5]
## Maroon5V        "alien"          "word"
## between         "OddAnatomy"     "AccessoriesStickerPack"
## pellington      "julyfourth"     "tropicalparadise"
## StickerLite     "Emome"          "Random"
## saintvalentine  "togetherwerise" "lovestinks2016"
## HipsterChicSara "Emome"          "StickerLite"
```

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

```
# Find the top 5 recommendations for the bundle of halloweenparty
recommend_matrix1['halloweenparty', ]
```

```
## [1] "family"          "babyanimals"     "hellobaby"       "toMomwithLove"
## [5] "hipsteroverlays"
```

(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.

```
# Remove the 1st column so that we can do arithmetic operation
df <- ac_bundles_dt[, -1]

# Calculate means of each bundle
bundle_means <- apply(df , 2, mean)

# Repeated means of each column vector
bundle_means_matrix <- t(replicate(nrow(df), bundle_means))

# Calculate column minus centered data matrix
ac_bundles_mc_b <- df - bundle_means_matrix

# Since the data type is list, we change the data type from list to matrix
ac_bundles_mc_b <- as.matrix(ac_bundles_mc_b)

# Correlation: cosine of mean-centered vectors
cor_sim <- cosine(ac_bundles_mc_b)

# Store the result in recommend_matrix and reuse the function
recommend_matrix2 <- t(apply(cor_sim, 1, sort_data))

# Show head of it
head(recommend_matrix2)
```

```
##                 [,1]               [,2]
## Maroon5V        "OddAnatomy"       "beatsmusic"
## between         "BlingStickerPack" "xoxo"
## pellington      "springrose"       "8bit2"
## StickerLite     "HeartStickerPack" "AnimalFriendsStickerPack"
## saintvalentine  "nashnext"         "givethanks"
## HipsterChicSara "Random"           "HeartStickerPack"
##                 [,3]               [,4]              [,5]
## Maroon5V        "xoxo"             "alien"           "word"
## between         "gwen"             "OddAnatomy"      "AccessoriesStickerPack"
## pellington      "tropicalparadise" "mmlm"            "julyfourth"
## StickerLite     "between"          "Emome"           "HipsterChicSara"
## saintvalentine  "teenwitch"        "togetherwerise"  "lovestinks2016"
## HipsterChicSara "wonderland"       "Emome"           "StickerLite"
```

3

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

```
# Find the top 5 recommendations for the bundle of halloweenparty
recommend_matrix2['halloweenparty', ]
```

```
## [1] "family"          "babyanimals"     "hellobaby"       "toMomwithLove"
## [5] "hipsteroverlays"
```

(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.

```
# count the result and reuse the function
recommend_matrix3 <- t(apply(cosine(ac_bundles_matrix - rowMeans(ac_bundles_matrix)), 1, sort_data))

# Show head of data
head(recommend_matrix3)
```

```
##                 [,1]               [,2]               [,3]
## Maroon5V        "OddAnatomy"       "word"             "xoxo"
## between         "BlingStickerPack" "xoxo"             "gwen"
## pellington      "springrose"       "8bit2"            "backtocool"
## StickerLite     "HeartStickerPack" "Mom2013"          "HipsterChicSara"
## saintvalentine  "togetherwerise"   "givethanks"       "teenwitch"
## HipsterChicSara "Random"           "HeartStickerPack" "wonderland"
##                 [,4]               [,5]
## Maroon5V        "beatsmusic"       "supercute"
## between         "Monsterhigh"      "OddAnatomy"
## pellington      "tropicalparadise" "julyfourth"
## StickerLite     "Emome"            "Random"
## saintvalentine  "mrcurlsport"      "arrows"
## HipsterChicSara "Emome"            "StickerLite"
```

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

```
# Find the top 5 recommendations for the bundle of halloweenparty
recommend_matrix3['halloweenparty', ]
```

