Assignment1 Q9

Jiahang Wang

```
# Load necessary libraries
library(readr)
library(dplyr)
```

```
# load data
data.source <- paste0("https://raw.githubusercontent.com/",
"mcgillstat/regression/main/data/medals_2008.csv")
medals <- read_csv(data.source, col_names = FALSE)
colnames(medals) <- c("Country", "Gold", "Silver", "Bronze")
# e.g. print out the first 11 countries
print(medals, n = 11)
```

```
## # A tibble: 87 x 4
##    Country       Gold Silver Bronze
##    <chr>        <dbl>  <dbl>  <dbl>
##  1 China           48     22     30
##  2 UnitedStates    36     39     37
##  3 Russia          24     13     23
##  4 GreatBritain    19     13     19
##  5 Germany         16     11     14
##  6 Australia       14     15     17
##  7 SouthKorea      13     11      8
##  8 Japan            9      8      8
##  9 Italy            8      9     10
## 10 France           7     16     20
## 11 Netherlands      7      5      4
## # ... with 76 more rows
```

Q1

```
# Calculate the number of nations with at least one gold medal
nations_with_gold <- medals %>%
  filter(Gold > 0) %>%
  nrow()

# Print the result
print(nations_with_gold)
```

```
## [1] 69
```

Q2

```
# Center and scale the medal counts, keeping Country as the first column
medals_p <- medals %>%
  mutate(across(c(Gold, Silver, Bronze), scale))

medals_p
```

```
## # A tibble: 87 x 4
##    Country      Gold[,1] Silver[,1] Bronze[,1]
##    <chr>           <dbl>      <dbl>      <dbl>
##  1 China           6.07       3.24       3.93
##  2 UnitedStates    4.43       6.23       5.01
##  3 Russia          2.79       1.67       2.85
##  4 GreatBritain    2.10       1.67       2.24
##  5 Germany         1.69       1.32       1.47
##  6 Australia       1.42       2.02       1.93
##  7 SouthKorea      1.28       1.32       0.542
##  8 Japan           0.731      0.790      0.542
##  9 Italy           0.594      0.965      0.850
## 10 France          0.457      2.19       2.39
## # ... with 77 more rows
```

- Centering:

Centering is essential when data doesn't naturally have a mean of zero. If the data is not centered, the first principal component may capture the data's overall average instead of its underlying patterns and variances. It's also necessary when variables have different means. Failing to center can lead to the PCA being sensitive to variables with larger means, potentially distorting the interpretation of principal components. Centering ensures that the origin (0,0) in the multidimensional space corresponds to the center of data distribution, making it easier to interpret the directions of maximum variance.

- Scaling:

Scaling is important when variables have different units or scales (e.g., one variable is measured in meters, another in grams). Without scaling, variables with larger numeric ranges can dominate the PCA analysis, as PCA is based on variance, and larger values contribute more to variance. By scaling the data, it will give equal importance to each variable, ensuring that the PCA components are influenced by the relative variations in each variable rather than their absolute values. Scaling also helps in making the analysis less sensitive to the choice of units in which variables are measured, making the results more robust and interpretable.

Q3

```
# Perform Principal Component Analysis
pca_result <- prcomp(medals_p[, c("Gold", "Silver", "Bronze")])

# Extract the loadings (principal components)
```

```r
loadings <- pca_result$rotation[, 1:3]

# Extract the eigenvalues
eigenvalues <- pca_result$sdev[1:3]^2

# Display the principal components
print("Principal Components (PC1, PC2, PC3):")
```

```
## [1] "Principal Components (PC1, PC2, PC3):"
```

```r
print(loadings)
```

```
##               PC1        PC2        PC3
## Gold   -0.5710882 -0.8050619  0.1604172
## Silver -0.5783182  0.5332652  0.6173948
## Bronze -0.5825859  0.2598147 -0.7701233
```

```r
# Display their corresponding eigenvalues
print("Corresponding Eigenvalues:")
```

```
## [1] "Corresponding Eigenvalues:"
```

```r
print(eigenvalues)
```

```
## [1] 2.7713651 0.1450191 0.0836158
```

