HW

110078509

202204

Question 1

```
data_df <- read.csv("piccollage_accounts_bundles.csv", row.names = 'acc
ount_id' )
```

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

i. **(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 tota
l 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 computin
g burden

# Find the index of non-zero values' index number of colname 'sweetmoth
ersday'
non_zero_index <- which(data_df$sweetmothersday !=0)
# it show row 3 and 19104 is not zero
```

```r
#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 2165 *to* 280.

```r
# 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
```

```
I instinct recommend the StickerLite , wonderland ,PhotoboothFest, Wint
erWonderland , CutieV.
```

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

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

```r
# Cosine-Sim
cos_all <- cosine(as.matrix(data_df))

# transfer matrix coss_all into df then add a added column as "IndexNam
e"
ac_bundles_df <- as.data.frame(cos_all) %>% rownames_to_column("IndexNa
me")

# 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 c
ase)
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 "IndexN
ame"
  ac_bundles_df <- as.data.frame(cos_all) %>% rownames_to_column("Index
Name") ;

  # 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 th
is 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 3 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 a
bove above.Therefore, due to my result, my can get any recommendation f
or the any bundle.
    However, for better comparison for the following question, I'd choo
se bundle 'sweetmothersday' as my baseline. And I re-constructed a dumm
y 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" "bestda
ddy"
## [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 f
unction an accounts-bundles matrix where each bundle (column) has alrea
dy 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 chang
e it)

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

(3). What are the top 5 recommendations for the bundle you chose to exp
lore 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);
```

```r
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 co
sine-sim into my consideration without the part of the denominator part
 of the function. Moverover, I did not apply any normalization to my da
ta.
```

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

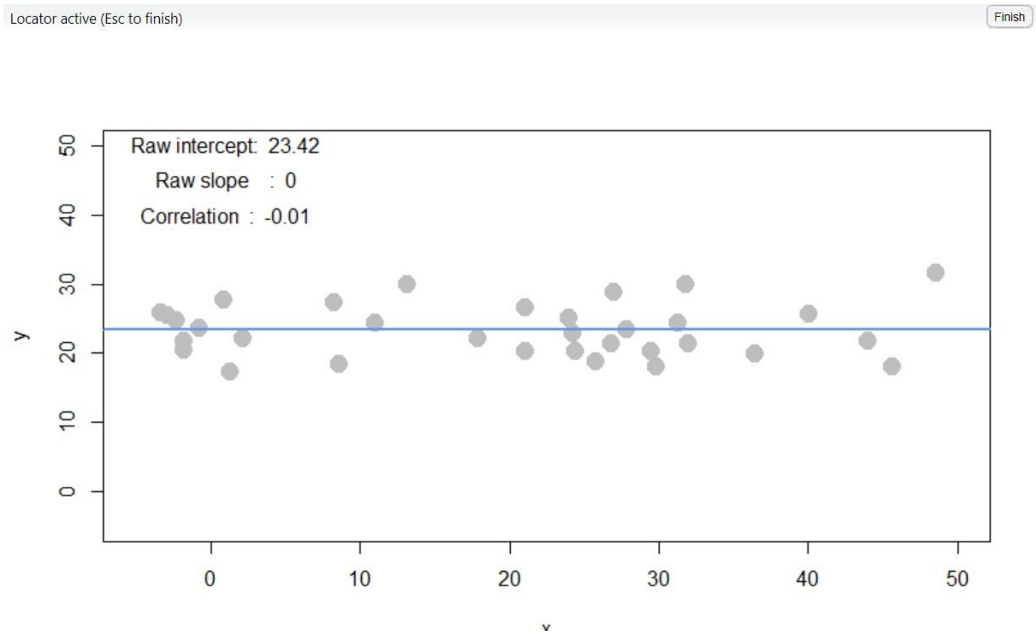
Cosine similarity + row-based mean normalization = adjusted-cosine

Cosine similarity + col-based mean normalization = correlation

This is my personal thoughts.

