# DA_HW5

Howard

2025-04-02

# Problem 1

## Prepare dataset

- Convert the 400 images into a 400 × 2576 data matrix.

```
library(png)
image_dir <- "D:/DA_HW/DAHW/ORL Faces"
image_files <- list.files(image_dir, pattern = "\\.png$", full.names = TRUE)
n <- length(image_files)
if(n != 400){
  stop("Expected 400 images, but found ", n)
}
data_matrix <- matrix(NA, nrow = n, ncol = 46 * 56)
for (i in 1:n) {
  img <- readPNG(image_files[i])
  data_matrix[i, ] <- as.vector(t(img))
}
dim(data_matrix)
```

```
## [1]  400 2576
```

- Add an additional column indicating the physical gender label (same in HW2) .

```
subject_ids <- rep(1:40, each = 10)
subject_gender <- c(0,rep(1,6),0,1,0,1,0,rep(1,19),0,rep(1,8))
gender_labels <- subject_gender[subject_ids]
final_data <- cbind(data_matrix, gender = gender_labels)
dim(final_data)
```

```
## [1]  400 2577
```

# (a) Identify the value of$\lambda$ that minimum MSE in both LASSO and Ridge regression models.

- **Using cross-validation to get $\lambda$**

```
library(glmnet)
```

```
## 載入需要的套件：Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
# Split to X and y
X <- as.matrix(final_data[,-2577])

# y as an binary
y_bin <- as.matrix(final_data[,2577])

# Set up cross-validation for LASSO
cv_lasso <- cv.glmnet(X, y_bin, alpha = 1, family = "binomial", type.measure = "mse")
lambda_min_lasso <- cv_lasso$lambda.min

# Set up cross-validation for Ridge
cv_ridge <- cv.glmnet(X, y_bin, alpha = 0, family = "binomial", type.measure = "mse")
lambda_min_ridge <- cv_ridge$lambda.min

# Print optimal lambdas
cat("Lasso optimal lambda:", lambda_min_lasso, "\n")
```

```
## Lasso optimal lambda: 0.006514476
```

```r
cat("Ridge optimal lambda:", lambda_min_ridge, "\n")
```

```
## Ridge optimal lambda: 1.061697
```

# Compare the selected features

```r
#  selected features
lasso_coef <- coef(cv_lasso, s = "lambda.min")
ridge_coef <- coef(cv_ridge, s = "lambda.min")
selected_features_lasso <- which(lasso_coef != 0)[-1]  #exclude intercept
selected_features_ridge <- which(ridge_coef != 0)[-1]
cat("Numbers of selected features from lasso:",length(selected_features_lasso),"\n\n")
```

```
## Numbers of selected features from lasso: 70
```

```r
cat("Numbers of selected features from ridge:",length(selected_features_ridge),"\n\n")
```

```
## Numbers of selected features from ridge: 2576
```

## Observation

- Ridge regression will keep all the features nonzero.
- Lasso regression will selected the important features from the full model.
- According to the HW2, stepwise selected 14 features while lasso selected 96 features.

# The model performance

```r
mse_lasso <- min(cv_lasso$cvm)
mse_ridge <- min(cv_ridge$cvm)
cat("lasso MSE:", mse_lasso,"\n")
```

```
## lasso MSE: 0.1006881
```

```r
cat("ridge MSE:", mse_ridge,"\n")
```

