Lecture 19

PCA/PCR/Factor Analysis

Let's start with Factor Analysis. **Factor analysis** is a statistical method used primarily for data reduction, identifying underlying relationships between measured variables, and modeling latent (unobserved) variables. It is widely used in fields like psychology, social sciences, finance, and more, where researchers are interested in understanding the structure of a set of variables and reducing data complexity.

Key Concepts:

- *Latent Variables*: Latent variables are not directly observed but are inferred from the observed variables (also called manifest variables). In factor analysis, these latent variables are called "factors."

- *Factors:* Factors are underlying constructs that explain the correlations among observed variables. Each factor is a linear combination of the observed variables, and the goal of factor analysis is to determine these factors.

- *Factor Loadings*: Factor loadings represent the correlation or relationship between observed variables and the latent factors. A higher factor loading indicates that the variable is more strongly related to the factor.

- *Communality*: Communality represents the proportion of a variable's variance that is explained by the extracted factors. High communality indicates that the factors explain most of the variance in a variable.

- *Specific Variance*: This is the part of the variance in an observed variable that is not explained by the common factors, including unique variance and error variance.

- *Eigenvalues and Scree Plot*: Eigenvalues represent the amount of variance accounted for by each factor. A scree plot is used to visualize eigenvalues and help decide how many factors to retain by identifying an "elbow" in the plot, where the eigenvalues drop off.

Types of Factor Analysis:

- ***Exploratory Factor Analysis (EFA)****:* EFA is used when you do not have a predetermined idea of the underlying factor structure. It is used to explore the data, identify potential factors, and understand the relationships among variables.
- ***Confirmatory Factor Analysis (CFA):*** CFA is used when you have a specific hypothesis or model about the factor structure, and you want to test whether the data fits this model. CFA requires a more rigorous theoretical framework than EFA.

Steps in Factor Analysis:

1. *Data Collection:* Gather a large sample of observed variables. The data should be suitable for factor analysis, with enough correlations among variables to justify the method.

2. *Factor Extraction:* Determine the initial factors. Common methods include Principal Component Analysis (PCA) or Maximum Likelihood Estimation (MLE).

3. *Determine the Number of Factors:* Decide how many factors to retain based on criteria such as eigenvalues, the scree plot, or the proportion of variance explained.

4. *Factor Rotation:* Apply a rotation method (e.g., Varimax, Promax) to make the factor structure more interpretable. Rotation redistributes the variance across factors without changing the total variance explained.

5. *Interpretation:* Interpret the factor loadings to understand the meaning of each factor. Label the factors based on the variables that load highly on them.

6. *Model Fit (CFA):* In CFA, assess the fit of the model using goodness-of-fit indices such as the Chi-square test, Root Mean Square Error of Approximation (RMSEA), and Comparative Fit Index (CFI).

Pros and Cons of Factor Analysis:
Pros:
- *Data Reduction:* Reduces the dimensionality of data by identifying key underlying factors, simplifying analysis and interpretation.
- *Insight into Structure:* Helps in understanding the structure and relationships among variables, which is useful in scale development and validation.
- *Identification of Latent Constructs*: Useful in identifying underlying constructs that may not be directly measurable.

Cons:
- *Subjectivity:* The interpretation of factors can be subjective, and different analysts may reach different conclusions.
- *Requires Large Samples:* Factor analysis typically requires large samples to produce stable and reliable results.
- *Assumptions:* Factor analysis assumes linear relationships among variables and normally distributed data, which may not always hold.
- *Sensitivity to Model Specification:* Incorrectly specifying the number of factors or the rotation method can lead to misleading results.

Applications of Factor Analysis:
- *Psychometrics:* Used to develop and validate psychological tests and questionnaires by identifying the underlying constructs (e.g., intelligence, personality traits).
- *Marketing:* Helps in understanding consumer preferences and behaviors by identifying key factors that influence purchasing decisions.
- *Finance:* Used to identify underlying factors that explain asset returns, aiding in portfolio management and risk analysis.
- *Social Sciences:* Used to uncover hidden patterns in social behavior, attitudes, and opinions.

Factor analysis is a powerful tool for uncovering the structure of complex datasets, but it requires careful application and interpretation to avoid misleading results.

Factor Analysis (FA) and Principal Component Analysis (PCA) are both statistical techniques used for data reduction and uncovering the underlying structure of a dataset. However, they have different goals, underlying assumptions, and interpretations, which make them distinct from one another.

Similarities between Factor Analysis and PCA:

1. *Dimensionality Reduction*: Both techniques aim to reduce the number of variables in a dataset by identifying a smaller number of underlying components or factors.

2. *Linear Combinations*: Both FA and PCA create new variables as linear combinations of the original variables.

3. *Variance Explanation*: Both techniques explain the variance in the original data through the identified components (in PCA) or factors (in FA).