Q4

a

```r
# Perform Principal Component Analysis
pca_result <- prcomp(medals_p[, c("Gold", "Silver", "Bronze")])

# Calculate the amount of variance captured by the first principal component
variance_captured_pc1 <- pca_result$sdev[1]^2

# Print the amount of variance captured by PC1
print(paste("Amount of variance captured by PC1:", variance_captured_pc1))
```

```
## [1] "Amount of variance captured by PC1: 2.77136508282023"
```

```r
# Calculate the total variance of the dataset
total_variance <- sum(pca_result$sdev^2)

# Calculate the reconstruction error in terms of variance
reconstruction_error_variance <- total_variance - variance_captured_pc1

# Print the reconstruction error in terms of variance
print(paste("Reconstruction error:", reconstruction_error_variance))
```

```
## [1] "Reconstruction error: 0.228634917179774"
```

b

```r
# Perform Principal Component Analysis
pca_result <- prcomp(medals_p[, c("Gold", "Silver", "Bronze")])

# Calculate the amount of variance captured by the first two principal components
variance_captured_pc1_pc2 <- sum(pca_result$sdev[1:2]^2)

# Print the amount of variance captured by PC1 and PC2
print(paste("Amount of variance captured by PC1 and PC2:", variance_captured_pc1_pc2))
```

```
## [1] "Amount of variance captured by PC1 and PC2: 2.9163842043154"
```

```r
# Calculate the total variance of the dataset
total_variance <- sum(pca_result$sdev^2)

# Calculate the reconstruction error in terms of variance
reconstruction_error_variance <- total_variance - variance_captured_pc1_pc2

# Print the reconstruction error in terms of variance
print(paste("Reconstruction error:", reconstruction_error_variance))
```

```
## [1] "Reconstruction error: 0.0836157956846009"
```

c

```r
# Perform Principal Component Analysis
pca_result <- prcomp(medals_p[, c("Gold", "Silver", "Bronze")])

# Calculate the amount of variance captured by the first three principal components
variance_captured_pc1_pc2_pc3 <- sum(pca_result$sdev[1:3]^2)

# Print the amount of variance captured by PC1, PC2, and PC3
print(paste("Amount of variance captured by PC1, PC2, and PC3:", variance_captured_pc1_pc2_pc3))
```

```
## [1] "Amount of variance captured by PC1, PC2, and PC3: 3"
```

```r
# Calculate the total variance of the dataset
total_variance <- sum(pca_result$sdev^2)

# Calculate the reconstruction error in terms of variance
# For three components, this error should be very small or zero, as we are using all components
reconstruction_error_variance <- total_variance - variance_captured_pc1_pc2_pc3

# Print the reconstruction error in terms of variance
print(paste("Reconstruction error:", reconstruction_error_variance))
```

```
## [1] "Reconstruction error: 0"
```

Q5

```r
# Perform PCA on the scaled data to obtain the principal component vectors
pca_result_scaled <- prcomp(medals_p[, c("Gold", "Silver", "Bronze")])

# Extract the first principal component vector (b1)
b1 <- pca_result_scaled$rotation[, 1]

# Project the data onto b1
original_centered <- scale(medals_p[, c("Gold", "Silver", "Bronze")])
coor_b1 <- as.data.frame(original_centered %*% b1)

# Add country names to the coordinates table
coor_b1$Country <- medals$Country

# Rename the first column to "PC1_Coordinates"
names(coor_b1)[1] <- "PC1_Coordinates"

coor_b1
```

```
##    PC1_Coordinates           Country
## 1      -7.63399950             China
## 2      -9.04703361     UnitedStates
## 3      -4.21591208            Russia
## 4      -3.46587390     GreatBritain
## 5      -2.57977907           Germany
## 6      -3.09807286         Australia
## 7      -1.80674737       SouthKorea
## 8      -1.18965406             Japan
## 9      -1.39226939             Italy
## 10     -2.92104543            France
## 11     -0.37011789       Netherlands
## 12     -0.89679665           Ukraine
## 13     -0.36994542             Kenya
## 14     -0.73230082             Spain
## 15      0.06718114           Jamaica
## 16      0.04401515            Poland
## 17      0.43792151          Ethiopia
## 18      0.27014141           Romania
## 19     -1.73059643              Cuba
## 20     -0.82168054            Canada
## 21      0.12224305           Hungary
## 22      0.21196771            Norway
## 23     -0.49416037            Brazil
## 24     -0.22498638           Belarus
## 25      0.41475551    CzechRepublic
## 26      0.50448018          Slovakia
## 27      0.24697542       NewZealand
## 28      0.42642475           Georgia
## 29      0.22380942        Kazakhstan
## 30      0.41492798           Denmark
## 31      0.50465265        NorthKorea
```

```
## 32      0.14575399          Thailand
## 33      0.68410198          Mongolia
## 34      0.42659722       Switzerland
## 35      0.52799112         Argentina
## 36      0.70744045            Mexico
## 37      0.88688978           Belgium
## 38      0.66093598          Zimbabwe
## 39      0.58288055          Slovenia
## 40      0.50482512        Azerbaijan
## 41      0.02119409         Indonesia
## 42      0.59454979          Bulgaria
## 43      0.21231266            Turkey
## 44      0.68427445     ChineseTaipei
## 45      0.40343122           Finland
## 46      0.77399912            Latvia
## 47      0.86372378 DominicanRepublic
## 48      0.78566835           Estonia
## 49      0.78566835          Portugal
## 50      0.78566835 TrinidadandTobago
## 51      0.78566835             India
## 52      0.87539302              Iran
## 53      0.96511768          Cameroon
## 54      0.95362092            Panama
## 55      0.95362092           Tunisia
## 56      0.54804532            Sweden
## 57      0.55971456         Lithuania
## 58      0.15725075           Nigeria
## 59      0.57138379           Croatia
## 60      0.75083312          Colombia
## 61      0.51632188            Greece
## 62      0.58305302           Armenia
## 63      0.67277769        Uzbekistan
## 64      0.76250236           Austria
## 65      0.49315589           Ireland
## 66      0.49315589        Kyrgyzstan
## 67      0.49315589            Serbia
## 68      0.85222702           Algeria
## 69      0.68427445           Bahamas
## 70      0.68427445           Morocco
## 71      0.68427445        Tajikistan
## 72      0.94195169             Chile
## 73      0.87539302           Ecuador
## 74      0.87539302           Iceland
## 75      0.87539302          Malaysia
## 76      0.87539302             Samoa
## 77      0.87539302         Singapore
## 78      0.87539302       SouthAfrica
## 79      0.87539302             Sudan
## 80      0.87539302           Vietnam
## 81      0.86389625             Egypt
## 82      0.95362092       Afghanistan
## 83      0.85222702            Israel
## 84      0.85222702         Mauritius
## 85      0.85222702           Moldova
```

```
## 86      0.85222702             Togo
## 87      0.85222702        Venezuela
```