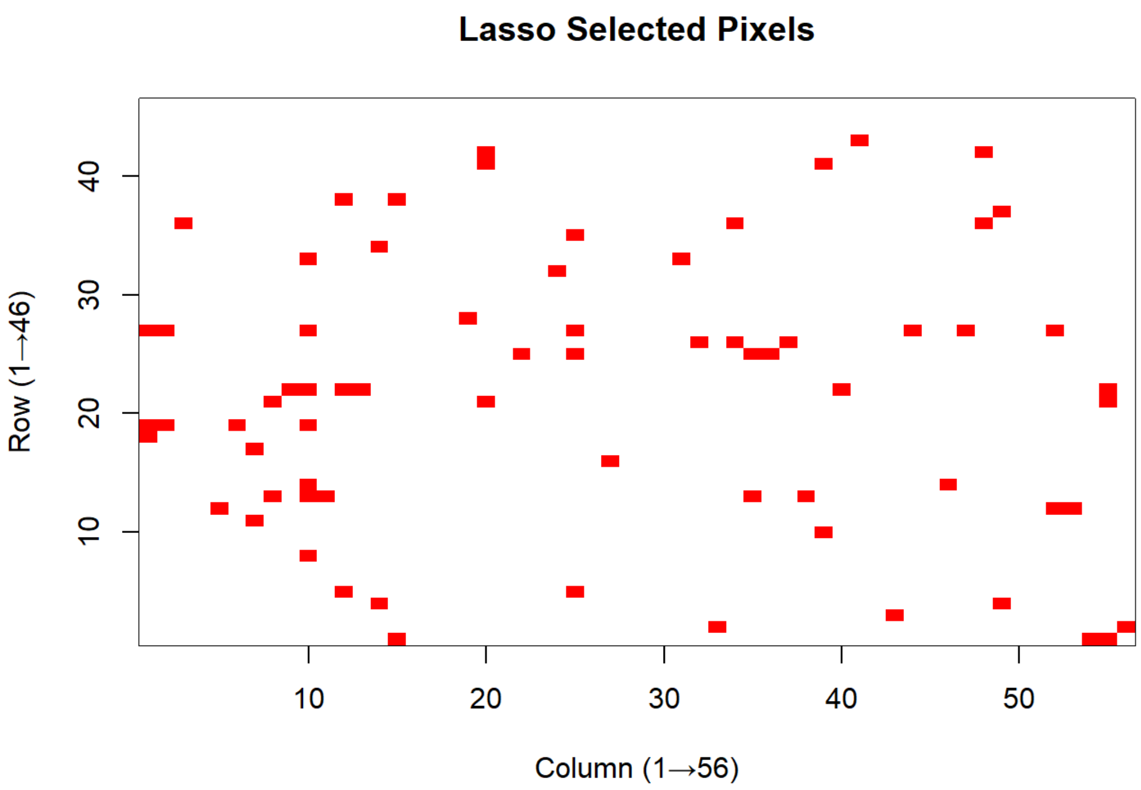
```
## ridge MSE: 0.118241
```

Observation

- The MSE of ridge and lasso is almost the same.
- Ridge MSE has a little bit higher than lasso.

## (b)Plot the chosen pixels from Lasso on $46 \times 56$ canvas.

```
# Initialize all zeros
important_pixel_map <- rep(0, 46 * 56)

# Mark selected pixel indices as 1
important_pixel_map[selected_features_lasso] <- 1

# Convert to a 46x56 matrix. We assume row-major flattening.
pixel_matrix <- matrix(important_pixel_map, nrow = 46, ncol = 56, byrow = TRUE)

# We'll use base R's image() function with two colors:
image(1:56, 1:46,
      t(apply(pixel_matrix, 2, rev)),  # flip to put row=1 at top
      col = c("white", "red"),
      xlab = "Column (1→56)",
      ylab = "Row (1→46)",
      main = "Lasso Selected Pixels")
```




## Problem 2(a): Estimate $\beta_1$ and $\beta_2$ (unconstrained Cobb-Douglas)

- The model is:

$$V_t = \alpha K_t^{\beta_1} L_t^{\beta_2} \eta_t$$

- Taking logs:

$$log(V_t) = log(\alpha) + \beta_1 log(K_t) + \beta_2 log(L_t) + log(\eta_t)$$

- So we can fit this using linear regression.

```
# Data
year <- 72:86
capital <- c(1209188, 1330372, 1157371, 1070860, 1233475, 1355769, 1351667, 1326248, 1089545, 1111942, 988165, 10696
51, 1191677, 1246536, 1281262)
labor <- c(1259142, 1371795, 1263084, 1118226, 1274345, 1369877, 1451595, 1328683, 1077207, 1056231, 947502, 105715
9, 1169442, 1195255, 1171664)
value_added <- c(11150.0, 12853.6, 10450.8, 9318.3, 12097.7, 12844.8, 13309.9, 13402.3, 8571.0, 8739.7, 8140.0, 1095
8.4, 10838.9, 10030.5, 10836.5)

# Log-transform
log_K <- log(capital)
log_L <- log(labor)
log_V <- log(value_added)

# Fit unconstrained linear model
model_unconstrained <- lm(log_V ~ log_K + log_L)
summary(model_unconstrained)
```

```
##
## Call:
## lm(formula = log_V ~ log_K + log_L)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.090518 -0.043559 -0.009005  0.022608  0.180405
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.6259     2.8996  -3.320  0.00611 **
## log_K         0.5057     0.5061   0.999  0.33743
## log_L         0.8455     0.4216   2.006  0.06799 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07495 on 12 degrees of freedom
## Multiple R-squared:  0.8184, Adjusted R-squared:  0.7881
## F-statistic: 27.03 on 2 and 12 DF,  p-value: 3.591e-05
```

