Econ 573: Problem Set 5

Juan M. Alvarez

Part I

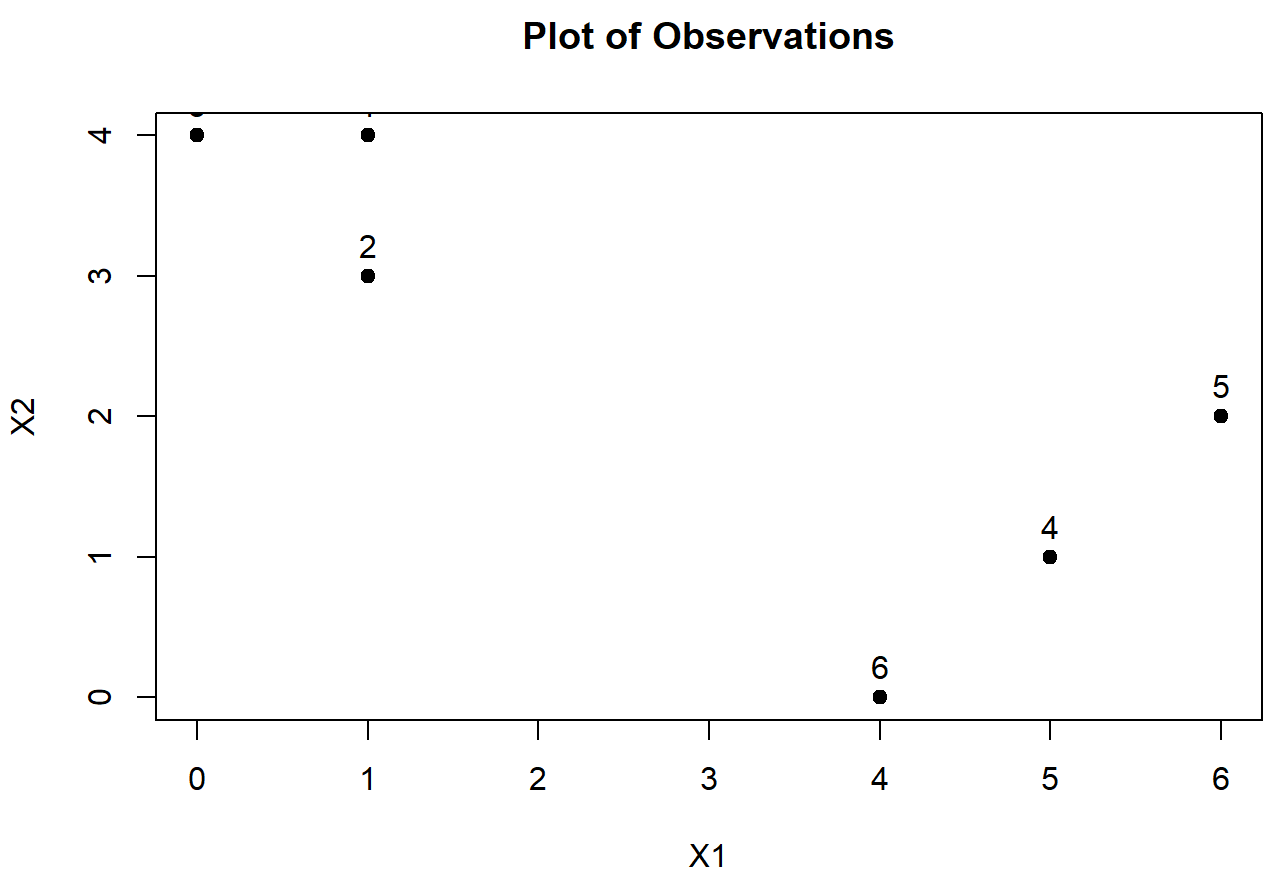
Exercise 3

```
# Creating a data frame with 6 observations and 2 features (X1 and X2)
data <- data.frame(
  Obs = 1:6,                 # Observation number
  X1 = c(1, 1, 0, 5, 6, 4),  # First feature
  X2 = c(4, 3, 4, 1, 2, 0)   # Second feature
)
```