Question 2

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



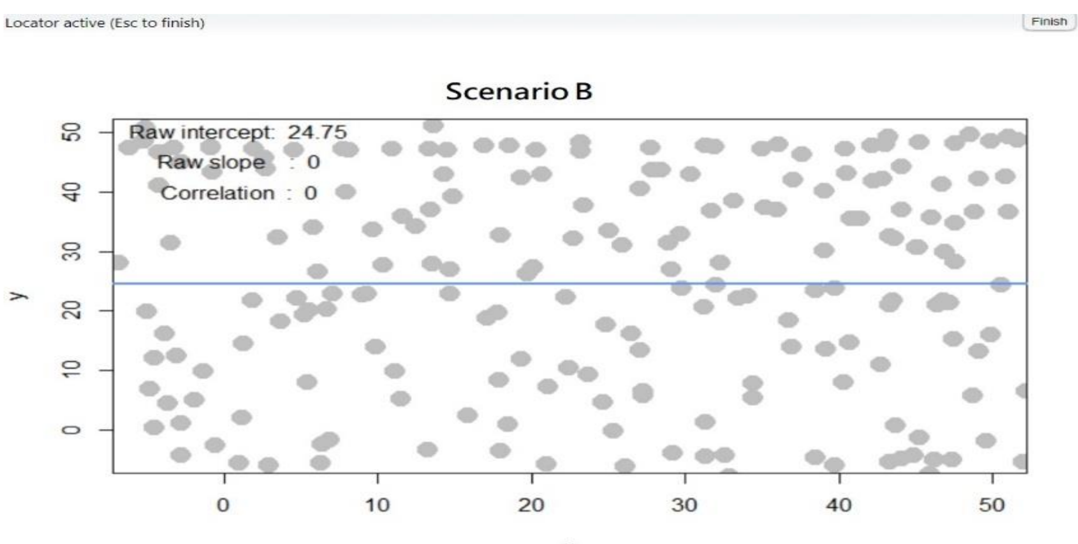
b. What raw slope of x and y would you generally expect?

`- *Ans:*   slope: 0`

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

`- *Ans:* correlation: 0`

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



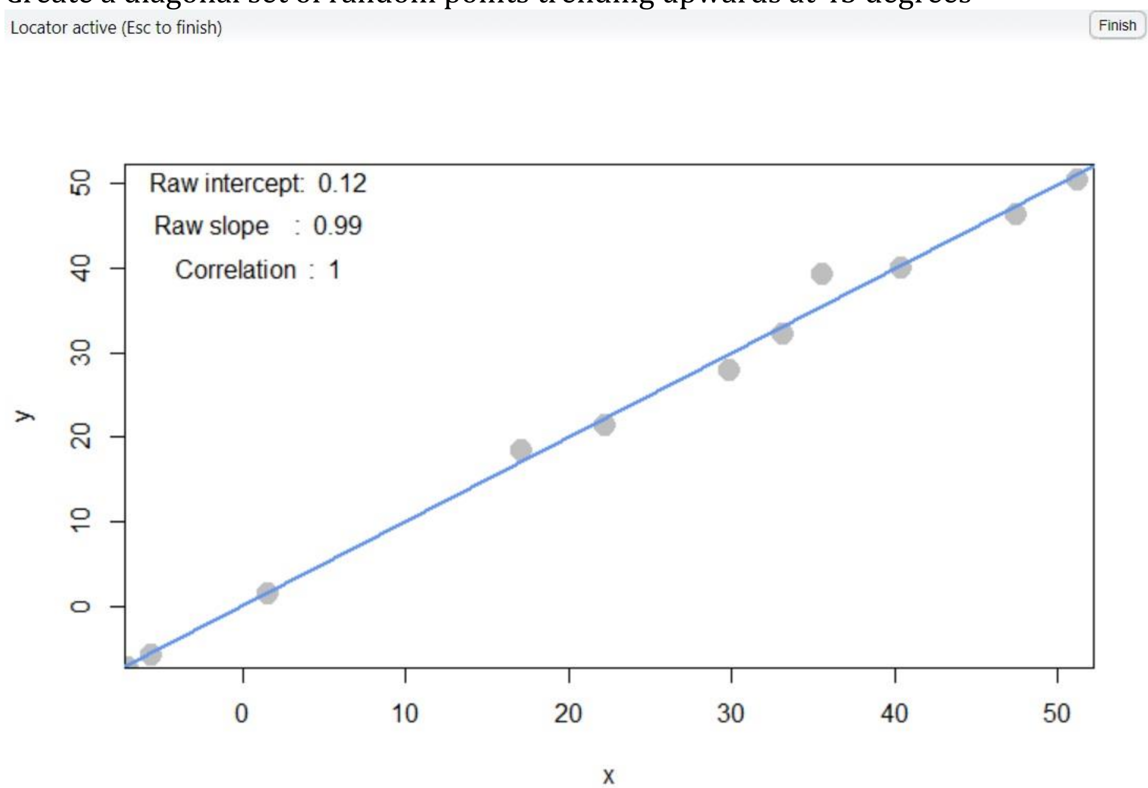
c. What raw slope of the x and y would you generally expect?

