Tutorial8

Rongqian Zhang

1 Scree plot and elbow method

In multivariate statistics, a `scree plot` is a line plot of the eigenvalues of principal components in an analysis. The scree plot is used to determine the number of principal components to keep in a principal component analysis (PCA).

```
# Load required library
library(ggplot2)
library(factoextra)
```

1.1 Generate synthetic data (20 variables, 3 clusters)

```
set.seed(123)
data <- as.data.frame(
  rbind(
    matrix(rnorm(100*20, mean = 0, sd = 1),ncol=20),    # Cluster 1 (mean=0)
    matrix(rnorm(100*20, mean = 1, sd = 1),ncol=20),    # Cluster 2 (mean=1)
    matrix(rnorm(100*20, mean = -1, sd = 1),ncol=20)   # Cluster 3 (mean=-1)
  )
)
df <- scale(data) # Standardize the data (mean=0, variance=1)
```

1.2 Perform PCA

```
pca_result <- prcomp(df,scale= FALSE)
```

1.3 Calculate variance explained by each PC

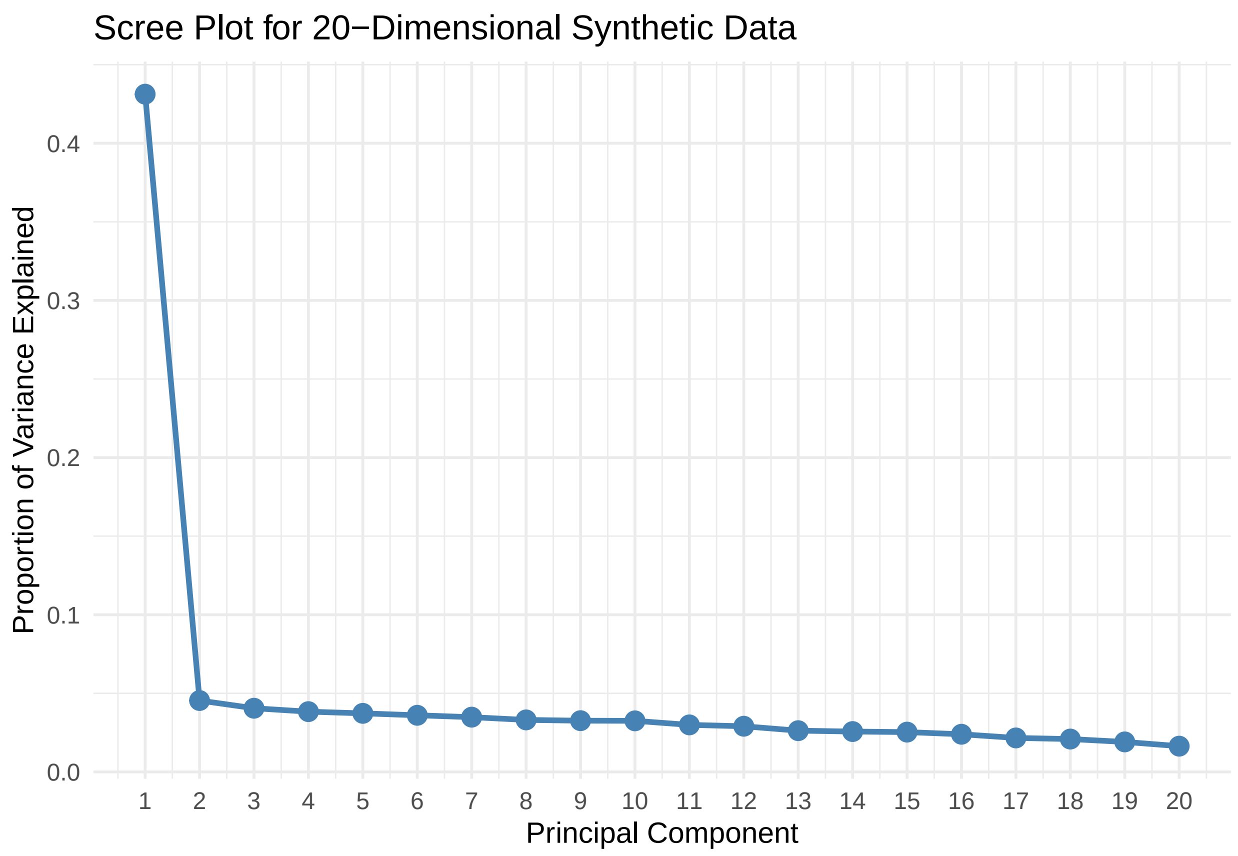
```
var_explained <- pca_result$sdev^2 / sum(pca_result$sdev^2)

# Create a scree plot (elbow plot)
scree_data <- data.frame(
  PC = 1:length(var_explained),
  Variance = var_explained
)
```