a.

```
# Plotting the observations
plot(data$X1, data$X2, col = "black", pch = 19, xlab = "X1", ylab = "X2", main = "Plot of Observ
ations")

# Adding the observation number in each point
text(data$X1, data$X2, labels = data$Obs, pos = 3)
```



b.

```
# Ensuring results are reproducible
set.seed(123)

# Randomly assigning each point to cluster 1 or 2
data$cluster <- sample(1:2, size = nrow(data), replace = TRUE)
data
```

Obs <int>	X1 <dbl>	X2 <dbl>	cluster <int>
1	1	4	1
2	1	3	1
3	0	4	1
4	5	1	2
5	6	2	1
6	4	0	2

6 rows

c.

```
# Computing centroids for each cluster
centroids <- aggregate(cbind(X1, X2) ~ cluster, data = data, FUN = mean)
centroids
```

	cluster <int>	X1 <dbl>	X2 <dbl>
	1	2.0	3.25
	2	4.5	0.50

2 rows

d.

```
# Assigning each observation to the centroid to which it is closest (in terms of Euclidean dista
nce)
assign_clusters <- function(df, centroids) {
  dist_matrix <- as.matrix(dist(rbind(centroids[, -1], df[, c("X1", "X2")])))
  dist_matrix <- dist_matrix[-c(1:2), 1:2]
  df$cluster <- apply(dist_matrix, 1, which.min)
  return(df)
}

data <- assign_clusters(data, centroids)
data
```

Obs <int>	X1 <dbl>	X2 <dbl>	cluster <int>
1	1	4	1
2	1	3	1
3	0	4	1
4	5	1	2
5	6	2	2
6	4	0	2

6 rows

e.

```
# Iteratively repeating the previous two steps until the answers obtained stop changing
repeat {
  old_clusters <- data$cluster
  centroids <- aggregate(cbind(X1, X2) ~ cluster, data = data, FUN = mean)
  data <- assign_clusters(data, centroids)

  if (all(old_clusters == data$cluster)) break
}
data
```