Differences between Factor Analysis and PCA:

1. ***Objective:***
   - *PCA*: The primary goal of PCA is to explain as much of the total variance in the data as possible with a few components (principal components). It transforms the original variables into a new set of uncorrelated variables (components) that are ordered by the amount of variance they explain.
   - *FA*: The goal of FA is to identify the underlying latent factors that are responsible for the observed correlations among variables. It assumes that the observed variables are influenced by these latent factors and some unique variance (including error).

2. ***Model Structure:***
   - *PCA*: PCA does not assume any underlying structure in the data. The components are purely mathematical constructs that maximize the explained variance.
   - *FA*: FA is based on a model where each observed variable is expressed as a linear combination of underlying latent factors and unique factors (which include measurement errors). The focus is on explaining the shared variance (commonality) among variables, not the total variance.

3. ***Variance Consideration:***
   - *PCA*: PCA accounts for all the variance in the data, including both shared and unique variance. The components are derived from the covariance matrix of the original data.
   - *FA*: FA primarily focuses on the shared variance among the variables (common factors). Unique variances (specific factors and errors) are separated from the common factors, and the analysis is often based on the correlation matrix.

4. ***Interpretation:***
   - *PCA*: The components in PCA are interpreted as linear combinations of the observed variables that capture the most variance. Each component is orthogonal (uncorrelated) to the others.
   - *FA*: The factors in FA are interpreted as latent constructs that explain the correlations among the observed variables. The factors are often assumed to be correlated (oblique rotation) or uncorrelated (orthogonal rotation).

5. ***Rotation:***
   - *PCA*: After extracting the components, rotation methods (e.g., Varimax) can be applied to make the components more interpretable, though rotation is not essential to PCA.
   - *FA*: Rotation is commonly used in FA to achieve a more interpretable factor solution, whether orthogonal (e.g., Varimax) or oblique (e.g., Promax) rotation.

6. ***Eigenvalues and Scree Plot:***
   - *PCA*: Eigenvalues represent the amount of variance explained by each component. A scree plot can be used to determine the number of components to retain.
   - *FA*: Eigenvalues in FA are used similarly, but they represent the variance explained by the factors, with a focus on the common variance among variables.

7. ***Applications:***
   - *PCA*: PCA is commonly used for data compression, noise reduction, and exploratory data analysis when the primary interest is in reducing the dimensionality of the data.
   - *FA*: FA is often used in psychology, social sciences, and other fields where the interest is in identifying underlying latent constructs (e.g., intelligence, personality traits) that explain the relationships among observed variables.

Example:

- Suppose you have data on several observed variables (e.g., responses to a survey measuring different aspects of customer satisfaction). **PCA** might be used to reduce these variables to a few principal components that capture most of the variability in the data, without necessarily interpreting what each component represents. On the other hand, **FA** would attempt to identify latent factors (e.g., "service quality," "product satisfaction") that explain why these variables are correlated.

When to Use Which:

- *PCA* is typically used when the goal is to reduce the dimensionality of the data and retain as much total variance as possible.
- *FA* is more appropriate when the interest lies in identifying and understanding the latent constructs that explain the patterns of correlations among observed variables.

In summary, while PCA and FA are related in that they both aim to reduce the dimensionality of the data, their goals, interpretations, and methods differ significantly. PCA is more about data reduction for explaining variance, while FA is about understanding the underlying latent structure that causes the observed correlations.

Principal Component Analysis (PCA) is a statistical technique used primarily for dimensionality reduction and data exploration. Here's an overview of how PCA works:

1. Standardization:

- *Objective:* Ensure that all variables contribute equally to the analysis, regardless of their units or scales.
- *Process:* Subtract the mean and divide by the standard deviation for each variable to get standardized variables (with mean = 0 and standard deviation = 1).

$$Standardized\ Variable\ x_i = \frac{x_i - \mu_i}{\sigma_i}$$

  where x_i is the original value, μ_i is the mean, and σ_i is the standard deviation.

2. Covariance Matrix Computation:

- *Objective:* Identify how the variables vary together.
- *Process:* Compute the covariance matrix, which is a square matrix that shows the covariances between pairs of variables.

$$Cov(X) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$$

where x_i is the standardized data matrix.

3. Eigenvalues and Eigenvectors:

- *Objective:* Determine the principal components by identifying the directions (eigenvectors) that maximize variance.
- *Process:* Calculate the eigenvalues and eigenvectors of the covariance matrix. Eigenvectors represent the directions of the principal components, while eigenvalues represent the magnitude of variance captured by each principal component.

$$Cov(X)v = \lambda v$$

where v is the eigenvector and λ is the corresponding eigenvalue.

4. Sorting and Selecting Principal Components:

- *Objective:* Reduce the dimensionality by selecting only the most important components.
- *Process:* Sort the eigenvalues and corresponding eigenvectors in descending order. Select the top k eigenvectors based on the largest eigenvalues, where k is the number of components you want to retain.
- The variance explained by each principal component is proportional to its eigenvalue.