1.4 Plot scree plot

```r
ggplot(scree_data, aes(x = PC, y = Variance)) +
  geom_point(size = 3, color = "steelblue") +
  geom_line(linewidth = 1, color = "steelblue") +

  labs(title = "Scree Plot for 20-Dimensional Synthetic Data",
       x = "Principal Component",
       y = "Proportion of Variance Explained") +
  scale_x_continuous(breaks = 1:20) +
  theme_minimal()
```

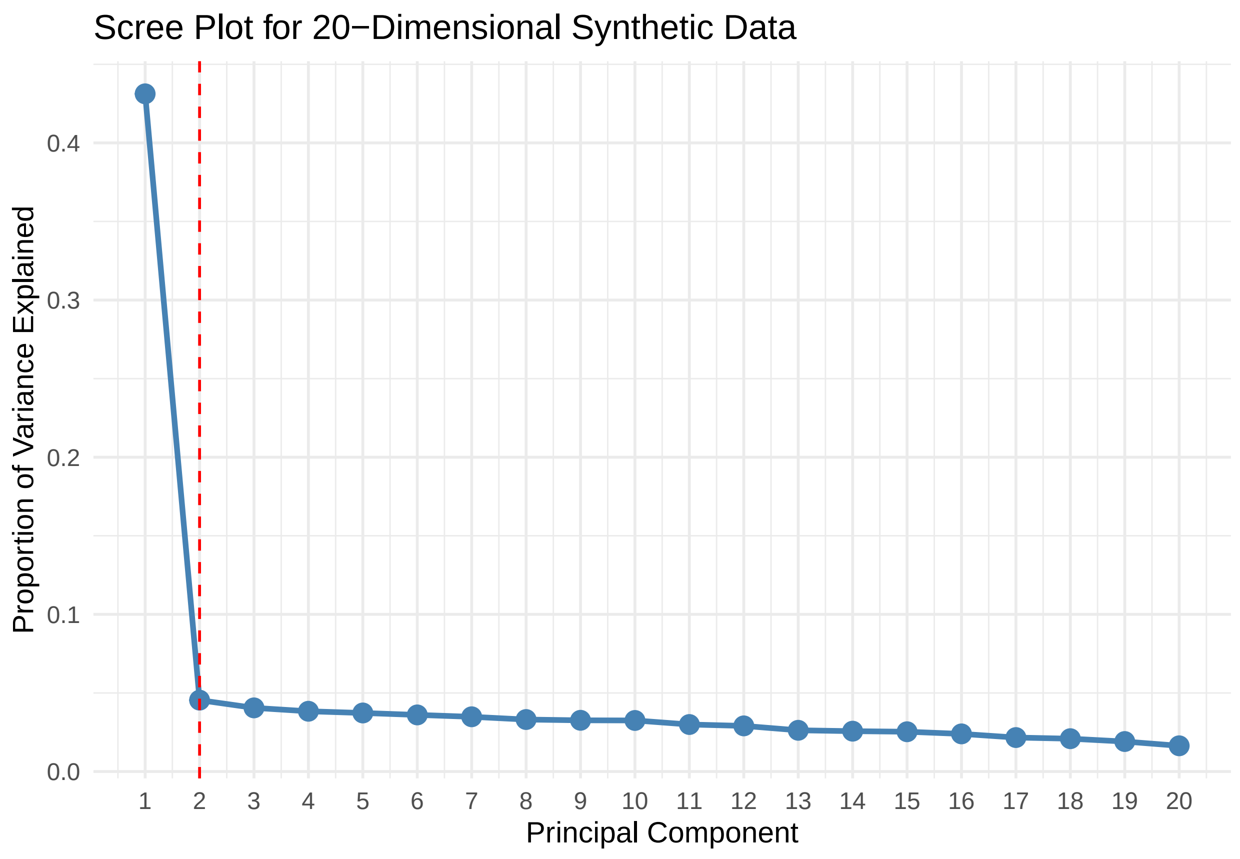


The "elbow point," where the slope of the curve flattens, indicates the optimal number of components r.
$r = \text{argmin}_k \frac{\lambda_k}{\lambda_{k-1}}$ for $k \geq 2$.

```r
print(which.min(var_explained[2:20]/var_explained[1:19])+1)
```