```
## [1] "hellobaby"     "family"        "christmassnow" "cny2017"
## [5] "frombierun"
```

Question 2)

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

In our compstatslib package, you will find an interactive_regression() function that runs a simulation. You can click to add data points to the plotting area and see a corresponding regression line (hitting ESC will stop the simulation). You will also see three numbers: regression intercept – where the regression line crosses the y-axis; regression coefficient – the slope of x on y; correlation - correlation of x and y.

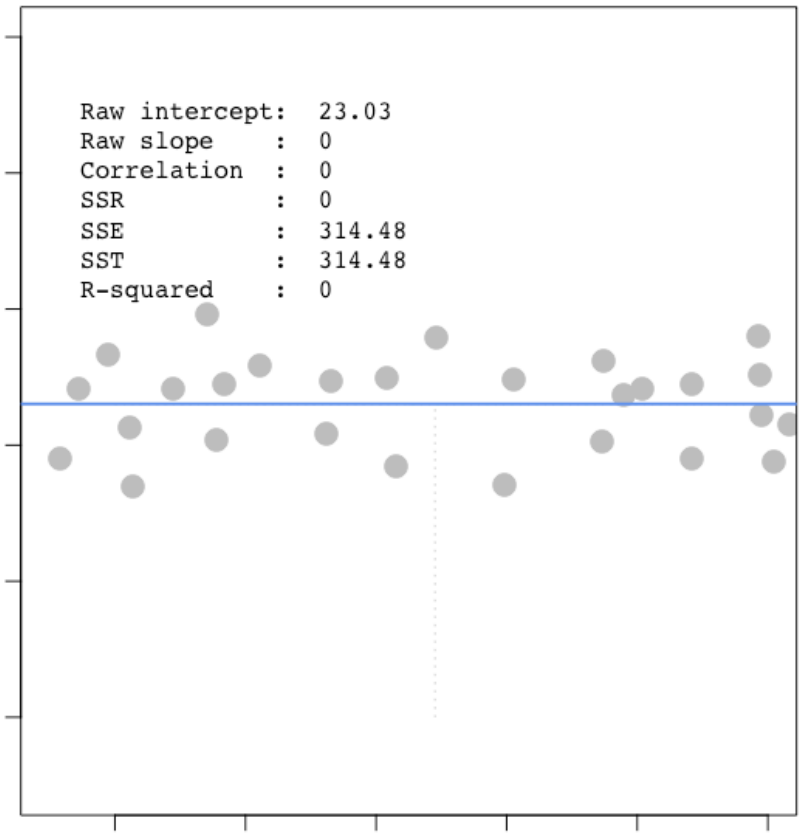
```
# import the library
library(compstatslib)
```

```
# Run interactive_regression() on terminal so that we can use it interactively.
# interactive_regression()
```

For each of the scenarios below, create the described set of points in the simulation. You might have to create each scenario a few times to get a general sense of them. Visual the scenarios a - d shown below.

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

