Homework8_BACS

109090035

4/12/2023

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:

```
ac_bundles_dt <- read_csv("/Users/user/Downloads/piccollage_accounts_bundles.csv")
```

```
Rows: 24649 Columns: 166
── Column specification ────────────────────────────────────────────────────────
────
Delimiter: ","
chr   (1): account_id
dbl (165): Maroon5V, between, pellington, StickerLite, saintvalentine, Hipst...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
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 App Store and take a look at the style and content of various bundles in their Sticker Store (iOS app: can see how many recommendations does each bundle have? Android app might not have recommendations)

Based on the bundle in the PicCollege apps , I choose “Note to Self” as the bundle i want to examine.

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.

I use my intuition to guess five other similar bundle :

1.Thank_You_Teacher

2.Sale_label

3.Grad_Party

4.Back_to_school_craft

5.Autumn_Journaling

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

```
library(lsa)
```

```
Loading required package: SnowballC
```

```
cosine_ac <- cosine(ac_bundles_matrix)

# Number of top recommendations
top_n <- 5
num_bundles <- 165

# Find the top N recommendations for each bundle
recommendation_indices <- t(apply(cosine_ac, 1, function(row) {
  sort(row, decreasing = TRUE, index.return = TRUE)$ix[-1][1:top_n]
}))

# Create the recommendation matrix
recommendation_matrix <- matrix("", nrow = top_n + 1, ncol = num_bundles)
rownames(recommendation_matrix) <- c("Bundle", paste(1:top_n, "st", sep = ""))
colnames(recommendation_matrix) <- paste("Bundle", 1:num_bundles)


# Fill in the recommendation matrix
for (i in 1:num_bundles) {
  recommendation_matrix[1, i] <- colnames(cosine_ac)[i]
  for (j in 1:top_n) {
    recommendation_matrix[j + 1, i] <- colnames(cosine_ac)[recommendation_indices[i, j]]
  }
}
```

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_top_recommendations <- function(ac_bundles_matrix) {
  # Calculate cosine similarity
  cosine_ac <- cosine(ac_bundles_matrix)

  # Number of top recommendations
  top_n <- 5
  num_bundles <- ncol(ac_bundles_matrix)

  # Find the top N recommendations for each bundle
  recommendation_indices <- t(apply(cosine_ac, 1, function(row) {
    sort(row, decreasing = TRUE, index.return = TRUE)$ix[-1][1:top_n]
  }))

  # Create the recommendation matrix
  recommendation_matrix <- matrix("", nrow = top_n + 1, ncol = num_bundles)
  rownames(recommendation_matrix) <- c("Bundle", paste(1:top_n, "st", sep = ""))
  colnames(recommendation_matrix) <- colnames(ac_bundles_matrix)

  # Fill in the recommendation matrix
  for (i in 1:num_bundles) {
    recommendation_matrix[1, i] <- colnames(cosine_ac)[i]
    for (j in 1:top_n) {
      recommendation_matrix[j + 1, i] <- colnames(cosine_ac)[recommendation_indices[i, j]]
    }
  }

  return(recommendation_matrix)
}
result <- recommendation_matrix
```

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

Recall the bundle i chose to explore earlier: “notetoself”. Now, we use the get_top_5_recommendations function to get the top 5 recommendations for this bundle.

```r
print_bundle_recommendations <- function(recommendation_matrix, bundle_name) {
  bundle_index <- which(recommendation_matrix[1,] == bundle_name)

  if (length(bundle_index) > 0) {
    bundle_recommendations <- recommendation_matrix[, bundle_index, drop = FALSE]
    print(bundle_recommendations)
  } else {
    cat("Bundle not found:", bundle_name)
  }
}

# Call the function to print recommendations for the 'notetoself' bundle
print_bundle_recommendations(result, "notetoself")
```

```
       Bundle 163
Bundle "notetoself"
1st    "salelabels"
2st    "gradparty"
3st    "AnimalFriendsStickerPack"
4st    "StampStickerPack"
5st    "PartyStickerPack"
```

And based on the intuition i guess in Question1, some of my intuition is the same as the cosine similarity such as “salelabels”and “gradparty”.