# Results:

- $\hat{\beta}_1$ is 0.5057
- $\hat{\beta}_2$ is 0.8455

# Problem 2(b): Re-estimate with constraint $\beta_1$ + $\beta_2$ = 1

- Substitute

$$\log(V_t) = \log(\alpha) + \beta_1 \log(K_t) + (1 - \beta_1) \log(L_t)$$
$$\Rightarrow \log(V_t) = \log(\alpha) + \beta_1 (\log(K_t) - \log(L_t)) + \log(L_t)$$

$$Z_t = \log(K_t) - \log(L_t)$$

```
# Create transformed variable Z = log(K) - log(L)
Z <- log_K - log_L

# Fit constrained model: log(V) = a + b * Z + log(L)
model_constrained <- lm(log_V - log_L ~ Z)
summary(model_constrained)
```

```
##
## Call:
## lm(formula = log_V - log_L ~ Z)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.12097 -0.07111  0.02166  0.04319  0.14354
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -4.712885   0.020753 -227.090   <2e-16 ***
## Z            0.009609   0.441507    0.022    0.983
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08017 on 13 degrees of freedom
## Multiple R-squared:  3.643e-05,  Adjusted R-squared:  -0.07688
## F-statistic: 0.0004737 on 1 and 13 DF,  p-value: 0.983
```

```
# Recover β₁ and β₂
beta1_constrained <- coef(model_constrained)[2]
beta2_constrained <- 1 - beta1_constrained
alpha_constrained <- exp(coef(model_constrained)[1])

cat("Under beta_1 + beta_2 = 1 constraint:\n")
```

```
## Under beta_1 + beta_2 = 1 constraint:
```

```
cat("  alpha =", alpha_constrained, "\n")
```

```
##   alpha = 0.008978832
```

```
cat("  beta_1 =", beta1_constrained, "\n")
```

```
##   beta_1 = 0.009608932
```

```
cat("  beta_2 =", beta2_constrained, "\n")
```

```
##   beta_2 = 0.9903911
```

# Problem3 (a): Create a PCA function in R as following:

## **Input**:

- Data matrix `x`
- Boolean `isCorrMX` : if `TRUE` ,use correlation matrix, if `FALSE` , use covariance matrix.

## **Output**:

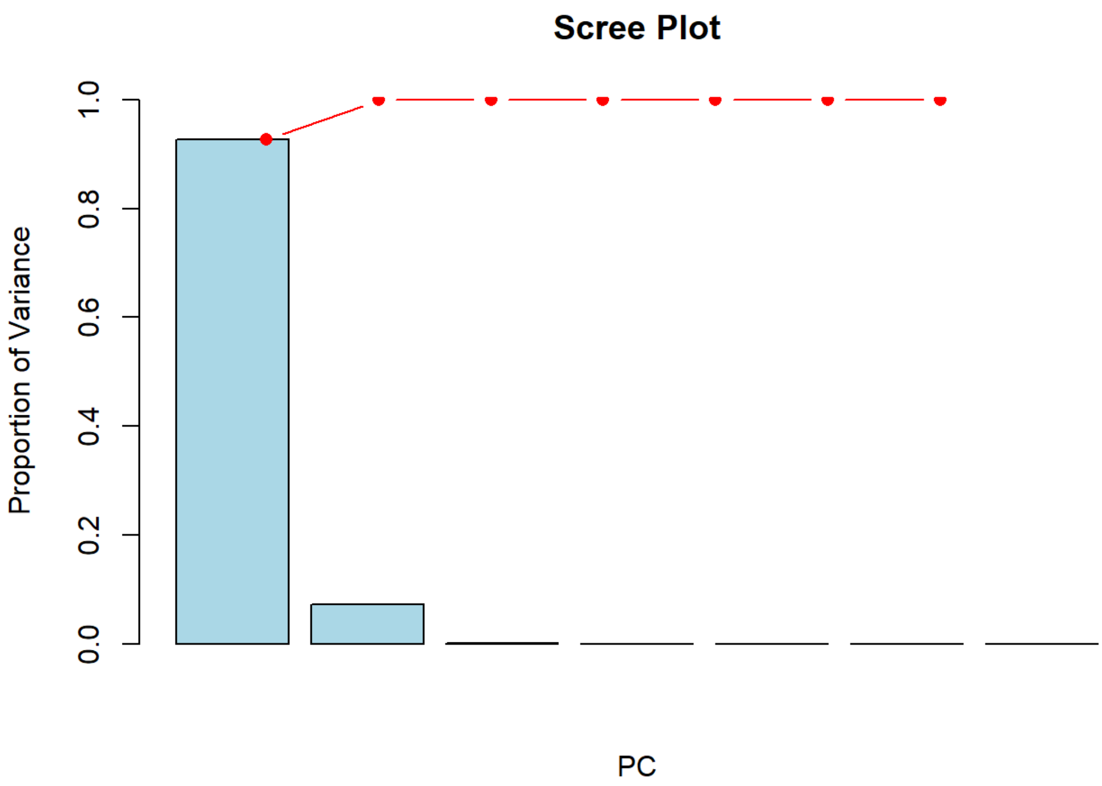
- Loading matrix (eigenvectors)
- Eigenvalue vector
- Score matrix(PCs)
- Screen plot

```r
myPCA <- function(X, isCorrMX = FALSE) {
  # Center the data
  X_centered <- scale(X, center = TRUE, scale = FALSE)

  # Use correlation or covariance matrix
  S <- if (isCorrMX) cor(X_centered) else cov(X_centered)

  # Spectral decomposition
  eig <- eigen(S)
  eig_values <- eig$values
  eig_vectors <- eig$vectors

  # Scores (projected data)
  scores <- X_centered %*% eig_vectors

  # Scree plot
  var_explained <- eig_values / sum(eig_values)
  cum_var <- cumsum(var_explained)

  barplot(var_explained, main = "Scree Plot", xlab = "PC", ylab = "Proportion of Variance", ylim = c(0,1.005), col =
"lightblue")
  lines(x = 1:length(cum_var), y = cum_var, type = "b", col = "red", pch = 16)
  return(list(
    loadings = eig_vectors,
    eigenvalues = eig_values,
    scores = scores,
    explained_variance = var_explained,
    cumulative_variance = cum_var
  ))
}
```