5. Transformation:

- *Objective:* Project the data onto the new lower-dimensional space.
- *Process:* Multiply the original standardized data by the selected eigenvectors to get the principal component scores, which are the coordinates of the data in the new space.

$$PCA\ Scores = XV_k$$

where V_k is the matrix of the top k eigenvectors.

Principal Component Regression (PCR) and Its Relationship to PCA

Principal Component Regression (PCR) is a regression technique that uses PCA as a preprocessing step to address issues related to multicollinearity and high-dimensionality in regression models.

How PCR Works:

- *Apply PCA to the Independent Variables (Predictors):* Perform PCA on the predictor variables to extract the principal components. This reduces the predictors to a set of uncorrelated components while capturing most of the variance in the original data.

- *Select Principal Components:* Choose the number of principal components to retain. This can be done by looking at the explained variance ratio and selecting enough components to capture a substantial proportion of the total variance.

- *Fit the Regression Model:* Use the selected principal components as the predictors in a linear regression model instead of the original variables.

$$Response = \beta_0 + \beta_1 PC_1 + \beta_2 PC_2 + \cdots + \beta_k PC_k + \varepsilon$$

where PC_i are the principal components.

- *Make Predictions:* Use the regression model to make predictions on new data by first projecting the new data onto the principal components and then applying the regression model.

Relationship between PCA and PCR:

- *Dimensionality Reduction:* PCA is used in PCR to reduce the dimensionality of the predictor space, which helps to alleviate problems related to multicollinearity, where predictors are highly correlated.

- *Regression Model:* PCR leverages the uncorrelated nature of principal components to build a more stable regression model, especially when dealing with high-dimensional data or when the number of predictors is larger than the number of observations.

- *Interpretability:* A downside is that the principal components may not have a direct interpretability related to the original variables, making it harder to understand the impact of each predictor on the response variable in the final model.

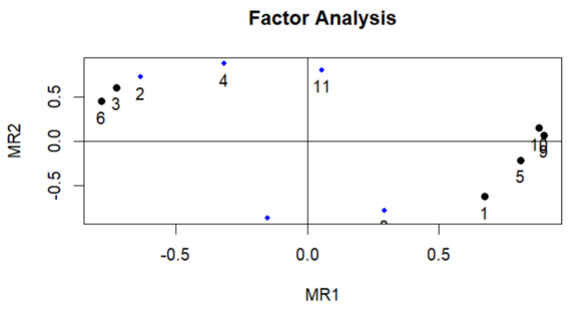
- *Variance Explanation:* The effectiveness of PCR depends on how well the selected principal components explain the variance in the response variable. If the components that explain most of the variance in the predictors do not explain much of the variance in the response variable, PCR might not perform well.

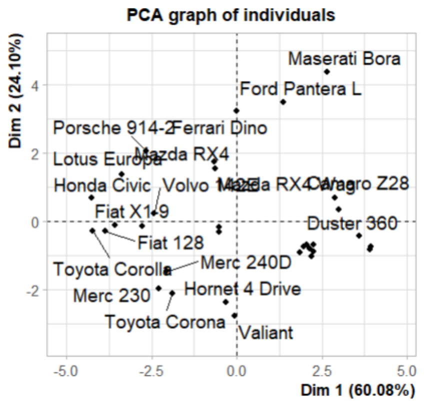
In summary, **PCA** is a technique for dimensionality reduction by finding new variables (principal components) that capture the most variance in the data. **PCR** combines PCA with regression by using these principal components as predictors in a regression model, which helps in handling multicollinearity and improving the model's stability.

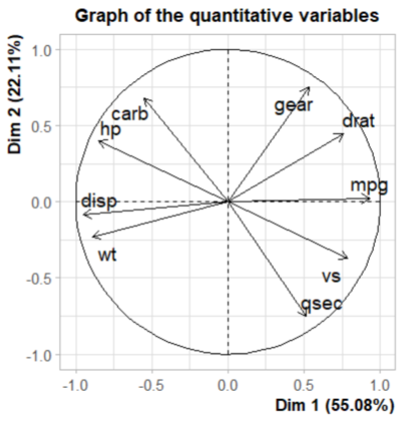
Before we look at the algorithms for each of these, let's look at what packages can be used (and which we'll be able to compare our algorithms to).

```
#factor analysis
library(psych)
fa_result <- fa(mtcars, nfactors = 2, rotate = "varimax")
print(fa_result)

library(FactoMineR)
pca_result <- PCA(mtcars, graph = FALSE)
plot(pca_result)

# Convert some variables to factors to create mixed data
mtcars$am <- as.factor(mtcars$am)
mtcars$cyl <- as.factor(mtcars$cyl)

# Perform FAMD
library(FactoMineR)
famd_result <- FAMD(mtcars, ncp = 5)
plot(famd_result)


library(lavaan)
```

```
# Use built-in dataset for demonstration
data("HolzingerSwineford1939")

# Model specification
model <- '
  # latent variables
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9
'

# Fit the model
fit <- lavaan::sem(model, data = HolzingerSwineford1939)

# Summary of the model
summary(fit, fit.measures = TRUE)
```