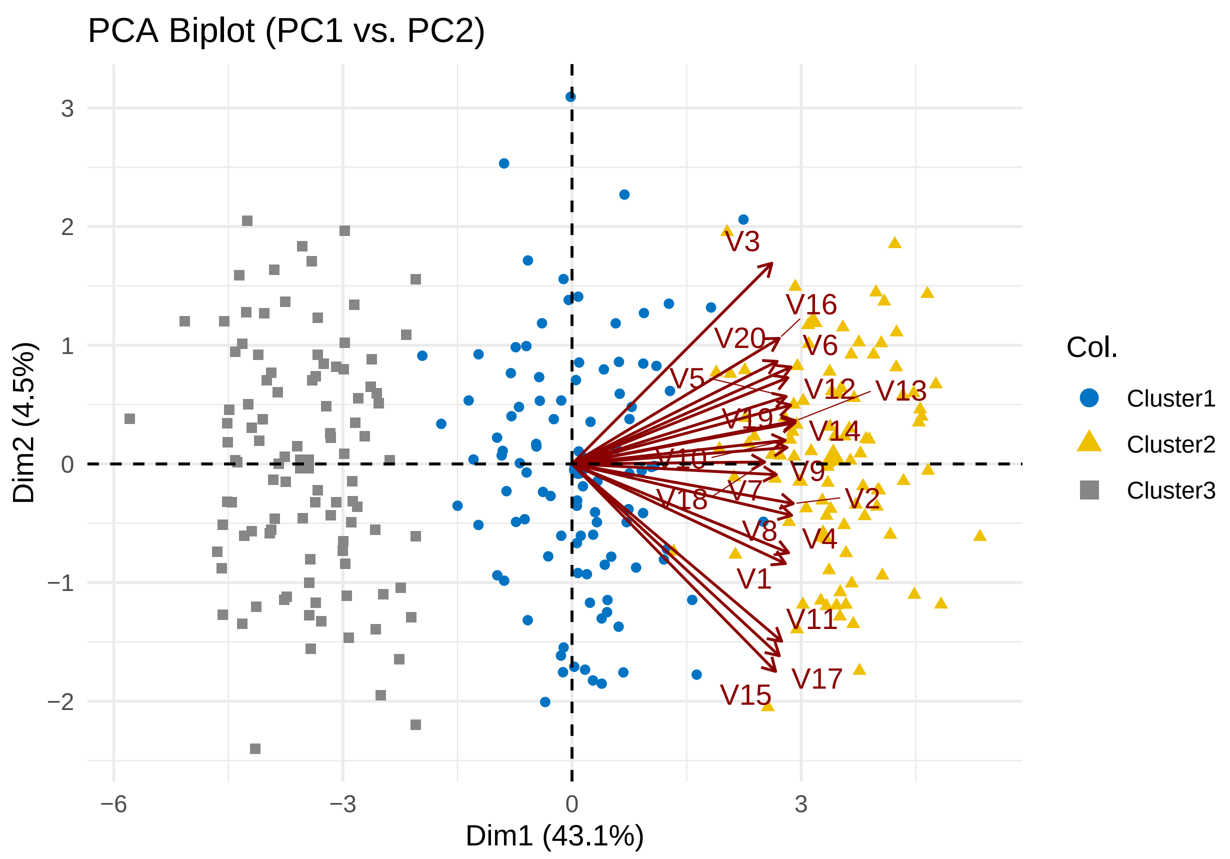
```
## [1] 2
```

```r
# Plot scree plot (elbow method)
ggplot(scree_data, aes(x = PC, y = Variance)) +
  geom_point(size = 3, color = "steelblue") +
  geom_line(linewidth  = 1, color = "steelblue") +
  geom_vline(xintercept = 2, linetype = "dashed", color = "red") +  # Suggested elbow
  labs(title = "Scree Plot for 20-Dimensional Synthetic Data",
       x = "Principal Component",
       y = "Proportion of Variance Explained") +
  scale_x_continuous(breaks = 1:20) +
  theme_minimal()
```



1.5 Check the biplot

```
fviz_pca_biplot(pca_result,
                axes = c(1, 2),             # Focus on PC1 and PC2
                geom = "point",             # Show points (observations)
                col.var = "darkred",        # Color for variable arrows
                 col.ind = rep(c("Cluster1", "Cluster2", "Cluster3"), each = 100),
                palette = "jco",  # Use journal color palette
                repel = TRUE,               # Prevent label overlap
                title = "PCA Biplot (PC1 vs. PC2)") +
  theme_minimal()
```



2 Probabilistic PCA

`Probabilistic PCA` is a linear latent variable model that generates data by sampling latent variables from a standard normal distribution and using them to generate observed data with added noise.