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.

```r
# Mean-center the accounts-bundles matrix
mean_centered_ac_bundles_matrix <- scale(ac_bundles_matrix, scale = FALSE)


# Get the top recommendations using the mean-centered matrix
result_mean_centered <- get_top_recommendations(mean_centered_ac_bundles_matrix)

# Print the result
```

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

```r
# Assuming 'result_mean_centered' is the recommendation matrix from the previous step
bundle_name <- "notetoself"

# Find the column index of the desired bundle
bundle_index <- which(colnames(result_mean_centered) == bundle_name)

# Check if the bundle exists
if (length(bundle_index) > 0) {
  # Extract the recommendations for the desired bundle
  bundle_recommendations_mean_centered <- result_mean_centered[, bundle_index, drop = FALSE]
  print(bundle_recommendations_mean_centered)
} else {
  cat("Bundle not found:", bundle_name)
}
```

```
       notetoself
Bundle "notetoself"
1st    "salelabels"
2st    "gradparty"
3st    "AnimalFriendsStickerPack"
4st    "StampStickerPack"
5st    "PartyStickerPack"
```

We can see they are the same result even if question 2 was mean centered.

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.

```
# Mean-center each account (row) in the matrix
ac_bundles_matrix_mean_centered <- t(apply(ac_bundles_matrix, 1, function(row) {
  row - mean(row)
}))


# Use the get_top_recommendations function with the mean-centered matrix
result_mean_centered <- get_top_recommendations(ac_bundles_matrix_mean_centered)
```

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

```
# Print the recommendations for the 'notetoself' bundle
print_bundle_recommendations(result_mean_centered, "notetoself")
```

```
       notetoself
Bundle "notetoself"
1st    "gradparty"
2st    "christmassnow"
3st    "cny2017"
4st    "frombierun"
5st    "floralwedding"
```

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?

In the final result, we can see cosine similarity and correlation have the same recommend result, and adjusted-cosine is slightly different, this might caused by the below reason :

1.Limited human perspective: Intuition-based recommendations are based on personal experiences and knowledge, which might be limited in scope. Computational models, on the other hand, can analyze a much larger dataset and identify patterns that may not be apparent to humans.

2.Bias: Humans can be biased in their recommendations, while computational models are more objective as they are based on mathematical calculations.

3.Different similarity measures: The three geometric recommendation models (cosine similarity, correlation, and adjusted cosine similarity) use different similarity measures. Each measure captures different aspects of the relationship between bundles, which might result in different recommendations.

4.Mean-centering effects: The correlation and adjusted-cosine based recommendations use mean-centered data, which accounts for the differences in user preferences and bundle ratings. This can lead to recommendations that may not be apparent from the raw data.

5.Noise in data: The presence of noise in the data, such as outliers or missing values, can affect the recommendations generated by the models. Intuition-based recommendations may not be affected by noise in the same way, as humans can filter out irrelevant information.

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

1.Cosine similarity: Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. It is a geometric measure that focuses on the direction of the vectors, ignoring their magnitude. Cosine similarity ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating no similarity. In the context of recommendation systems, cosine similarity is typically used to compare users or items based on their ratings or preferences, with higher values indicating greater similarity.

2.Correlation: Correlation, specifically Pearson correlation, measures the linear relationship between two variables. It ranges from -1 (strong negative correlation) to 1 (strong positive correlation), with 0 indicating no correlation. In recommendation systems, correlation is often used to compare users or items by assessing how their ratings or preferences change together. Correlation-based methods account for differences in user rating scales, as they consider the mean rating for each user/item.