- *Ans:* slope: 0

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

- *Ans:* correlation: 0

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



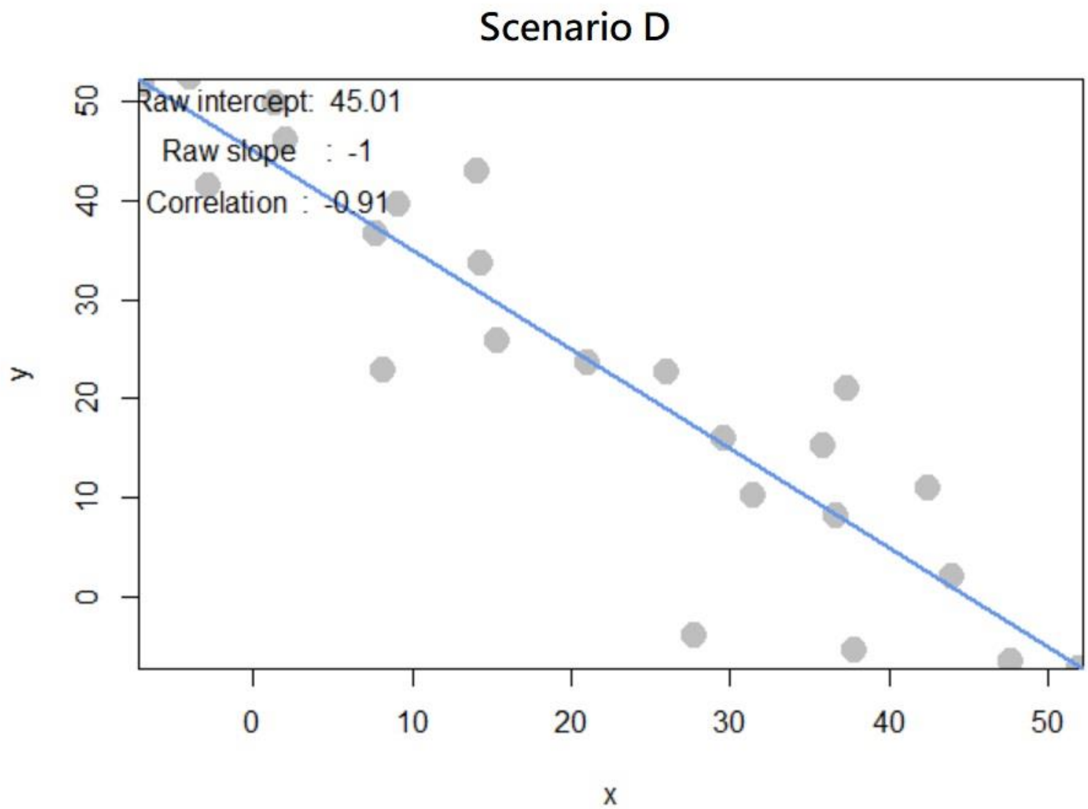
d. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)