# Problem3 (b): Demonstrate using the AutoMPG dataset

- Choose continous features in `mtcars`

```r
# Load AutoMPG
data <- mtcars[, c("mpg", "cyl", "disp", "hp", "drat", "wt", "qsec")]

# Apply PCA with covariance matrix
pca_cov <- myPCA(data, isCorrMX = FALSE)
```




```
pca_cov$loadings
```

```
##              [,1]         [,2]        [,3]        [,4]         [,5]
## [1,]  0.038118683 -0.009182403  0.99018630 -0.06777104  0.091032234
## [2,] -0.012035357  0.003374958 -0.06293054  0.27013664  0.915542928
## [3,] -0.899595869 -0.435453668  0.03180796  0.00149054 -0.008470542
## [4,] -0.434786012  0.899788639  0.02307582 -0.02831634  0.001506989
## [5,]  0.002660185  0.003899832  0.04043028  0.03767426 -0.309878145
## [6,] -0.006239546 -0.004865696 -0.08407174 -0.17014740  0.028601267
## [7,]  0.006671139 -0.025018357 -0.07296532 -0.94406555  0.237860800
##              [,6]          [,7]
## [1,] -0.047546856  0.0534159080
## [2,] -0.195351163 -0.2157052870
## [3,]  0.003456672 -0.0031159748
## [4,]  0.002142719  0.0005991073
## [5,] -0.615423407 -0.7226017728
## [6,] -0.759760099  0.6211809371
## [7,]  0.059827262 -0.2063705962
```

```
pca_cov$eigenvalues
```