3.Adjusted-cosine: Adjusted-cosine similarity is a variation of cosine similarity that accounts for differences in user preferences. It is calculated by first mean-centering the data by subtracting the mean rating of each user (or item) from their individual ratings. Then, the cosine similarity is calculated using the mean-centered data. By doing this, adjusted-cosine similarity accounts for differences in user rating scales and can provide more accurate recommendations in cases where users have different baselines for their ratings.

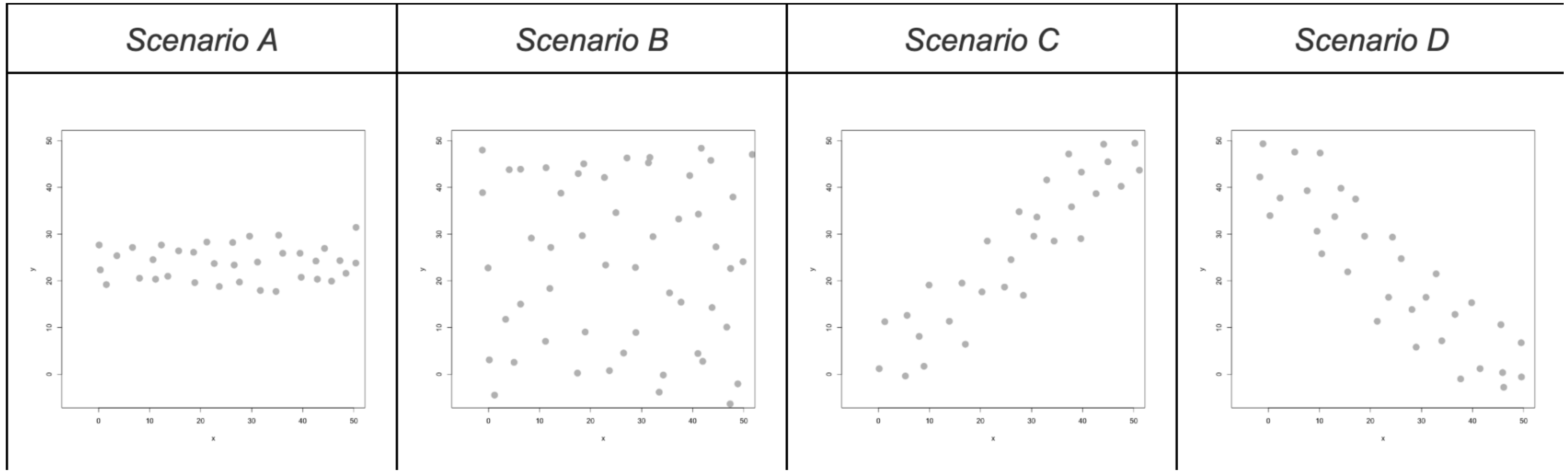
Question 2) Correlation is at the heart of many data analytic methods so let’s explore it further.

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.

```r
library(compstatslib)
#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.