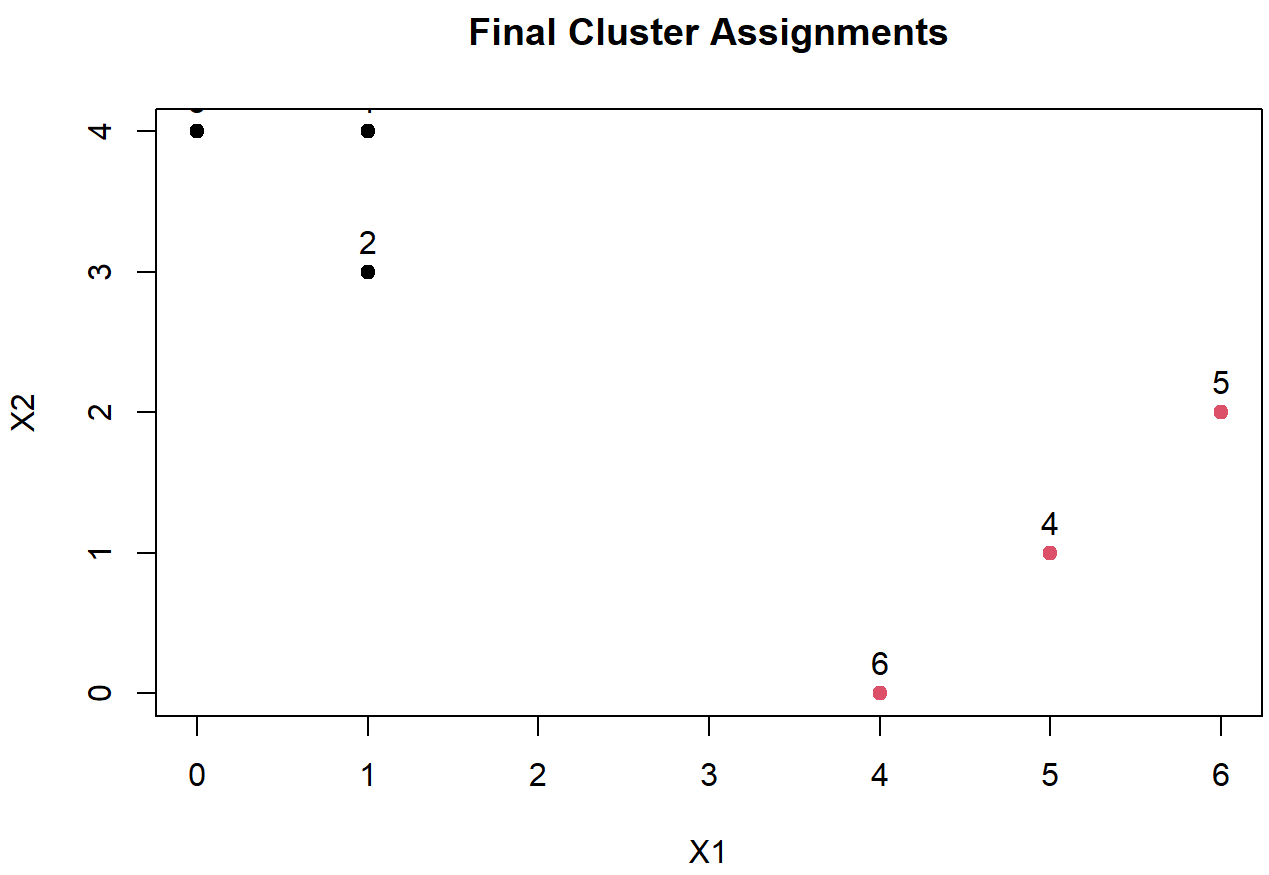
Obs <int>	X1 <dbl>	X2 <dbl>	cluster <int>
1	1	4	1
2	1	3	1
3	0	4	1
4	5	1	2
5	6	2	2
6	4	0	2

6 rows

f.

```
# Plotting the clusters
plot(data$X1, data$X2, col = data$cluster, pch = 19,
     xlab = "X1", ylab = "X2", main = "Final Cluster Assignments")

text(data$X1, data$X2, labels = data$Obs, pos = 3)
```



Exercise 10

a.

```
set.seed(123)  # for reproducibility

# 20 observations per class, 50 variables
n <- 20
p <- 50

# Generate 3 clusters with different means
class1 <- matrix(rnorm(n * p, mean = 0), nrow = n)
class2 <- matrix(rnorm(n * p, mean = 2), nrow = n)
class3 <- matrix(rnorm(n * p, mean = -2), nrow = n)

# Combine into one dataset
X <- rbind(class1, class2, class3)

# Create class labels
true_labels <- c(rep(1, n), rep(2, n), rep(3, n))
```

b.