```
knitr::include_graphics("Plot_1.png")
```



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

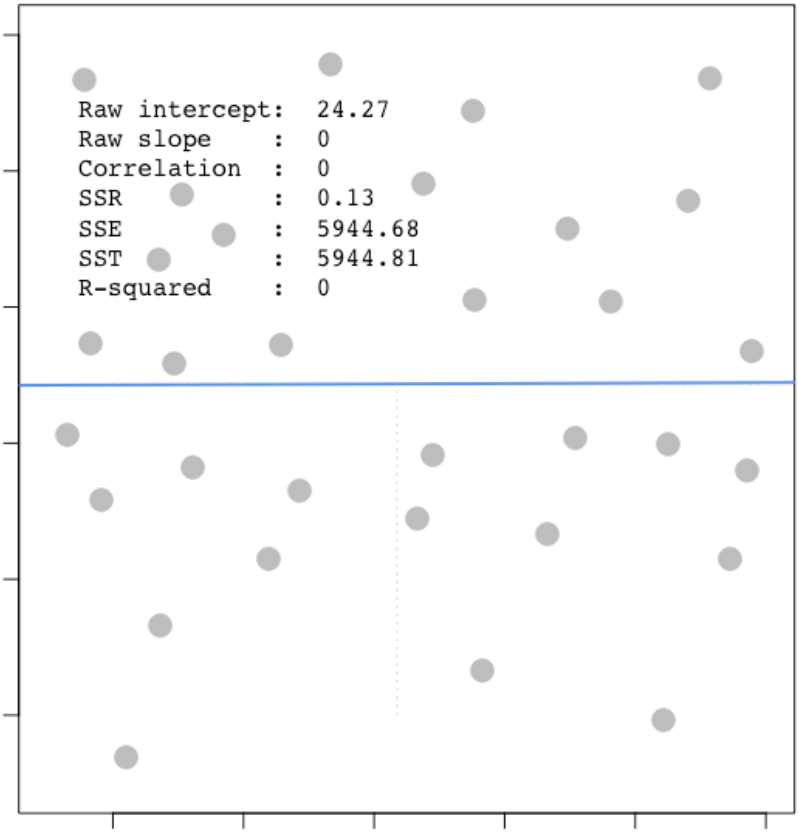
```
# The raw slope of x and y would be close to 0 because the points are randomly scattered along the x-ax
```

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

```
# The correlation of x and y would be close to 0 as well because there is no relationship between the x
```

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

```
knitr::include_graphics("Plot_2.png")
```



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

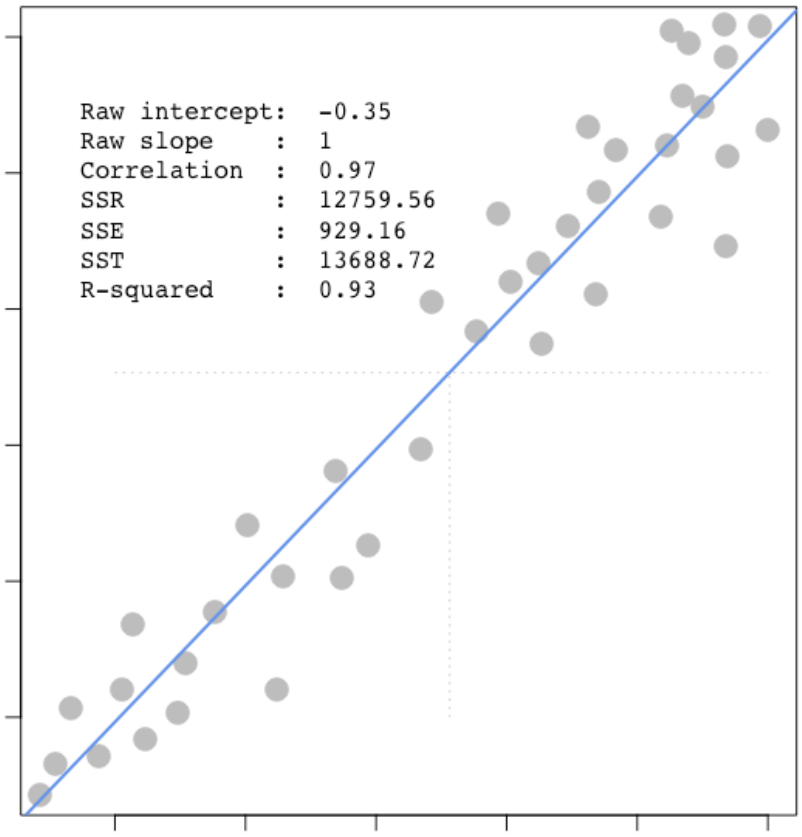
```
# The raw slope of x and y would be close to 0 because the points are randomly scattered across the ent
```

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

```
# The correlation of x and y would be close to 0 because there is no relationship between the x and y v
```

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