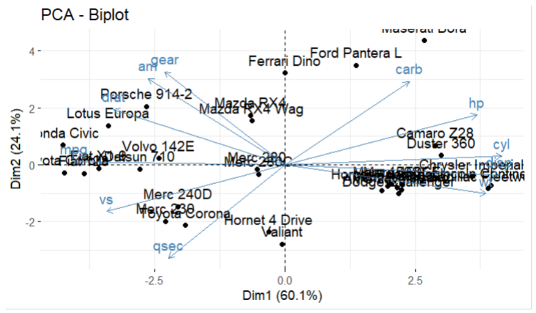






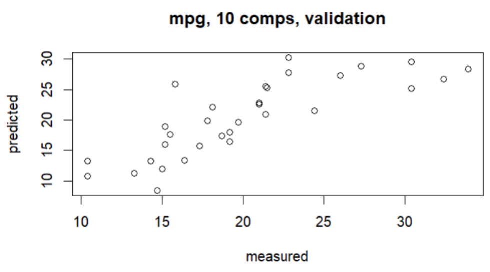
```
#PCA
data(mtcars)
pca_result <- prcomp(mtcars, scale. = TRUE)
summary(pca_result)

library(FactoMineR)
pca_result <- PCA(mtcars, graph = FALSE)
plot(pca_result)
```

```
library(factoextra)
fviz_pca_biplot(pca_result)
```



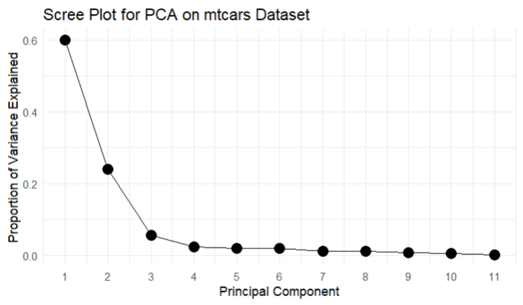
The graphs are a little messy here because the mtcars dataset has very long label names instead of observation numbers.

```
#PCR
library(pls)
pcr_model <- pcr(mpg ~ ., data = mtcars, scale = TRUE, validation = "CV")
summary(pcr_model)
plot(pcr_model, validationplot = TRUE)

library(caret)
pcr_model <- train(mpg ~ ., data = mtcars, method = "pcr", preProcess = "scale", trControl =
trainControl(method = "cv"))
summary(pcr_model)
```



We can also construct a scree plot for our results.

```
# Load necessary libraries
library(ggplot2)

# Perform PCA on the mtcars dataset
pca_result <- prcomp(mtcars, scale. = TRUE)

# Calculate the proportion of variance explained
explained_variance <- pca_result$sdev^2 / sum(pca_result$sdev^2)

# Create a scree plot
scree_data <- data.frame(Principal_Component = seq_along(explained_variance),
                    Variance_Explained = explained_variance)

ggplot(scree_data, aes(x = Principal_Component, y = Variance_Explained)) +
  geom_point(size = 4) +
  geom_line() +
  scale_x_continuous(breaks = scree_data$Principal_Component) +
  xlab("Principal Component") +
  ylab("Proportion of Variance Explained") +
  ggtitle("Scree Plot for PCA on mtcars Dataset") +
  theme_minimal()
```



Coding factor analysis up from scratch can be extremely challenging. If there is collinearity in the variables, you can get non-invertible matrices which break the algorithm and have to be handled with exceptions. So, let's look at an analysis using some built in functions and see what is going on with the process.

```
scores <- read.table("https://raw.githubusercontent.com/sunbeomk/PSYC490/main/scores.txt")
head(scores)
cor_subtests <- cor(scores)

round(cor_subtests, 2)
efa_1 <- factanal(x = scores, factors = 1)
class(efa_1)
efa_1
loadings <- efa_1$loadings # Factor loadings
loadings[1]
as.numeric(loadings)
sum(loadings^2) # SS Loadings
sum(loadings^2) / ncol(scores) # Proportion Var
factanal(x = scores,
     factors = 2)

cov_mat <- cov(scores)
round(cov_mat, 2)
factanal(covmat = cov_mat,
     factors = 2,
     n.obs = nrow(scores))
```

```
output <- factanal(x = scores,
                   factors = 2,
                   scores = "regression")
head(output$scores)

varimax <- factanal(scores,
                    factors = 2,
                    rotation="varimax",
                    scores="regression")
cor(varimax$scores)

promax <- factanal(scores,
                   factors = 2,
                   rotation = "promax",
                   scores = "regression")
cor(promax$scores)

library(psych)
output2 <- fa(scores, # input data
              nfactors = 2, # number of factors
              rotate = "varimax", # rotation
              scores = "regression") # factor score estimation

output2$loadings # factor loadings
output2$uniquenesses # uniqueness
output2$communality # communality
output2$uniquenesses + output2$communalities
```

This code appears to be performing Exploratory Factor Analysis (EFA) on a dataset of scores, using various rotations and methods. The goal of EFA is to identify underlying factors that explain the pattern of correlations within a set of observed variables.