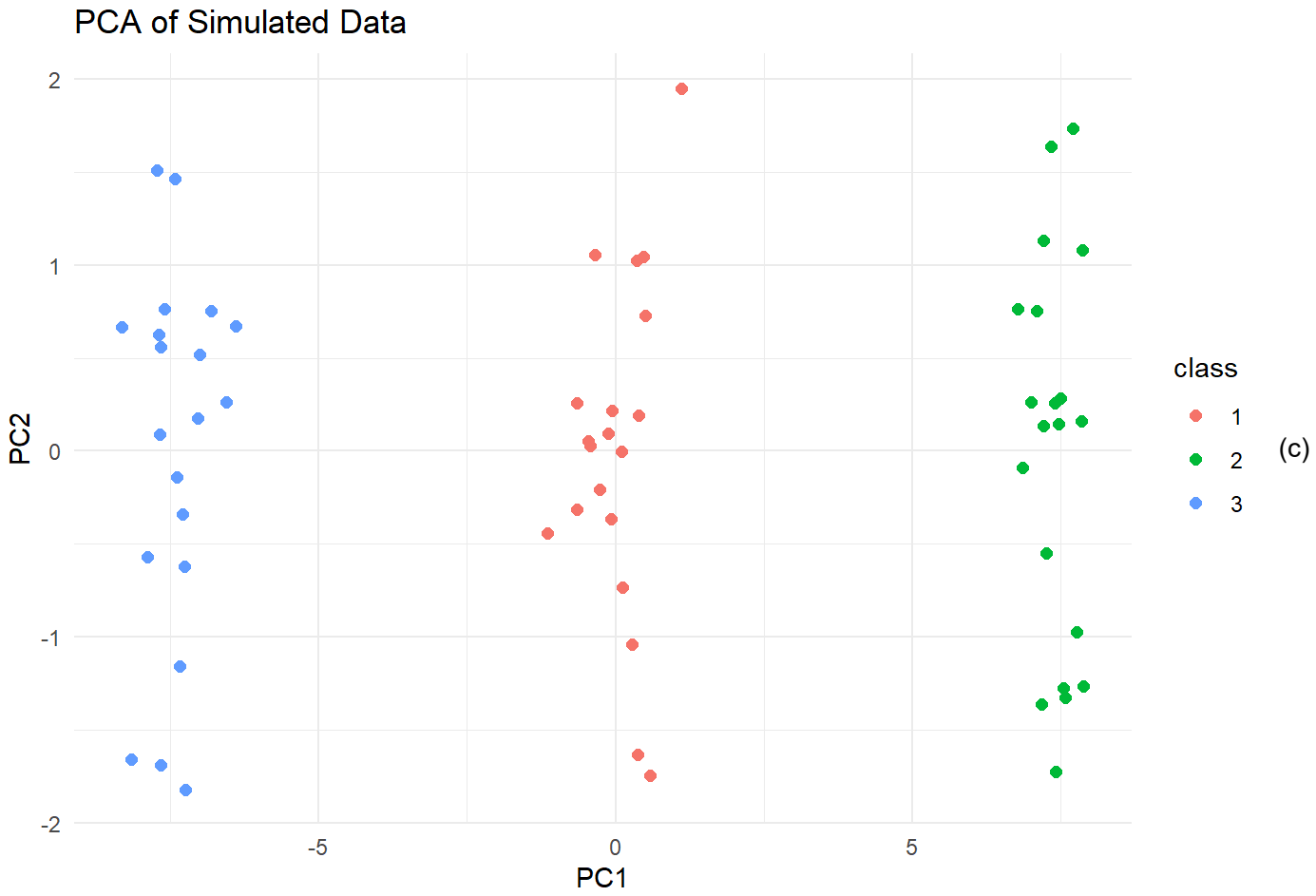
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```
# Run PCA
pca_result <- prcomp(X, scale. = TRUE)

# Plot first two principal components
pc_data <- data.frame(pca_result$x[, 1:2], class = as.factor(true_labels))

ggplot(pc_data, aes(x = PC1, y = PC2, color = class)) +
  geom_point(size = 2) +
  ggtitle("PCA of Simulated Data") +
  theme_minimal()
```




```
k3 <- kmeans(X, centers = 3, nstart = 20)
table(true_labels, k3$cluster)
```

```
##
## true_labels  1  2  3
##           1  0  0 20
##           2  0 20  0
##           3 20  0  0
```

The K-means clustering with K=3 accurately identified the three distinct groups in the simulated data. Although the cluster labels (1, 2, 3) assigned by the algorithm differ from the true class labels, the contingency table shows a perfect one-to-one correspondence between each true class and a unique cluster. This result confirms that K-means was effective at recovering the underlying class structure in the dataset.

d.