```
knitr::include_graphics("Plot_3.png")
```



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

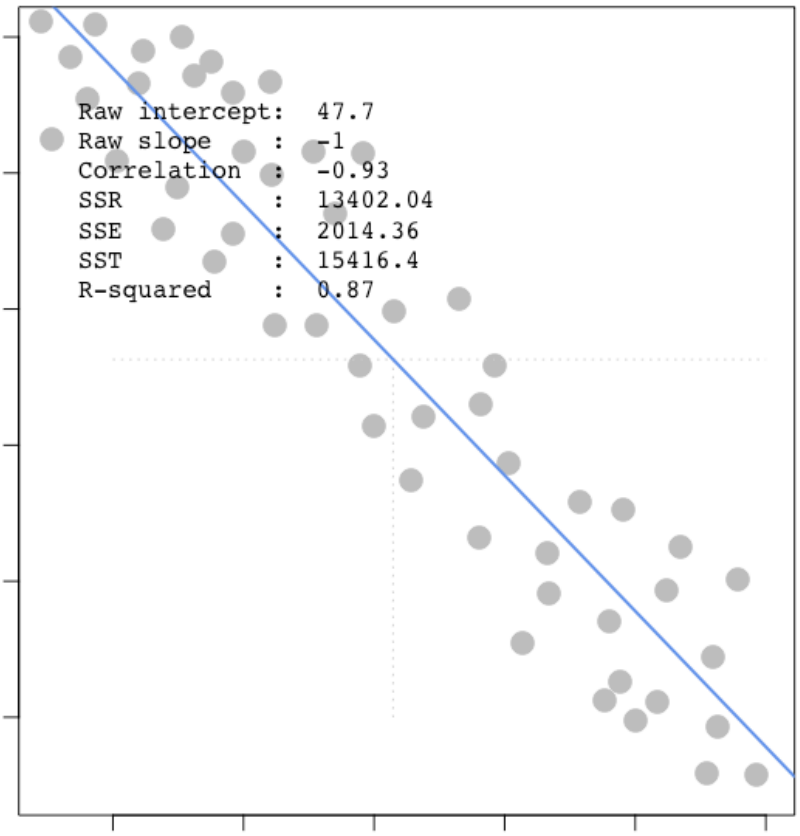
```
# The raw slope of x and y would be close to 1 because the points are trending upwards at a 45-degree a
```

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

```
# The correlation of x and y would be close to 1 because the points have a strong positive relationship
```

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

```
knitr::include_graphics("Plot_4.png")
```



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

```
# The raw slope of x and y would be close to -1 because the points are trending downwards at a 45-degre
```

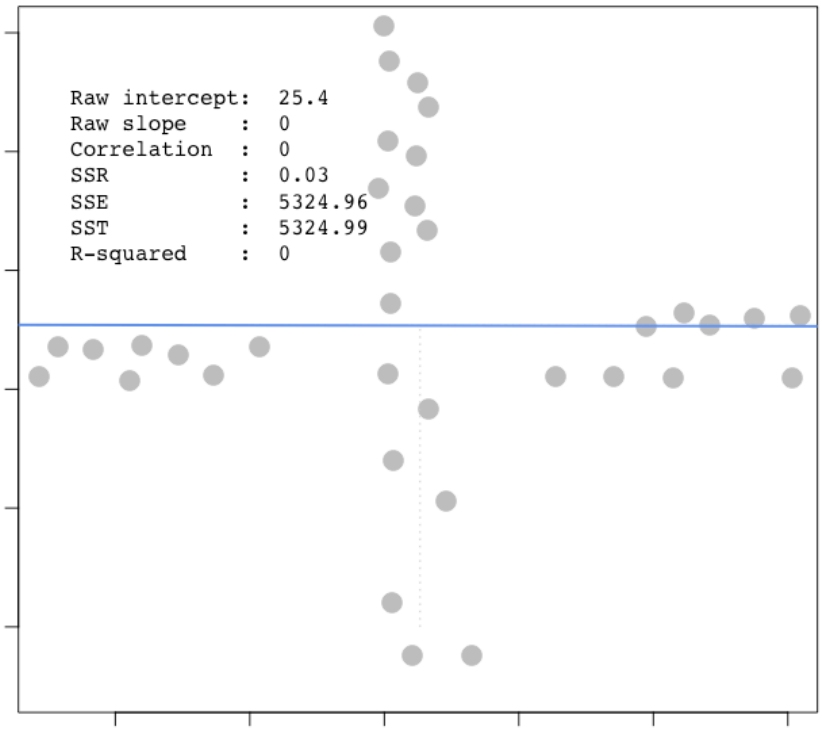
(ii). What is the correlation of x and y that you would generally expect?