```
knitr::include_graphics("/Users/user/Desktop/123.png")
```



a. Scenario 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?

Raw slope of x and y: Since the points are distributed horizontally, the slope of the regression line would be approximately 0.

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

Correlation of x and y: The correlation would be close to 0, as there is no apparent linear relationship between x and y.

b. Scenario B: Create a 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?

Raw slope of x and y: The slope of the regression line would vary depending on the distribution of the points, but it would generally be close to 0, as there is no clear linear relationship between x and y.

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

Correlation of x and y: The correlation would be close to 0, as there is no apparent linear relationship between x and y.

c. Scenario 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)

Raw slope of x and y: Since the points are trending upwards at a 45-degree angle, the slope of the regression line would be approximately 1 (assuming x and y have the same scale).

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

Correlation of x and y: The correlation would be close to 1, indicating a strong positive linear relationship between x and y.

d. Scenario 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)

Raw slope of x and y: Since the points are trending downwards at a 45-degree angle, the slope of the regression line would be approximately -1 (assuming x and y have the same scale).

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

Correlation of x and y: The correlation would be close to -1, indicating a strong negative linear relationship between x and y.

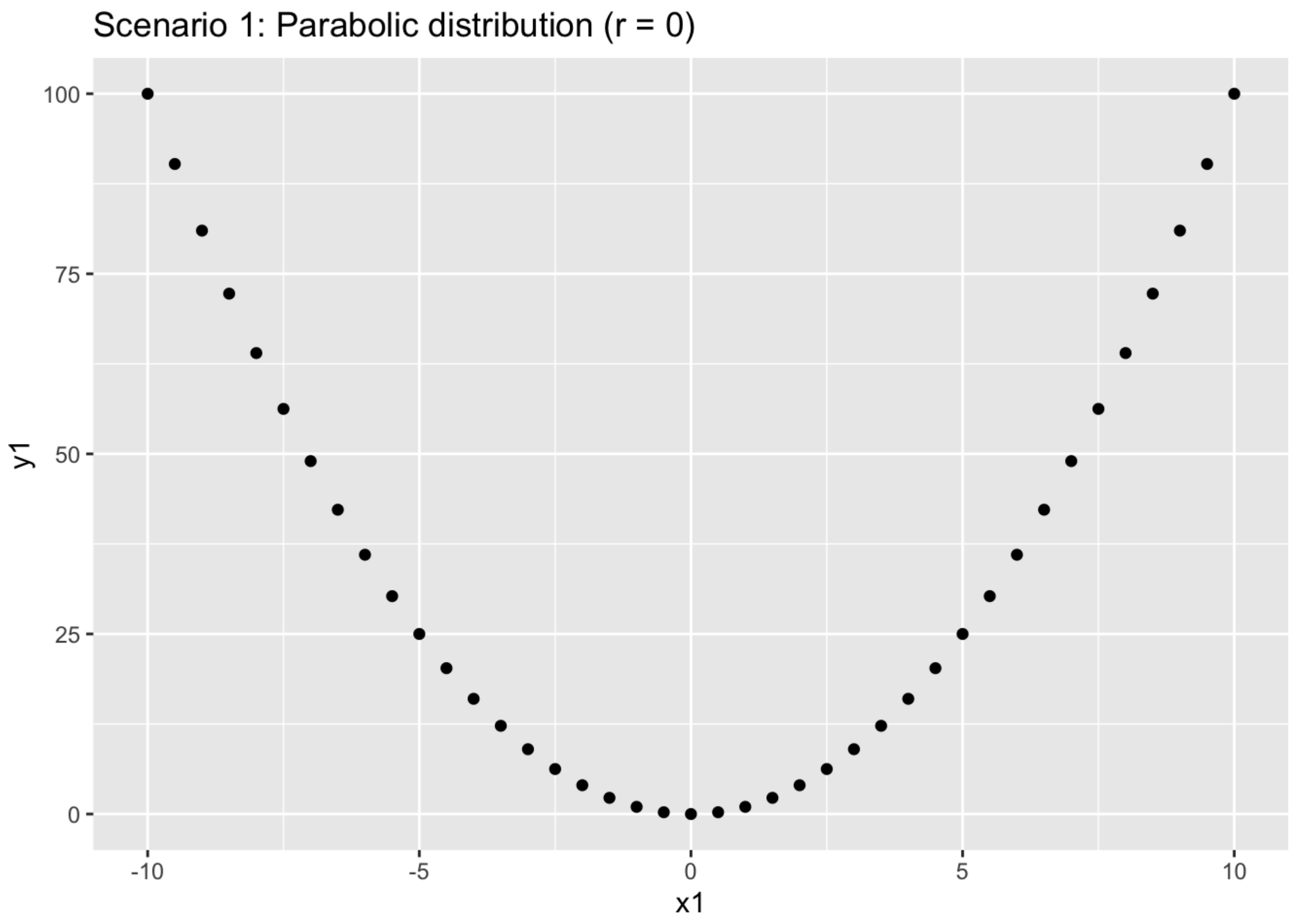
e. Apart from any of the above scenarios, find another pattern of data points with no correlation (r ≈ 0).(can create a pattern that visually suggests a strong relationship but produces r ≈ 0?)