```
k2 <- kmeans(X, centers = 2, nstart = 20)
table(true_labels, k2$cluster)
```

```
##
## true_labels  1  2
##           1 20  0
##           2 20  0
##           3  0 20
```

When applying K-means clustering with K=2, the algorithm grouped classes 1 and 2 together into a single cluster, while class 3 was assigned to its own distinct cluster. This suggests that the first two classes are more similar to each other in the feature space compared to class 3, which appears more distinct. However, since there are actually three true classes, setting K=2 leads to under-clustering and a loss of important class structure.

e.

```
k4 <- kmeans(X, centers = 4, nstart = 20)
table(true_labels, k4$cluster)
```

```
##
## true_labels  1  2  3  4
##           1  0  0 20  0
##           2  0  8  0 12
##           3 20  0  0  0
```

When clustering with K=4, K-means split one of the true classes (class 2) into two separate clusters, assigning 8 observations to one and 12 to another. Meanwhile, classes 1 and 3 each remained grouped together in their own clusters. This result suggests that class 2 may contain internal variation that K-means interprets as two subgroups. However, since the true number of classes is three, setting K=4 leads to over-clustering, dividing the data more than necessary.

f.

```
k3_pca <- kmeans(pc_data, centers = 3, nstart = 20)
table(true_labels, k3_pca$cluster)
```

```
##
## true_labels  1  2  3
##           1  0  0 20
##           2 20  0  0
##           3  0 20  0
```

Clustering on the reduced data was highly effective and yielded the same result as clustering on the full dataset. This suggests that PCA not only preserved the class structure but also simplified the data without losing important clustering information.

g.

```
X_scaled <- scale(X)
k3_scaled <- kmeans(X_scaled, centers = 3, nstart = 20)
table(true_labels, k3_scaled$cluster)
```

```
##
## true_labels  1  2  3
##           1  0  0 20
##           2  0 20  0
##           3 20  0  0
```

The results from part (g) were identical to those suggested in part (b). In part (b), PCA showed clear separation between the three classes, and in part (g), K-means clustering on the scaled data achieved perfect classification. This confirms that scaling preserved the structure of the data and ensured equal contribution of all variables, while PCA effectively visualized that structure.

Part 2

```
library(corrplot)
```

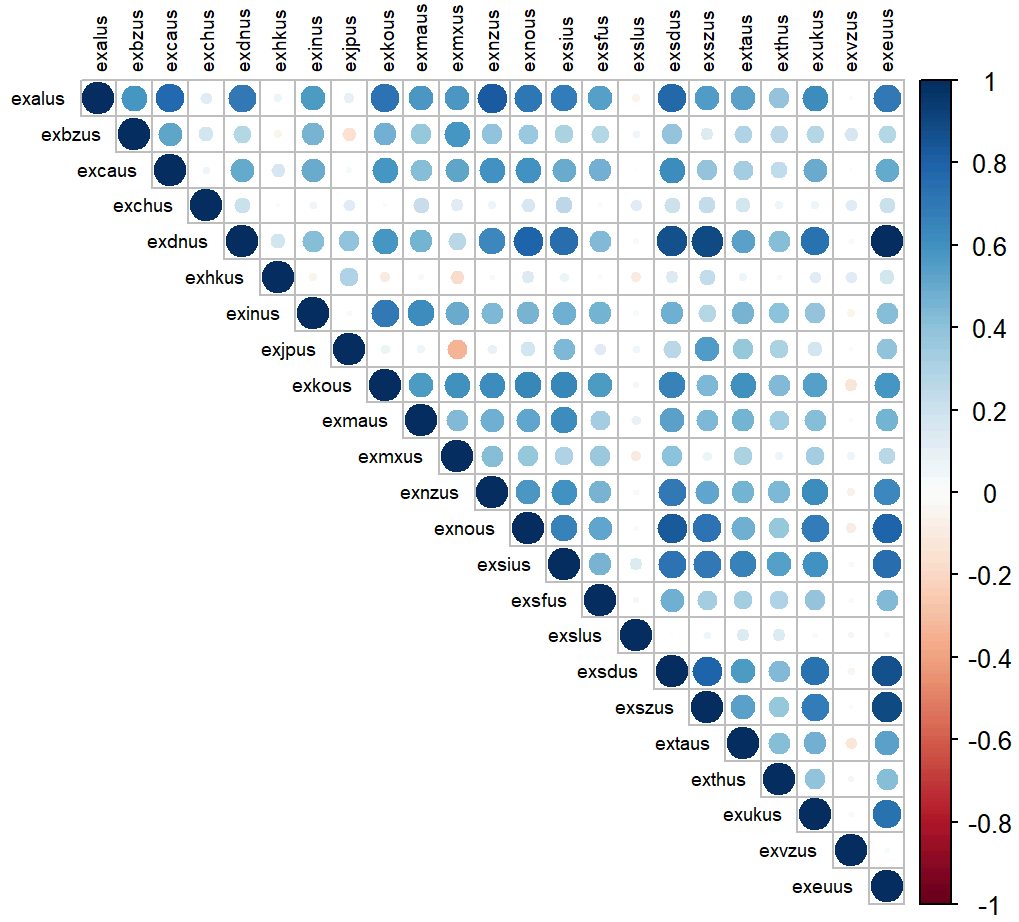
```
## Warning: package 'corrplot' was built under R version 4.4.3
```