In probabilistic PCA (pPCA), each observed data vector $\mathbf{x}_n \in \mathbb{R}^D$ is assumed to arise from a low-dimensional latent variable $\mathbf{z}_n \in \mathbb{R}^d$ via

$$\mathbf{z}_n \sim \mathcal{N}\big(\mathbf{0}, \mathbf{I}_d\big), \quad \mathbf{x}_n = \boldsymbol{\mu} + \mathbf{W}\,\mathbf{z}_n + \boldsymbol{\epsilon}_n, \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}\big(\mathbf{0}, \sigma^2\,\mathbf{I}_D\big),$$

where

- $\mathbf{W}$ is a $D \times d$ loading matrix,
- $\boldsymbol{\mu} \in \mathbb{R}^D$ is the mean,
- σ^2 is an isotropic noise variance,
- d is the latent dimension (with $d < D$).

Hence, marginalizing out the latent variable $\mathbf{z}_n$ gives

$$\mathbf{x}_n \sim \mathcal{N}\Big(\boldsymbol{\mu}, \mathbf{W}\,\mathbf{W}^\top + \sigma^2\,\mathbf{I}_D\Big).$$

Notationally, we also let N denote the total number of observations $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$.

2.1 Generating synthetic data

```r
generate_ppca_data <- function(N = 1000,
                               D = 10,
                               d = 2,
                               sigma_sq = 0.1,
                               mu = NULL,
                               seed = NULL) {
  # Optionally set seed for reproducibility
  if (!is.null(seed)) set.seed(seed)

  # Generate W ~ N(0,1)
  # W is D x d
  W <- matrix(rnorm(D * d, mean = 0, sd = 1), nrow = D, ncol = d)

  # Generate latent Z ~ N(0,I)
  # Z is N x d
  Z <- matrix(rnorm(N * d, mean = 0, sd = 1), nrow = N, ncol = d)

  # Generate noise Epsilon ~ N(0, sigma^2 I)
  # Epsilon is N x D
  Epsilon <- matrix(rnorm(N * D, mean = 0, sd = sqrt(sigma_sq)),
                    nrow = N, ncol = D)

  # Mean vector mu (default is 0 if not provided)
  if (is.null(mu)) {
    mu <- rep(0, D)
  }

  # Construct X: X[n, ] = mu + W z_n + epsilon
  # Z %*% t(W) => N x D
  # Add mu to each row
  X <- Z %*% t(W) + matrix(mu, nrow = N, ncol = D, byrow = TRUE) + Epsilon

  return(list(X = X, Z = Z, W = W, mu = mu))
}
```

```r
# Example:
data_sim <- generate_ppca_data(N = 1000, D = 10, d = 2, sigma_sq = 0.1, seed = 123)
X_true <- data_sim$X     # Synthetic observations (N x D)
Z_true <- data_sim$Z     # Latent factors (N x d)
W_true <- data_sim$W     # True loading matrix (D x d)
mu_true <- data_sim$mu   # True mean (length D)
```

2.2 Fitting pPCA by Closed-Form

We have N observations $\{\mathbf{x}_n\}_{n=1}^{N}$, each $\mathbf{x}_n \in \mathbb{R}^D$. For probabilistic PCA, assume

$$\mathbf{x}_n \;\sim\; \mathcal{N}\Big(\boldsymbol{\mu},\, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}_D\Big),$$

where $\boldsymbol{\mu} \in \mathbb{R}^D$ is the mean, $\mathbf{W}$ is $D \times d$ with $d < D$, and σ^2 is isotropic noise variance. The maximum-likelihood estimates can be obtained via the sample covariance:

1. **Center the data.** Let $\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n$ and define $\mathbf{X}_{\text{centered}} \in \mathbb{R}^{N\times D}$ by
$$\mathbf{X}_{\text{centered}}(n,:) \;=\; \mathbf{x}_n^\top \;-\; \widehat{\boldsymbol{\mu}}^\top.$$
2. **Compute the sample covariance.**
$$\widehat{\mathbf{S}} \;=\; \frac{1}{N-1}\,\mathbf{X}_{\text{centered}}^\top\,\mathbf{X}_{\text{centered}} \;\in\; \mathbb{R}^{D\times D}.$$
3. **Eigen decomposition of $\widehat{\mathbf{S}}$.**
$$\widehat{\mathbf{S}} \;=\; \mathbf{V}\,\boldsymbol{\Lambda}\,\mathbf{V}^\top,$$
where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1,\dots,\lambda_D)$ with $\lambda_1 \ge \cdots \ge \lambda_D \ge 0$, and $\mathbf{V}$ is orthonormal $(D\times D)$.
4. **Select the top d components.** Let
$$\mathbf{V}_d \;=\; [\,\mathbf{v}_1,\dots,\mathbf{v}_d\,], \quad \boldsymbol{\Lambda}_d \;=\; \operatorname{diag}(\lambda_1,\dots,\lambda_d).$$
Here $\mathbf{v}_i$ is the eigenvector for λ_i.
5. **Estimate the noise variance σ^2.** For $d < D$, the MLE is given by
$$\widehat{\sigma}^2 \;=\; \frac{1}{D-d}\sum_{i=d+1}^{D} \lambda_i.$$
6. **Estimate the loading matrix W.** A simple form (often used) is
$$\widehat{\mathbf{W}} \;=\; \mathbf{V}_d\,\operatorname{diag}\!\Big(\sqrt{\lambda_1},\dots,\sqrt{\lambda_d}\Big).$$
In the MLE, one includes a "shrinkage" term to account for $\widehat{\sigma}^2$, i.e.,
$$\widehat{\mathbf{W}} \;=\; \mathbf{V}_d\,\operatorname{diag}\!\Big(\sqrt{\lambda_i - \widehat{\sigma}^2}\Big)\,\mathbf{R},$$
where $\mathbf{R}$ is any $d\times d$ orthonormal matrix (it does not affect the likelihood).

Hence, fitting pPCA with latent dimension d amounts to taking the top d eigenvalues/eigenvectors of the sample covariance, then deducing the noise variance from the remaining eigenvalues.

```r
fit_ppca_via_eig <- function(X, d) {
  # X: N x D data matrix
  # d: latent dimension

  N <- nrow(X)
  D <- ncol(X)

  # 1. Center the data
  mu_hat <- colMeans(X)
  X_centered <- sweep(X, 2, mu_hat, FUN = "-")

  # 2. Compute sample covariance (here using 1/N-1)
  S <- (t(X_centered) %*% X_centered) / (N - 1)  # D x D

  # 3. Eigen decomposition
  eig_res <- eigen(S, symmetric = TRUE)
  # eig_res$values -> eigenvalues (largest first if 'decreasing=TRUE')
  # eig_res$vectors -> columns are eigenvectors

  lambdas <- eig_res$values
  V <- eig_res$vectors

  # 4. Top d eigenvalues & eigenvectors
  lambda_d <- lambdas[1:d]        # largest d eigenvalues
  V_d <- V[, 1:d, drop = FALSE]   # D x d

  # 5. Estimate sigma^2 as average leftover
  if (d < D) {
    sigma_sq_hat <- mean(lambdas[(d+1):D])
  } else {
    sigma_sq_hat <- 0
  }

  # 6. Option A (with "shrink" for sigma^2)
  # W_hat = V_d * diag( sqrt( lambda_d - sigma_sq_hat ) )
  # but we must ensure that (lambda_d - sigma_sq_hat) is non-negative
  # if sigma_sq_hat is large, consider the simpler version below.

  # For simplicity, do the simpler approximation ignoring the shrink:
  W_hat <- V_d %*% diag(sqrt(lambda_d))

  return(list(
    W_hat = W_hat,           # D x d
    mu_hat = mu_hat,         # length D
    sigma_sq_hat = sigma_sq_hat  # scalar
  ))
}
```

```
set.seed(1)

data_sim <- generate_ppca_data(N=500, D=5, d=2, sigma_sq=0.05)
X <- data_sim$X

# Fit pPCA with d=2
fit_eig <- fit_ppca_via_eig(X, d=2)


cat("\nTrue sigma^2 =", 0.05,
    " vs. fitted =", round(fit_eig$sigma_sq_hat, 4), "\n")
```

```
##
## True sigma^2 = 0.05  vs. fitted = 0.0549
```

```
cat("Dimension of W_hat:", dim(fit_eig$W_hat), "\n")
```

```
## Dimension of W_hat: 5 2
```

```
# Compare W_true and W_hat?
# They might differ by an orthonormal rotation
```