Summary of the Steps:

1. *Loading the Data:* The code starts by reading a dataset from a URL into a data frame named scores. head(scores) displays the first few rows of the dataset.

2. *Correlation Matrix:* cor_subtests <- cor(scores) calculates the correlation matrix for the dataset. round(cor_subtests, 2) rounds the correlation values to two decimal places for better readability.
3. *Factor Analysis with One Factor:* efa_1 <- factanal(x = scores, factors = 1) performs EFA on the data, assuming a single factor.
    - class(efa_1) checks the class of the efa_1 object, which is typically of class "factanal".
    - The code then extracts and analyzes the factor loadings from this one-factor model:
        - efa_1$loadings retrieves the loadings.
        - as.numeric(loadings) converts the loadings to numeric form.
        - sum(loadings^2) calculates the sum of squared loadings, which reflects the total variance explained by the factor.
        - sum(loadings^2) / ncol(scores) computes the proportion of variance explained by the factor.

4. *Factor Analysis with Two Factors:* factanal(x = scores, factors = 2) performs EFA assuming two factors. The output shows the factor loadings for the two-factor solution. The code also performs factor analysis on the covariance matrix directly using:
   - cov_mat <- cov(scores) to compute the covariance matrix.
   - factanal(covmat = cov_mat, factors = 2, n.obs = nrow(scores)) to apply factor analysis using the covariance matrix.

5. *Factor Scores Estimation:* output <- factanal(x = scores, factors = 2, scores = "regression") performs factor analysis with two factors and calculates factor scores using regression. head(output$scores) displays the first few factor scores.

6. *Rotations:*
   - *Varimax Rotation:*
     - varimax <- factanal(scores, factors = 2, rotation = "varimax", scores = "regression") performs a Varimax rotation on the two-factor solution.
     - cor(varimax$scores) computes the correlations between the factor scores after Varimax rotation.
   - *Promax Rotation:*
     - promax <- factanal(scores, factors = 2, rotation = "promax", scores = "regression") performs a Promax rotation, which is an oblique rotation allowing factors to be correlated.
     - cor(promax$scores) checks the correlations between factor scores after Promax rotation.

7. **Using the psych Package:**
   - library(psych) loads the psych package, which provides various functions for factor analysis.
   - output2 <- fa(scores, nfactors = 2, rotate = "varimax", scores = "regression") performs factor analysis using the fa function from the psych package, with Varimax rotation and regression-based factor scores.
   - The output displays the factor loadings, uniquenesses, and communalities:
     - output2$loadings shows the factor loadings.
     - output2$uniquenesses displays the uniqueness values.
     - output2$communality shows the communalities (portion of each variable's variance explained by the factors).
     - output2$uniquenesses + output2$communalities should sum to 1 for each variable (this is a consistency check).

Visualizations and Interpretations:

1. *Correlation Matrix Heatmap:* A heatmap can visually represent the correlation matrix, showing how variables relate to each other.

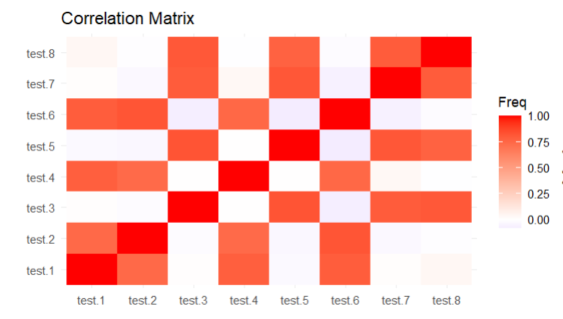
2. *Scree Plot:* A scree plot could be used to determine the number of factors to retain by plotting eigenvalues or variances explained by each factor in descending order.

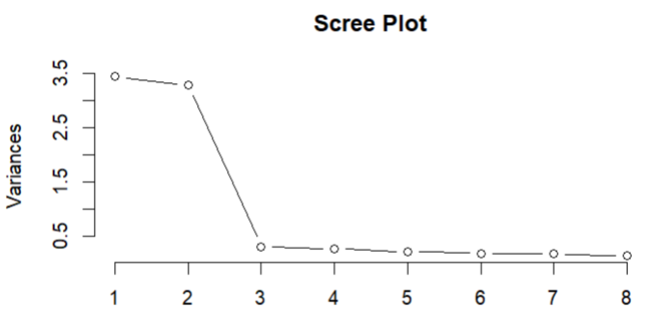
3. *Factor Loadings Plot:* Bar plots or biplots could visualize factor loadings, helping to interpret which variables load strongly on which factors.