```
## corrplot 0.95 loaded
```

```
fx <- read.csv("C:/Users/mateo/Downloads/Fxmonthly.csv")
sp500 <- read.csv("C:/Users/mateo/Downloads/sp500.csv")
fx_returns <- (fx[2:120, ] - fx[1:119, ]) / fx[1:119, ]
```

1)

```
cor_fx <- cor(fx_returns)
corrplot(cor(fx_returns), method = "circle", type = "upper",
         tl.col = "black",
         tl.cex = 0.6,
         number.cex = 0.6,
         mar = c(0, 0, 1, 0))
```

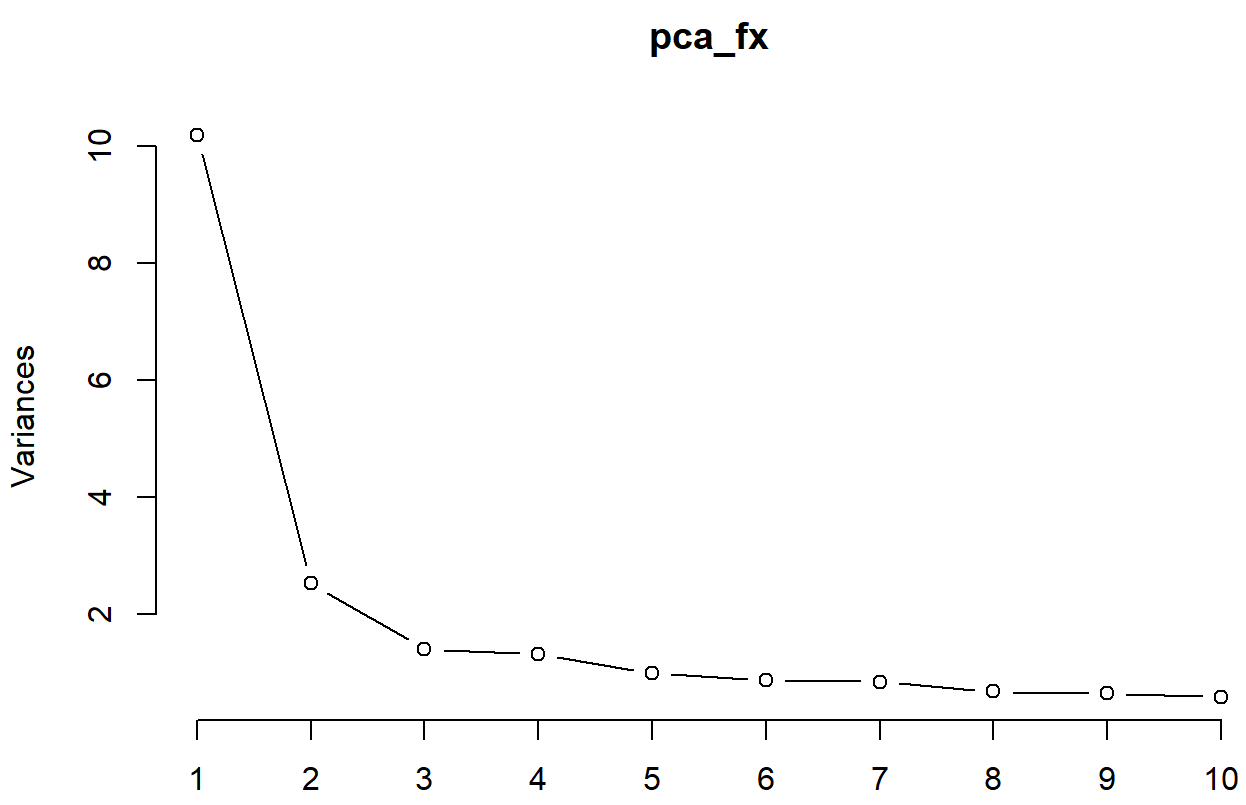