- *Ans:* slope: 1

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

- *Ans:* correlation: 1

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



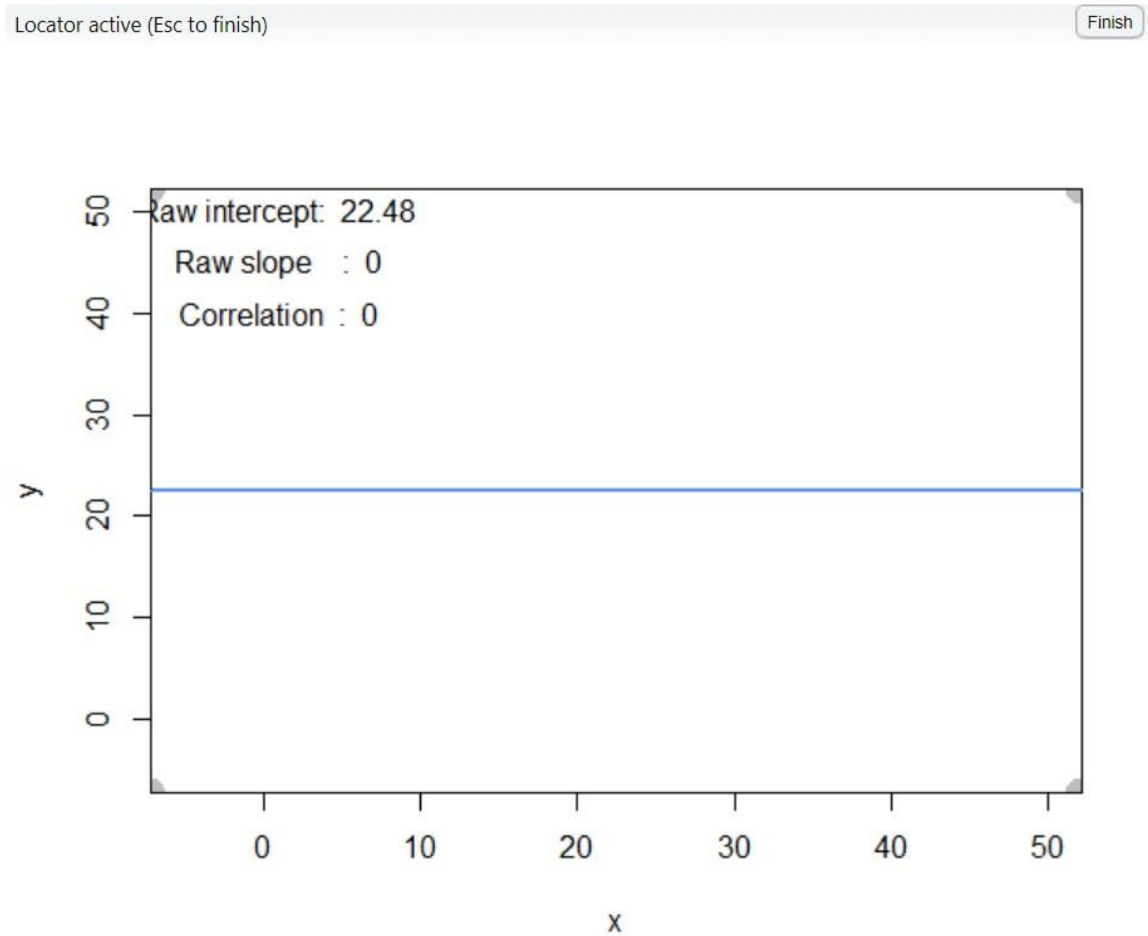
e. What raw slope of the x and y would you generally expect? (note that x, y have the same scale)