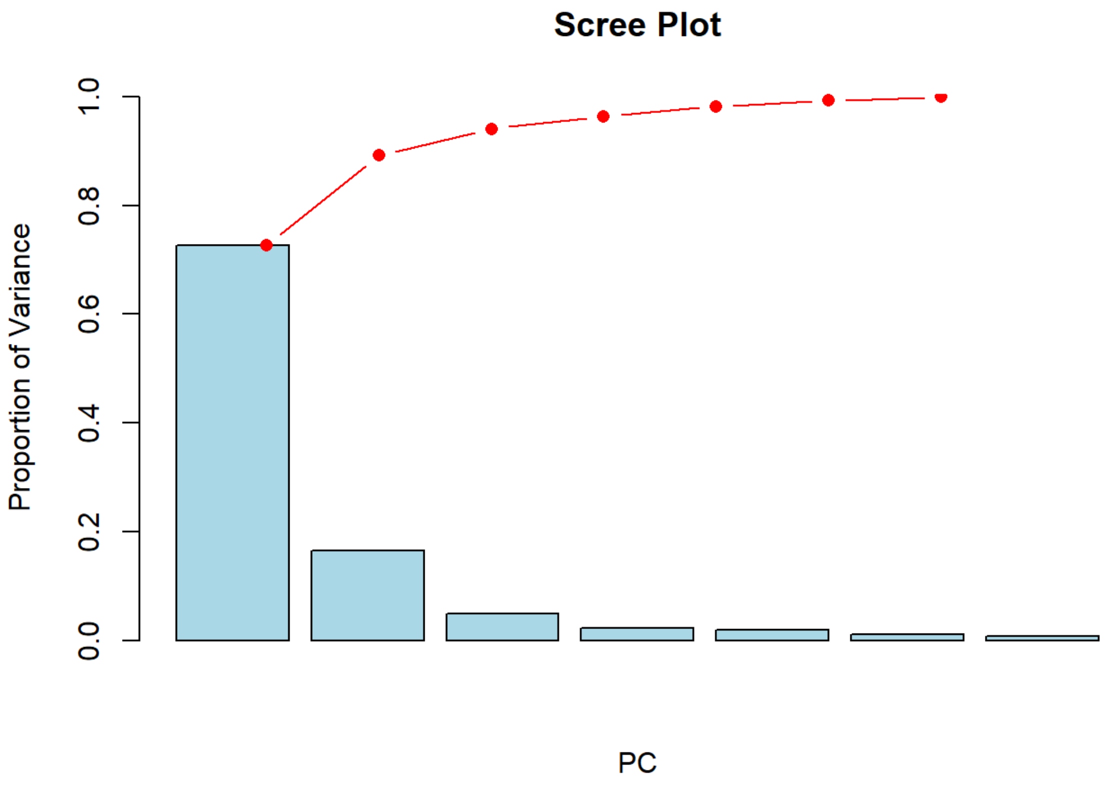
```
## [1] 1.864032e+04 1.453913e+03 9.287215e+00 1.547302e+00 3.767684e-01
## [6] 9.632947e-02 8.008942e-02
```

```
pca_cov$scores
```

```
##                            [,1]       [,2]       [,3]         [,4]        [,5]
## Mazda RX4             79.604508  -2.185045 -2.0200517  2.245285237  0.01344313
## Mazda RX4 Wag         79.606653  -2.200296 -2.0823506  1.673220944  0.15393850
## Datsun 710           133.883622   5.086336 -2.2918334  0.006313295 -0.72062110
## Hornet 4 Drive        -8.526653 -44.943825  1.1925904 -0.581196178  0.19968597
## Hornet Sportabout   -128.690876 -30.782573  3.2980561  0.702524796  0.42806449
## Valiant               23.210972 -35.064450 -3.3305143 -1.055270888  0.46297227
## Duster 360          -159.302142  32.272157  0.6341391  0.112712468 -0.16253840
## Merc 240D            112.612037 -39.713494 -0.3729508 -0.632937272 -0.54440602
## Merc 230             103.530932  -7.508062 -1.5823514 -4.190056002  0.02702953
## Merc 280              67.053960   6.169342 -3.4630474  0.134640852  0.25971920
## Merc 280C             67.004596   6.167186 -4.8930874 -0.336919020  0.27499055
## Merc 450SE           -55.208115  10.389804 -1.6261498  0.122361324  1.07264106
## Merc 450SL           -55.170353  10.378191 -0.7209908 -0.069595605  1.19241779
## Merc 450SLC          -55.248045  10.387223 -2.8337717 -0.313410010  1.09782449
## Cadillac Fleetwood  -242.811053 -52.517193 -0.8968358 -0.640083233 -0.88270619
## Lincoln Continental -236.365729 -38.290433 -1.0478970 -0.817051039 -0.82076238
## Chrysler Imperial   -224.733256 -16.112726  2.9650090 -1.163289413 -0.40598391
## Fiat 128             172.354040  -6.557351  5.6155723 -0.706234032  0.49064268
## Honda Civic          181.063169 -17.799738  3.3695791  0.849686753 -0.19320393
## Toyota Corolla       179.688451  -4.169901  6.8410068 -1.129472348  0.73851790
## Toyota Corona        121.217849   3.392122 -3.2223024 -1.352828440 -0.55179458
## Dodge Challenger     -80.162716 -34.957009 -1.7949185  1.678002929  0.54231083
## AMC Javelin          -67.575373 -28.866726 -2.5457469  1.300673985  0.61258531
## Camaro Z28          -150.347472  36.647349 -0.6444273  0.545177108 -0.42455965
## Pontiac Firebird    -164.658164 -48.208305  5.0263994  0.628392017  0.17516975
## Fiat X1-9            171.887607  -6.625607  0.6390338  0.223051855 -0.11932287
## Porsche 914-2        123.800069  -2.047783  1.3997952  1.720055078 -1.17571127
## Lotus Europa         137.075805  28.676071  5.4741583  0.654345796 -0.29439930
## Ford Pantera L      -159.413292  53.312861  2.4438242  0.830788451 -0.56427329
## Ferrari Dino          64.779716  62.867154 -2.2383697  1.340699650 -0.01717668
## Maserati Bora       -145.337438 138.970783  1.6312425 -1.388065608  0.13936792
## Volvo 142E           115.176688  13.833937 -2.9228092 -0.391523451 -1.00386179
##                             [,6]        [,7]
## Mazda RX4           -0.145743789 -0.01625996
## Mazda RX4 Wag       -0.305979348  0.02657364
## Datsun 710           0.330528872  0.06922414
## Hornet 4 Drive       0.404866456  0.04689261
## Hornet Sportabout    0.275590211 -0.21902795
## Valiant              0.494446856  0.19290457
## Duster 360           0.428496287 -0.13120571
## Merc 240D           -0.157560830  0.38471761
## Merc 230             0.031171567 -0.45211342
## Merc 280            -0.531262993 -0.01310717
## Merc 280C           -0.428801037 -0.21171180
## Merc 450SE          -0.302070804  0.29420739
## Merc 450SL          -0.074579089  0.08980607
## Merc 450SLC          0.011212208 -0.07385653
## Cadillac Fleetwood  -0.060680572  0.09179817
## Lincoln Continental -0.265583704  0.22570360
## Chrysler Imperial   -0.612485078  0.39397464
## Fiat 128            -0.283979549  0.23892019
## Honda Civic         -0.364740047 -0.64850149
## Toyota Corolla      -0.166834373 -0.07450829
## Toyota Corona        0.508642847 -0.12598074
## Dodge Challenger     0.399252213  0.08839916
## AMC Javelin          0.215413070 -0.30735639
## Camaro Z28          -0.109404694 -0.27275658
## Pontiac Firebird     0.127255262 -0.02098970
## Fiat X1-9            0.126781305 -0.08141744
## Porsche 914-2       -0.117848272  0.06387658
## Lotus Europa         0.527491729  0.43677211
## Ford Pantera L      -0.031064521 -0.71341860
## Ferrari Dino         0.004414121  0.49360236
```

```
## Maserati Bora        0.106839003  0.16138855
## Volvo 142E          -0.033783308  0.06345039
```

```
# Apply PCA with correlation matrix
pca_cor <- myPCA(data, isCorrMX = TRUE)
```




```
pca_cor$loadings
```

```