```
# The correlation of x and y would be close to -1 because the points have a strong negative relationshi
```

e. 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?)

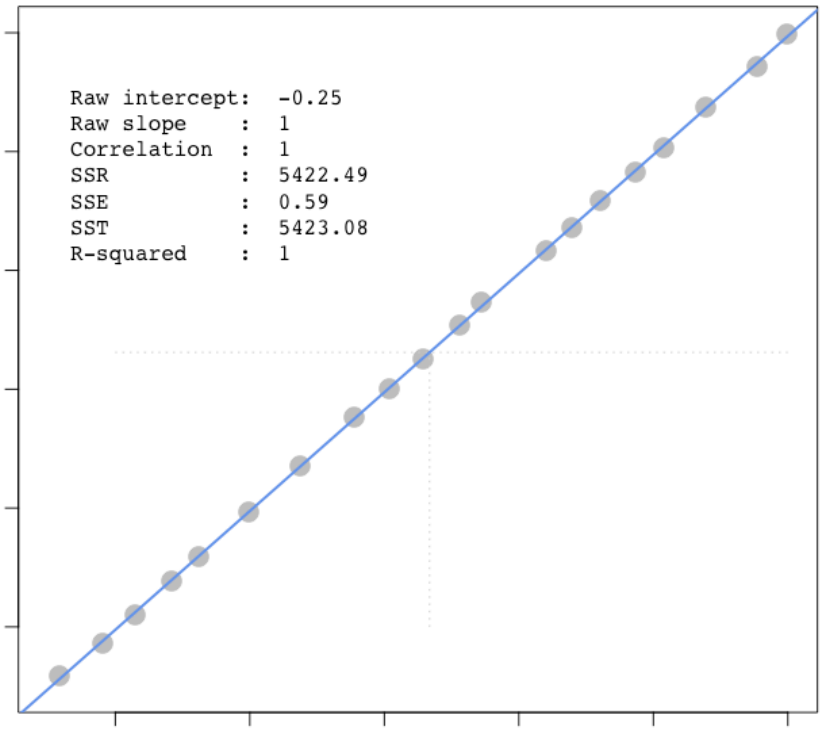
```
knitr::include_graphics("Plot_5.png")
```



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?)