- *Ans:* slope: -1

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

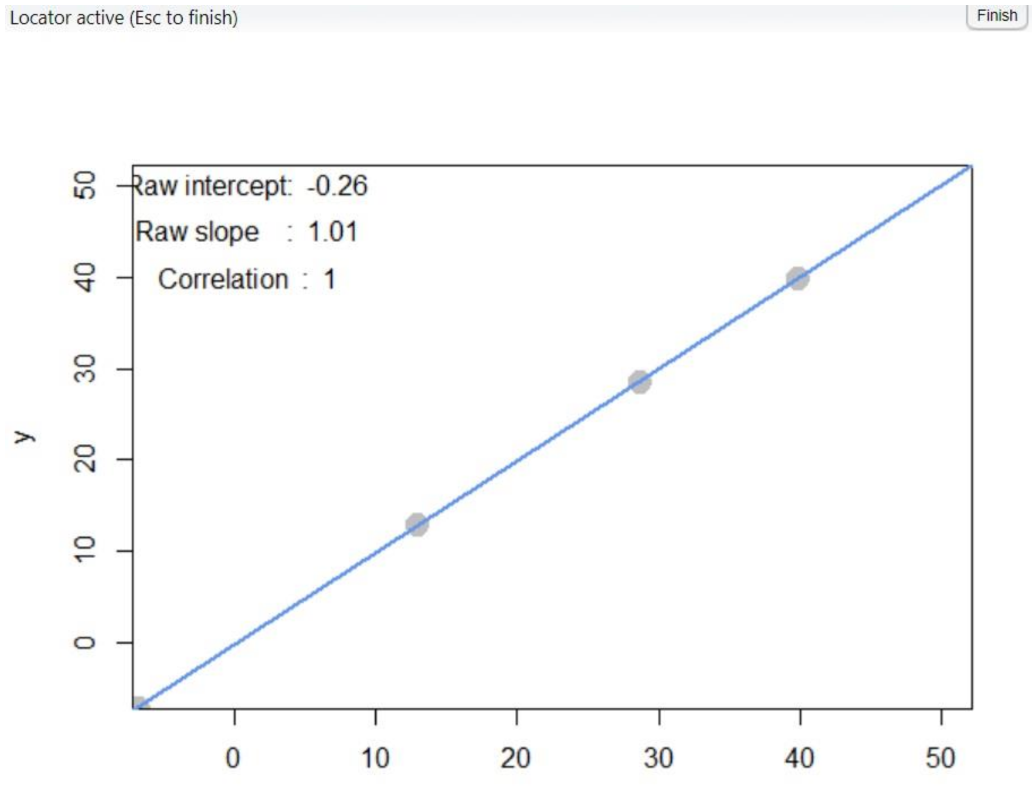
- *Ans:*correlation: -1

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



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

f. Apart from any of the above scenarios, find another pattern of data points with perfect correlation ($r \approx 1$). (can you find a scenario where the pattern visually suggests a different relationship?)



Done. If it's linear, it's 1.

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

I manually select the points below to form a positive slope regression plot:

	x	y
1	0.5342146	11.383443
2	11.6967341	5.270468
3	15.0326594	27.458302
4	28.3763608	34.929715
5	34.2783826	32.892057
6	46.4673406	38.552218
7	46.5956454	51.683793

h. Run the simulation and record the points you create: pts <- interactive_regression() (simulate either a positive or negative relationship)

```
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))

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

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

Residuals:
      1       2       3       4       5       6       7 
  3.009 -12.044   7.473   4.257  -2.507  -6.609   6.420 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)   7.9464     5.6744   1.400   0.2203   
pts$x         0.8009     0.1838   4.358   0.0073 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.991 on 5 degrees of freedom
Multiple R-squared:  0.7916,	Adjusted R-squared:  0.7499 
F-statistic: 18.99 on 1 and 5 DF,  p-value: 0.007305
```

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

```
> cor(pts)

cor(pts)
```

```
         x         y
x 1.0000000 0.8897137
y 0.8897137 1.0000000
```

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

```
temp =pts

mean_list <- sapply(pts,mean)
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)
```

[1] "slope: 0.890000"

Ans: The slope is 0.75.

v. 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)
```

[1] "correlation: 0.890000"

[1] "slope: 0.890000"

Ans: They’re the same.