##            [,1]        [,2]       [,3]         [,4]       [,5]        [,6]
## [1,]  0.4127573 -0.08296098  0.2416477  0.766798834 -0.2127946 -0.09002238
## [2,] -0.4247315 -0.07844163  0.1880252  0.193926827  0.2383825  0.78055217
## [3,] -0.4225036  0.08239922 -0.1180359  0.587679498  0.1488730 -0.16169873
## [4,] -0.3877611 -0.33696384 -0.2027400 -0.006884691 -0.8314576  0.04607937
## [5,]  0.3311703 -0.44858845 -0.7552915  0.117073263  0.2217502  0.23319304
## [6,] -0.3913153  0.32236122 -0.4405532  0.107346915  0.1666473 -0.36612190
## [7,]  0.2399275  0.74932087 -0.2943885  0.061382684 -0.3278152  0.40735764
##             [,7]
## [1,] -0.35069828
## [2,] -0.27276548
## [3,]  0.63803744
## [4,] -0.03873894
## [5,]  0.03702814
## [6,] -0.61280387
## [7,]  0.13083413
```

```
pca_cor$eigenvalues
```

```
## [1] 5.08609988 1.15656554 0.34485150 0.15793358 0.12949405 0.07585706 0.04919838
```

```
pca_cor$scores
```

```
##                            [,1]        [,2]       [,3]      [,4]        [,5]
## Mazda RX4             44.562221   5.1049302  16.413005 -40.76210   20.160326
## Mazda RX4 Wag         44.596795   5.6067520  16.135807 -40.70035   20.019244
## Datsun 710            75.333452   8.0928702  25.593362 -70.11810   24.928028
## Hornet 4 Drive         3.532565  15.9394935   4.422079  17.28800   33.605196
## Hornet Sportabout    -67.376666   0.7364498 -20.511654  74.98510   -3.356860
## Valiant               18.037186  15.9859066   8.437661  -4.56472   33.266069
## Duster 360           -96.650189 -23.3551977 -35.521914  71.07781  -60.200807
## Merc 240D             71.603636  22.9886462  27.025315 -45.77461   55.777970
## Merc 230              61.427492  13.5723823  19.634849 -50.49535   26.895661
## Merc 280              35.694548   3.1340790  11.527312 -37.56217   10.403748
## Merc 280C             35.260645   3.6998169  11.012373 -38.59885   10.504971
## Merc 450SE           -34.871862  -7.1718573 -12.471515  23.78602  -19.597152
## Merc 450SL           -34.319347  -7.2062608 -12.163121  24.45191  -19.910890
## Merc 450SLC          -35.109732  -6.7161964 -12.810364  22.87156  -19.586815
## Cadillac Fleetwood  -130.306592   1.9463352 -42.733399 134.46171   -9.922470
## Lincoln Continental -129.197455  -2.5072956 -43.426793 127.35776  -19.926553
## Chrysler Imperial   -124.777831  -9.9948401 -43.089248 118.79202  -36.121932
## Fiat 128             102.474290  14.4827285  36.471586 -79.72312   40.721660
## Honda Civic          108.627425  17.8372102  39.076154 -82.94497   52.743461
## Toyota Corolla       106.984577  14.2107605  37.862345 -83.02879   39.931749
## Toyota Corona         68.363012   9.0129746  22.677295 -63.94761   23.212154
## Dodge Challenger     -41.454763   6.0535938 -10.955442  47.97465   11.833878
## AMC Javelin          -35.397953   5.0447507  -9.759137  39.58002    9.744851
## Camaro Z28           -92.874526 -24.5646654 -34.968317  64.49768  -61.175477
## Pontiac Firebird     -84.244897   4.1753754 -25.246651  98.91280    2.533799
## Fiat X1-9            100.209417  14.4180106  35.488320 -83.52092   41.994268
## Porsche 914-2         72.037257   7.7654241  25.523786 -60.49077   28.465885
## Lotus Europa          76.044504  -1.8454572  25.817101 -72.21014    5.169514
## Ford Pantera L       -99.426476 -32.2096540 -38.141323  66.80113  -77.060970
## Ferrari Dino          24.776965 -18.4712476   5.119313 -51.09724  -35.563273
## Maserati Bora       -106.520269 -59.6787624 -46.519448  36.28437 -143.484790
## Volvo 142E            62.962568   3.9129441  20.080662 -63.58273   13.995559
##                            [,6]        [,7]
## Mazda RX4             9.2405905  -43.774205
## Mazda RX4 Wag         9.3753497  -43.857203
## Datsun 710           16.1164263  -75.916033
## Hornet 4 Drive       -5.8370290   18.608089
## Hornet Sportabout   -18.5828347   81.119324
## Valiant              -0.2808766   -1.156083
## Duster 360          -15.4754662   79.718844
## Merc 240D             8.4965793  -50.941398
## Merc 230             12.3648734  -55.010639
## Merc 280              9.2267343  -39.058494
## Merc 280C             9.5971802  -38.489016
## Merc 450SE           -4.6248696   27.670171
## Merc 450SL           -4.4999368   27.589063
## Merc 450SLC          -4.1662528   28.347223
## Cadillac Fleetwood  -34.8864452  153.336424
## Lincoln Continental -32.5978256  145.167616
## Chrysler Imperial   -29.1401416  130.322375
## Fiat 128             19.1937378  -96.736712
## Honda Civic          19.2391731  -97.141411
## Toyota Corolla       20.5829805 -101.787989
## Toyota Corona        14.9434524  -67.806071
## Dodge Challenger    -12.8367394   56.329369
## AMC Javelin         -10.2487211   47.624842
## Camaro Z28          -13.9212129   73.486706
## Pontiac Firebird    -25.2481772  106.218620
## Fiat X1-9            19.4691707  -94.668922
## Porsche 914-2        13.1704024  -69.231041
## Lotus Europa         18.0199807  -87.318957
## Ford Pantera L      -13.4435886   72.821622
## Ferrari Dino         14.2669843  -55.634779
```

```
## Maserati Bora        -2.2792829   38.192626
## Volvo 142E           14.7657847  -68.023962
```