```
knitr::include_graphics("Plot_6.png")
```



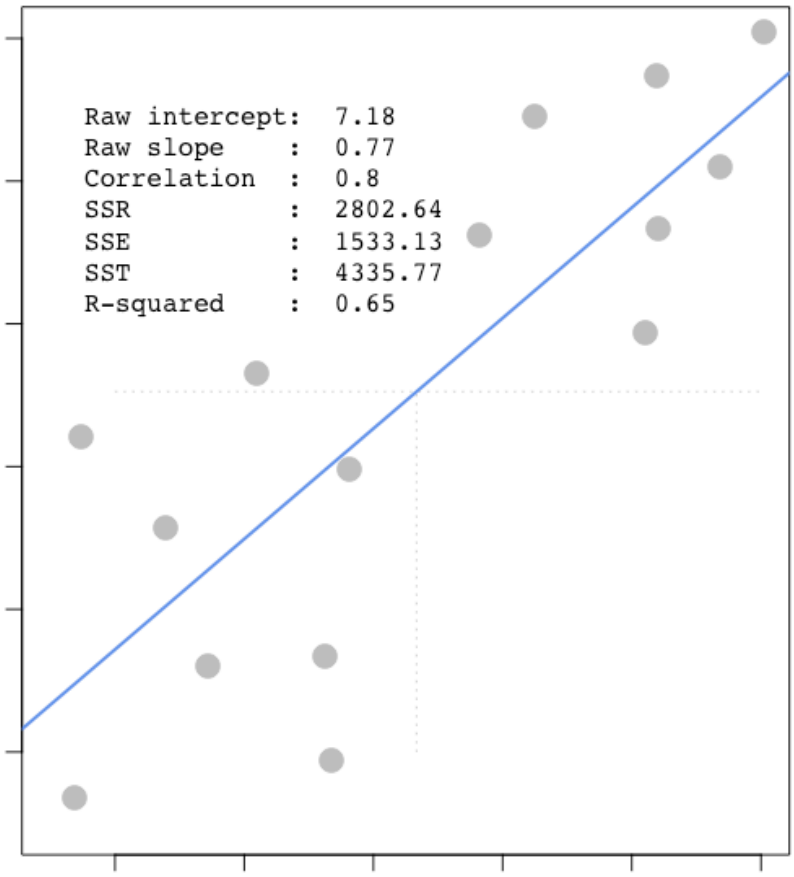
g. Let's see how correlation relates to simple regression, by simulating any linear relationship you wish:

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

(simulate either a positive or negative relationship)

```
# Since we can't use interactive_regression() in this chunk, we do in Console and record the data in th
x <- c(50.226545, -3.137300, 32.480549, 42.045767, 10.958810, 3.910755, 7.183066, 46.828375, 41.919908,
       , 28.201373, 41.038902, 16.244851, 16.748284, -2.633867)
y <- c(50.4591070, -3.2032377, 44.5346018, 36.6732392, 26.5332208, 15.7096056, 6.0253184, 41.0026853, 4
       , 36.2175081, 29.3815406, 6.7089151, -0.5827835, 22.0898420)
pts <- data.frame(x = x, y = y)
```

```
knitr::include_graphics("Plot_7.png")
```



(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))

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

```
##
## Call:
## lm(formula = pts$y ~ pts$x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.728  -7.337  -1.406   7.482  16.946
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.1825     4.6473   1.546 0.146212
## pts$x         0.7740     0.1588   4.875 0.000303 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.86 on 13 degrees of freedom
## Multiple R-squared:  0.6464, Adjusted R-squared:  0.6192
## F-statistic: 23.76 on 1 and 13 DF,  p-value: 0.0003034
```

```
# From the table, we can see that intercept = 7.1825, slope = 0.7740
```

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

```
# Estimate the correlation
cor(pts)
```

```
##           x         y
## x 1.0000000 0.8039903
## y 0.8039903 1.0000000
```

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

```
# Standardize the values of x, y and store them into new data frame
std_pts <- data.frame(std_x = scale(pts$x), std_y = scale(pts$y))

# Re-estimate the regression slope
summary(lm(std_pts$std_y ~std_pts$std_x))
```

```
##
## Call:
## lm(formula = std_pts$std_y ~ std_pts$std_x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.17785 -0.41692 -0.07987  0.42514  0.96293
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.720e-16  1.593e-01   0.000 1.000000
## std_pts$std_x  8.040e-01  1.649e-01   4.875 0.000303 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6171 on 13 degrees of freedom
## Multiple R-squared:  0.6464, Adjusted R-squared:  0.6192
## F-statistic: 23.76 on 1 and 13 DF,  p-value: 0.0003034
```

```
# From the table, we can see that intercept = -1.720e-16, slope = 8.040e-01
```

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

```
# When we do standardization, slope will equal to correlation and the intercept will be 0.
```