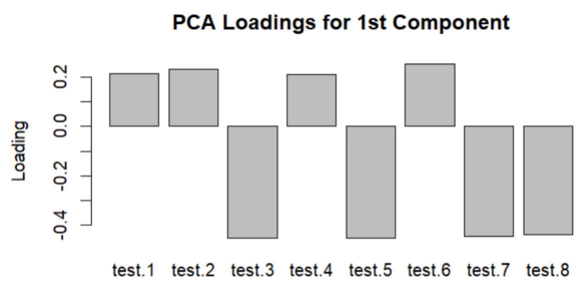
4. *Factor Scores Scatterplot:* Scatterplots of the factor scores (e.g., Varimax or Promax scores) can reveal clusters or patterns in the data.

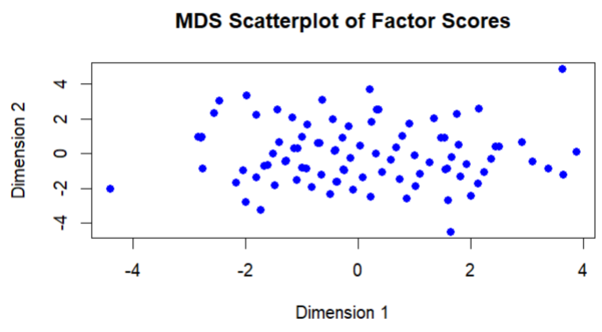
5. *Communality and Uniqueness Plots:* Bar plots could visualize the communalities and uniquenesses, showing how much variance in each variable is explained by the factors and how much is unique.

```
# Load necessary libraries
library(ggplot2)
library(psych)

# Load the dataset
scores <- read.table("https://raw.githubusercontent.com/sunbeomk/PSYC490/main/scores.txt",
header = TRUE)

# Compute the correlation matrix
cor_matrix <- cor(scores)

# Plot the correlation matrix heatmap
ggplot(data = as.data.frame(as.table(cor_matrix)), aes(Var1, Var2, fill = Freq)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
  theme_minimal() +
  labs(title = "Correlation Matrix", x = "", y = "")

# Perform PCA
pca <- prcomp(scores, scale. = TRUE)

# Scree Plot
screeplot(pca, type = "lines", main = "Scree Plot")

# Get PCA loadings
loadings <- pca$rotation

# Plot the loadings for the first principal component
barplot(loadings[,1], main = "PCA Loadings for 1st Component", ylab = "Loading", names.arg =
colnames(scores))

# Perform Multidimensional Scaling (MDS)
mds <- cmdscale(dist(scores))

# Scatterplot of MDS results
plot(mds, xlab = "Dimension 1", ylab = "Dimension 2", main = "MDS Scatterplot of Factor Scores",
pch = 16, col = "blue")
```