```
# Scenario 1: Parabolic distribution (y = x^2)
x1 <- seq(-10, 10, by = 0.5)
y1 <- x1^2
data1 <- data.frame(x1, y1)
cor1 <- cor(x1, y1)

# Scenario 2: Straight line with unevenly spaced points
x2 <- c(1, 2, 2.2, 4, 6)
y2 <- c(1, 2, 2.2, 4, 6)
data2 <- data.frame(x2, y2)
cor2 <- cor(x2, y2)

# Plotting
p1 <- ggplot(data1, aes(x1, y1)) +
  geom_point() +
  ggtitle(paste0("Scenario 1: Parabolic distribution (r = ", round(cor1, 2), ")"))

p2 <- ggplot(data2, aes(x2, y2)) +
  geom_point() +
  ggtitle(paste0("Scenario 2: Unevenly spaced points (r = ", round(cor2, 2), ")"))
```

After trying using teacher’s interactive_regression() function, I use ggplot to demonstrate the pattern i found that produced a r = 0

The first plot shows a parabolic distribution (y = x^2) with a correlation coefficient close to 0.

```
p1
```



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?)

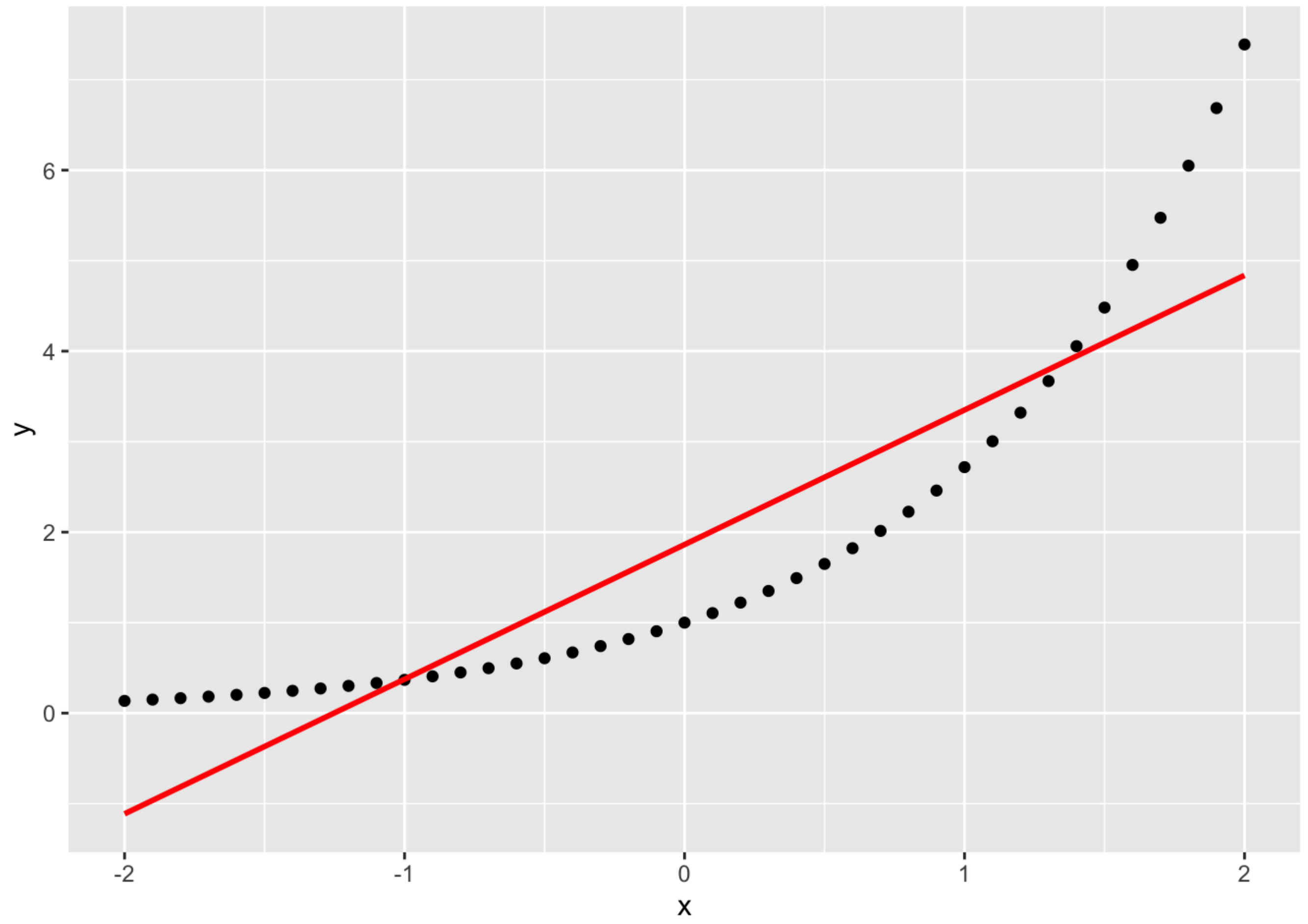
To create a scenario where data points have a perfect correlation (r ≈ 1) but visually suggest a different relationship, we can use a nonlinear transformation of the data, such as exponential or logarithmic relationships. Here's an example of ggplot using an exponential relationship:

```
x <- seq(-2, 2, 0.1)
y <- exp(x)
data <- data.frame(x, y)
cor(data$x, data$y)
```

```
[1] 0.8941043
```

```
ggplot(data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red")
```

```
`geom_smooth()` using formula = 'y ~ x'
```

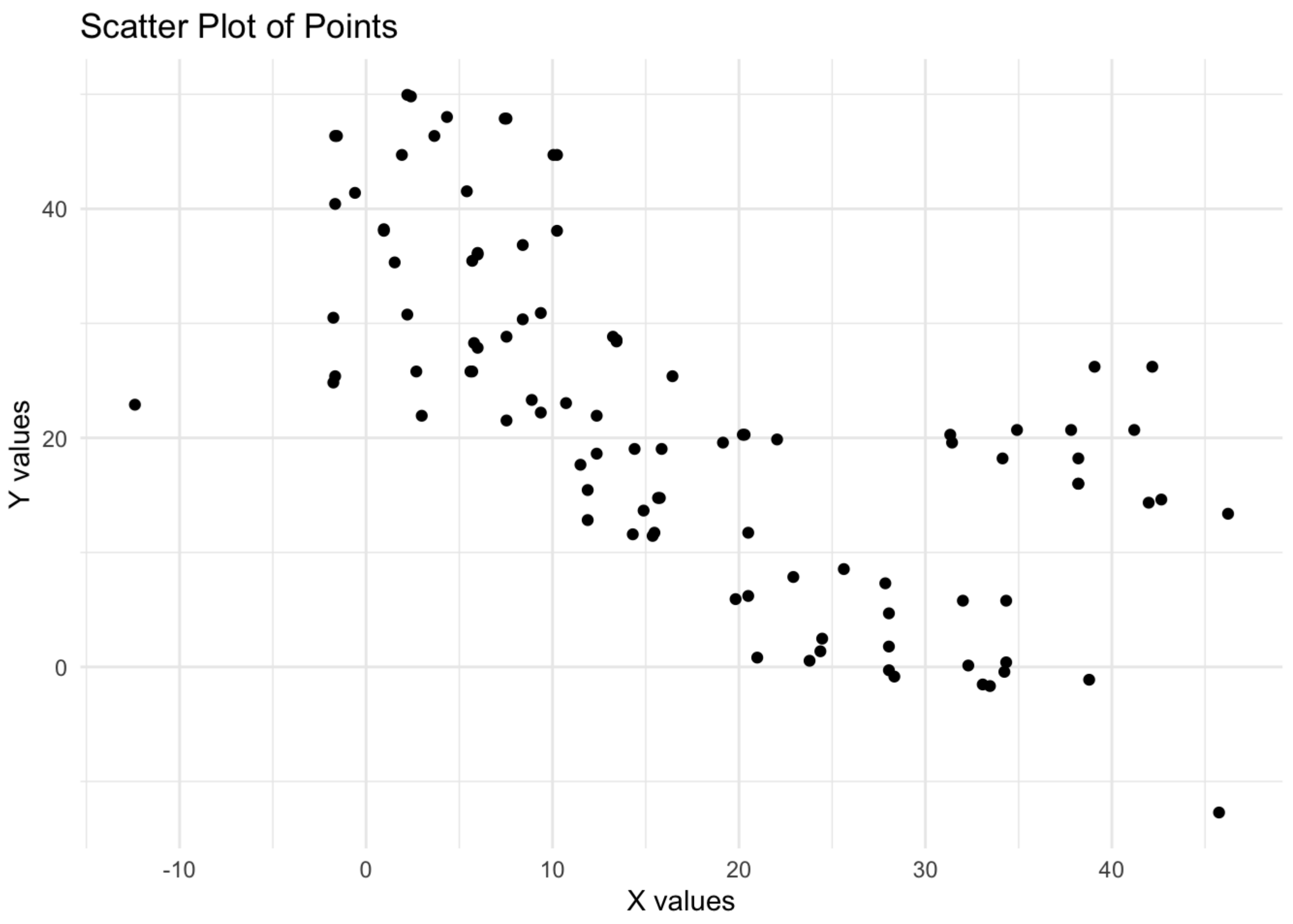


g, Let’s see how correlation relates to simple regression, by simulating any linear relationship you wish:

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

After simulate pts , i store it as pts and plot it out

```
pts <- read.csv("/Users/user/Downloads/hw1/pts.csv")
ggplot(data = pts, aes(x = x, y = y)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Scatter Plot of Points",
       x = "X values",
       y = "Y values")
```



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(ptsy ptsx))

```
model <- lm(pts$y ~ pts$x)
summary(model)
```

```


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

Residuals:
    Min      1Q  Median      3Q     Max 
-20.181  -9.430  -1.778   8.962  22.778 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 34.07378    1.73815  19.603  < 2e-16 ***
pts$x       -0.72657    0.07899  -9.198 7.85e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.75 on 96 degrees of freedom
Multiple R-squared:  0.4685,	Adjusted R-squared:  0.4629 
F-statistic: 84.61 on 1 and 96 DF,  p-value: 7.851e-15
```

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

```
cor(pts$x, pts$y)
```

```
[1] -0.6844461
```

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

```
pts_standardized <- data.frame(x = scale(pts$x), y = scale(pts$y))
```

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

```
model_standardized <- lm(y ~ x, data = pts_standardized)
summary(model_standardized)
```

```

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

Residuals:
    Min      1Q  Median      3Q     Max 
-1.3757 -0.6428 -0.1212  0.6109  1.5527 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3.898e-16  7.403e-02   0.000        1    
x           -6.844e-01  7.441e-02  -9.198 7.85e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7329 on 96 degrees of freedom
Multiple R-squared:  0.4685,    Adjusted R-squared:  0.4629 
F-statistic: 84.61 on 1 and 96 DF,  p-value: 7.851e-15
```

when we standardize both x and y variables (mean of 0 and standard deviation of 1), the slope of the simple linear regression becomes equal to the Pearson correlation coefficient between x and y. The standardized regression slope and the Pearson correlation coefficient are equivalent measures of the linear association between two variables.