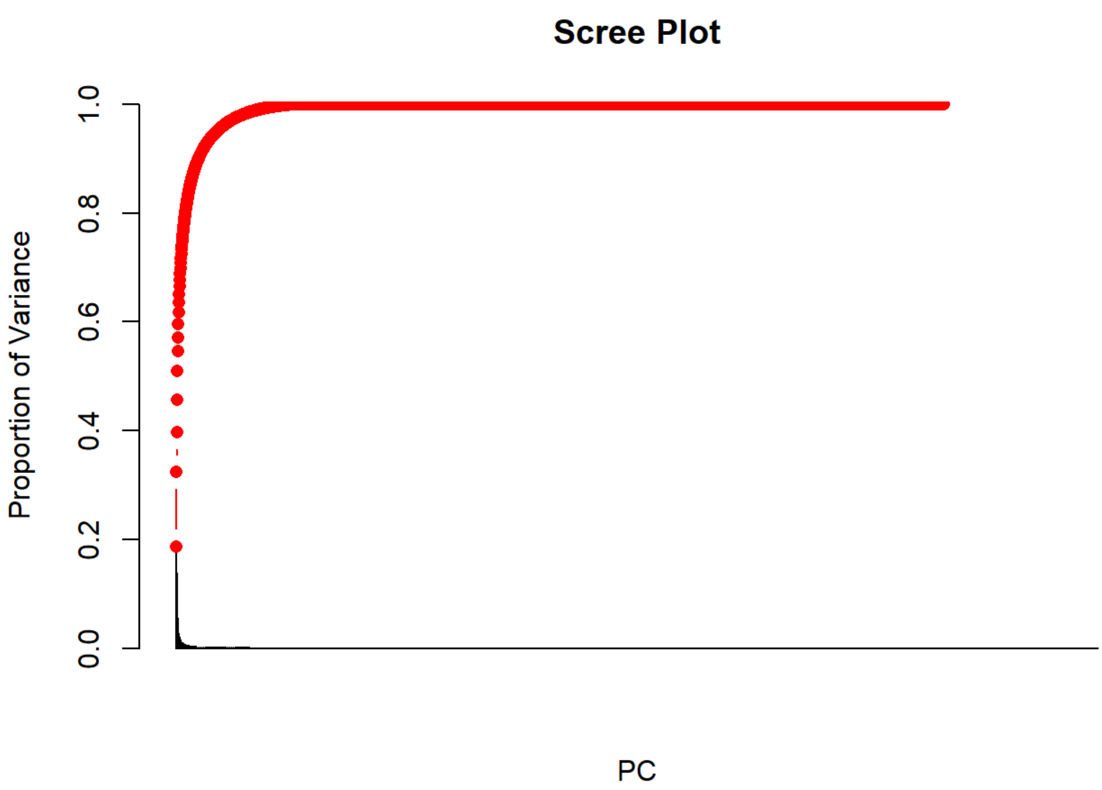
# Is PCA scale-invariant ?