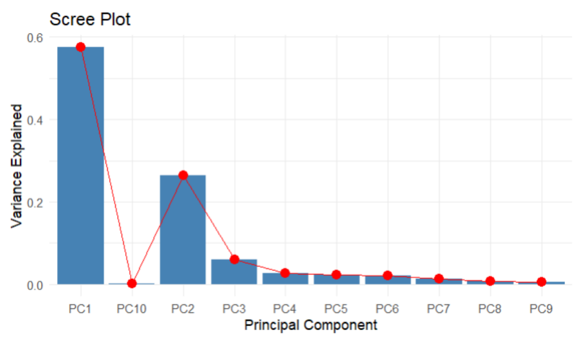


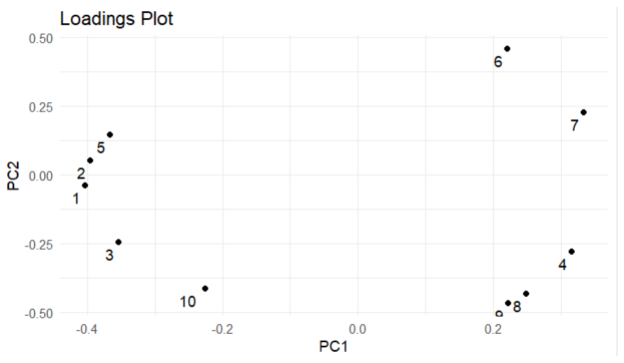


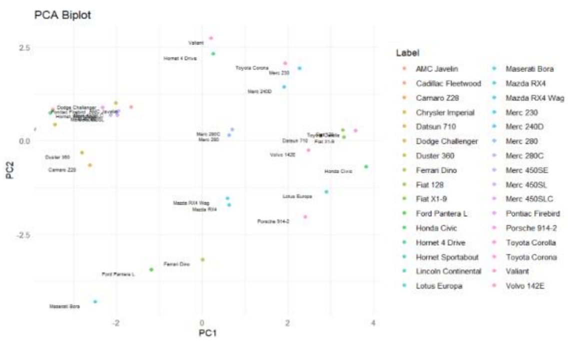
Let's look at the code for the Principal Component Analysis and some visualizations.

```
# Load necessary libraries
library(mlbench)
library(ggplot2)

# Load the mtcars dataset
data("mtcars")

# Extract the numeric variables (excluding the response variable 'mpg')
data_numeric <- mtcars[, -which(names(mtcars) == "mpg")]

# Standardize the data
data_standardized <- scale(data_numeric)

# Calculate the covariance matrix
cov_matrix <- cov(data_standardized)

# Perform eigen decomposition
eigen_decomp <- eigen(cov_matrix)
eigenvalues <- eigen_decomp$values
eigenvectors <- eigen_decomp$vectors

# Calculate the explained variance
explained_variance <- eigenvalues / sum(eigenvalues)

# Project the data onto the principal components
```

```
pca_scores <- data_standardized %*% eigenvectors
colnames(pca_scores) <- paste0("PC", 1:ncol(pca_scores))

# Create a scree plot
explained_variance_df <- data.frame(
  PC = paste0("PC", 1:length(explained_variance)),
  Variance_Explained = explained_variance
)

ggplot(explained_variance_df, aes(x = PC, y = Variance_Explained)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_point(size = 3, color = "red") +
  geom_line(group = 1, color = "red") +
  labs(title = "Scree Plot", x = "Principal Component", y = "Variance Explained") +
  theme_minimal()

# Loadings plot
loadings_df <- as.data.frame(eigenvectors[, 1:2])
colnames(loadings_df) <- c("PC1", "PC2")
loadings_df$Variable <- rownames(loadings_df)

ggplot(loadings_df, aes(x = PC1, y = PC2, label = Variable)) +
  geom_point() +
  geom_text(vjust = 1.5, hjust = 1.5) +
  labs(title = "Loadings Plot", x = "PC1", y = "PC2") +
  theme_minimal()

# Biplot
biplot_df <- data.frame(pca_scores[, 1:2])
biplot_df$Label <- rownames(mtcars)

ggplot(biplot_df, aes(x = PC1, y = PC2)) +
  geom_point(aes(color = Label), alpha = 0.6) +
  geom_text(aes(label = Label), vjust = 1.5, hjust = 1.5, size = 2) +
  labs(title = "PCA Biplot", x = "PC1", y = "PC2") +
  theme_minimal()
```



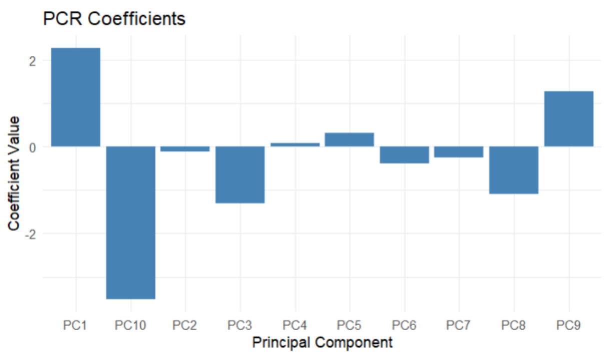


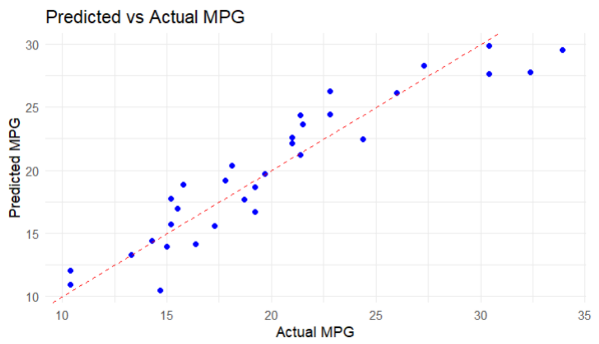


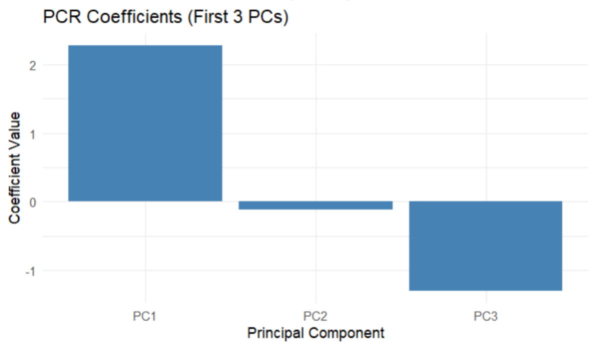
From this point, we can apply Principal Component Regression. First, we'll look at using all 10 principal components, and then a reduced set of just three (which could be reduced even further).