rank of nations based on the PC1 coordinates

```r
# Rank nations based on the PC1 coordinates
ranked_nations <- coor_b1 %>%
  arrange(PC1_Coordinates)

# Print the ranked list of nations
head(ranked_nations)
```

```
##   PC1_Coordinates      Country
## 1       -9.047034 UnitedStates
## 2       -7.634000        China
## 3       -4.215912       Russia
## 4       -3.465874 GreatBritain
## 5       -3.098073    Australia
## 6       -2.921045       France
```

rank of nations based on three ranking methods

```r
# Add overall rank to ranked_nations
ranked_nations$Rank_PCA1 <- seq_along(ranked_nations$PC1_Coordinates)

# Calculate ranks in the original medals data frame
medals$Rank_Gold = rank(-medals$Gold, ties.method = "min")
medals$Rank_Total = rank(-(medals$Gold + medals$Silver + medals$Bronze), ties.method = "min")

# Merge these ranks into the ranked_nations data frame and exclude the PC1_Coordinates column
ranked_nations_all <- ranked_nations %>%
  left_join(select(medals, Country, Rank_Gold, Rank_Total), by = "Country") %>%
  select(-PC1_Coordinates)

# Display the first few rows of the new data frame
head(ranked_nations_all, n=10)
```

```
##         Country Rank_PCA1 Rank_Gold Rank_Total
## 1  UnitedStates         1         2          1
## 2         China         2         1          2
## 3        Russia         3         3          3
## 4  GreatBritain         4         4          4
## 5     Australia         5         6          5
## 6        France         6        10          6
## 7       Germany         7         5          7
## 8    SouthKorea         8         7          8
## 9          Cuba         9        19          9
## 10        Italy        10         9         10
```

differences of three ranking methods

```
# Create a new dataframe with differences
result_diff <- data.frame(
  Countries = ranked_nations_all$Country,
  Diff_PCA1_Gold = abs(ranked_nations_all$Rank_PCA1 - ranked_nations_all$Rank_Gold),
  Diff_PCA1_Total = abs(ranked_nations_all$Rank_PCA1 - ranked_nations_all$Rank_Total),
  Diff_Gold_Total = abs(ranked_nations_all$Rank_Gold - ranked_nations_all$Rank_Total)
)

# Display the first few rows of the new data frame
head(result_diff)
```

```
##      Countries Diff_PCA1_Gold Diff_PCA1_Total Diff_Gold_Total
## 1 UnitedStates              1               0               1
## 2        China              1               0               1
## 3       Russia              0               0               0
## 4 GreatBritain              0               0               0
## 5    Australia              1               0               1
## 6       France              4               0               4
```

As shown above the ranking difference between 3 methods are both small. With the smallest difference between PCA1 ranking method and the total ranking method.

Q6

Collect and normalize medal data for each nation. Create an autoencoder neural network architecture with an encoder to reduce the data dimensionality and a decoder to reconstruct the original data. Train the autoencoder using the normalized medal data and minimize the reconstruction error. And using the trained encoder to obtain lower-dimensional representations. Rank nations based on their positions in this reduced space, where nations closer together may have similar medal performances(rank). It is more capable of capturing non-linear relationships in the data, which PCA, being a linear method, cannot. And also automatically learn features, prioritizing data aspects more relevant for reconstruction, which can be more meaningful for analyzing complex medal data.