- PCA is not scale-invariant when using covariance matrix, because features with large variance dominate the PCs.
- Using the correlation matrix makes PCA scale-invariant, giving equal weight to each feature.

# Problem 4 (a) :

- Run PCA, and determine the number of PCs

```
X <- t(data_matrix)

# Apply custom PCA
pca_result <- myPCA(t(X), isCorrMX = FALSE)
```




```
# Extract cumulative variance
cum_var <- pca_result$cumulative_variance

# Thresholds
thresholds <- c(0.5, 0.6, 0.7, 0.8, 0.9)

# Find number of PCs for each threshold
num_pcs_needed <- sapply(thresholds, function(th) which(cum_var >= th)[1])

# Print results
thresholds_percent <- thresholds * 100
for (i in seq_along(thresholds)) {
  cat(sprintf("To explain at least %d%% variance, need %d PCs\n", thresholds_percent[i], num_pcs_needed[i]))
}
```

```
## To explain at least 50% variance, need 5 PCs
## To explain at least 60% variance, need 9 PCs
## To explain at least 70% variance, need 16 PCs
## To explain at least 80% variance, need 32 PCs
## To explain at least 90% variance, need 76 PCs
```

# Problem 4 (b):

- Rescale the first PC to the range [0,255], reshape it into a $46 \times 56$ matrix, and visualize it as a grayscale image using the scaled PC score.

```
# Extract first loading (eigenvector)
pc1 <- pca_result$loadings[, 1]

# Scale to [0, 255]
pc1_scaled <- 255 * (pc1 - min(pc1)) / (max(pc1) - min(pc1))

# Reshape to 46 x 56
pc1_matrix <- matrix(pc1_scaled, nrow = 46, ncol = 56)
```

- Use `ggplot` to visulize

```
# Visualize grayscale
# Step 5: High-quality grayscale visualization using ggplot2
library(ggplot2)
```

```
## Warning: 套件 'ggplot2' 是用 R 版本 4.2.3 來建造的
```

```
library(reshape2)
```

```
## Warning: 套件 'reshape2' 是用 R 版本 4.2.3 來建造的
```

```
# Convert matrix to data frame for ggplot
pc1_df <- melt(pc1_matrix)
colnames(pc1_df) <- c("y", "x", "value")

# Invert y-axis to display image properly (top to bottom)
pc1_df$y <- max(pc1_df$y) - pc1_df$y + 1

# Plot with ggplot2
ggplot(pc1_df, aes(x = x, y = y, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "black", high = "white") +
  theme_void() +
  coord_fixed() +
  ggtitle("First Principal Component (PC1) - 46 × 56 Grayscale Image") +
  theme(plot.title = element_text(hjust = 0.5))
```

First Principal Component (PC1) - 46 × 56 Grayscale Image