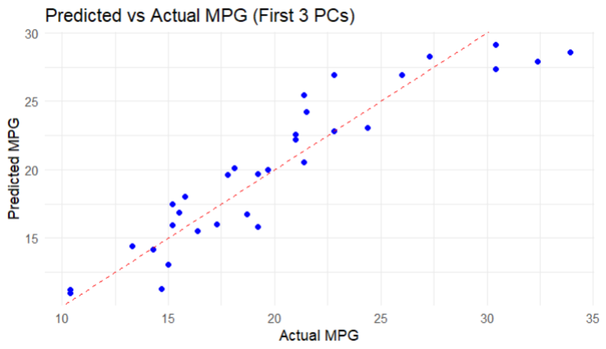
```
# Load necessary libraries
library(ggplot2)

# Use the mtcars dataset
data <- mtcars
target <- data$mpg
data_numeric <- data[, -which(names(data) == "mpg")]

# Standardize the data
data_standardized <- scale(data_numeric)

# Calculate the covariance matrix
cov_matrix <- cov(data_standardized)

# Perform eigen decomposition
eigen_decomp <- eigen(cov_matrix)
eigenvalues <- eigen_decomp$values
eigenvectors <- eigen_decomp$vectors

# Project the data onto the principal components
pca_scores <- data_standardized %*% eigenvectors
colnames(pca_scores) <- paste0("PC", 1:ncol(pca_scores))

# Fit a linear model using the principal components
pcr_model <- lm(target ~ pca_scores)

# Summary of the PCR model
summary(pcr_model)

# Extract the coefficients
coefficients_df <- data.frame(
  PC = paste0("PC", 1:ncol(pca_scores)),
  Coefficients = pcr_model$coefficients[-1]  # Exclude the intercept
)
```

```
# Plot the coefficients
ggplot(coefficients_df, aes(x = PC, y = Coefficients)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "PCR Coefficients", x = "Principal Component", y = "Coefficient Value") +
  theme_minimal()

# Predict the values using the PCR model
predicted_values <- predict(pcr_model)

# Plot the predicted vs actual values
pred_vs_actual <- data.frame(
  Actual = target,
  Predicted = predicted_values
)

ggplot(pred_vs_actual, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Predicted vs Actual MPG", x = "Actual MPG", y = "Predicted MPG") +
  theme_minimal()

# Use only the first three principal components
selected_pcs <- pca_scores[, 1:3]

# Fit a linear model using the first three principal components
pcr_model_selected <- lm(target ~ selected_pcs)

# Summary of the new PCR model
summary(pcr_model_selected)

# Extract the coefficients for the selected components
coefficients_df_selected <- data.frame(
  PC = colnames(selected_pcs),
  Coefficients = pcr_model_selected$coefficients[-1]  # Exclude the intercept
)

# Plot the coefficients
ggplot(coefficients_df_selected, aes(x = PC, y = Coefficients)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  labs(title = "PCR Coefficients (First 3 PCs)", x = "Principal Component", y = "Coefficient Value") +
  theme_minimal()

# Predict the values using the selected PCR model
predicted_values_selected <- predict(pcr_model_selected)

# Plot the predicted vs actual values
pred_vs_actual_selected <- data.frame(
  Actual = target,
  Predicted = predicted_values_selected
)

ggplot(pred_vs_actual_selected, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue") +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Predicted vs Actual MPG (First 3 PCs)", x = "Actual MPG", y = "Predicted MPG") +
  theme_minimal()
```









You can perform backward selection on the principal components in much the same way you do with regular regression, simply remove the principal components one by one starting with the highest P-value until all the components are significant.

Resources:

1. https://www.datacamp.com/tutorial/pca-analysis-r
2. https://www.r-bloggers.com/2021/05/principal-component-analysis-pca-in-r/
3. https://www.geeksforgeeks.org/principal-component-analysis-with-r-programming/
4. https://rpubs.com/KarolinaSzczesna/862710
5. https://rpubs.com/esobolewska/pcr-step-by-step
6. https://www.r-bloggers.com/2016/07/performing-principal-components-regression-pcr-in-r/
7. https://www.statology.org/principal-components-regression-in-r/
8. https://bookdown.org/ssjackson300/Machine-Learning-Lecture-Notes/pcr.html
9. https://www.geeksforgeeks.org/factor-analysis-in-r-programming/
10. https://bookdown.org/sz_psyc490/r4psychometics/factor-analysis.html
11. https://rpubs.com/pjmurphy/758265
12. https://www.geo.fu-berlin.de/en/v/soga-r/Advances-statistics/Multivariate-approaches/Factor-Analysis/A-Simple-Example-of-Factor-Analysis-in-R/index.html