The correlation matrix of the FX return variables shows that many currency pairs are moderately to highly correlated with one another. This suggests the presence of underlying common factors driving movements across multiple currencies.

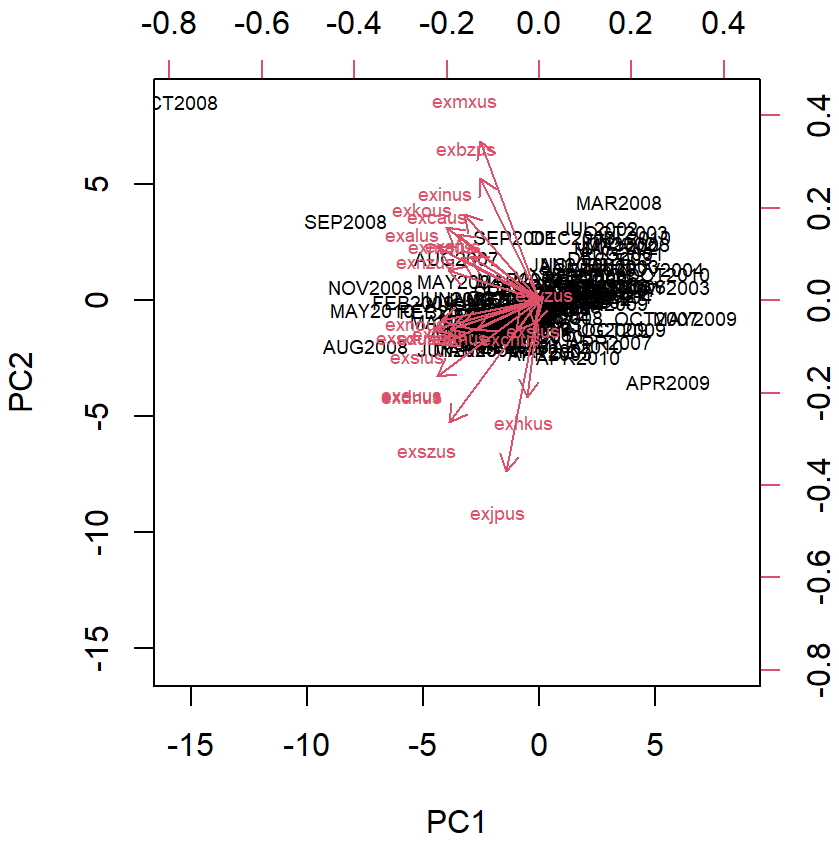
This high degree of correlation makes Principal Component Analysis (PCA) an appropriate technique for this dataset. PCA is most effective when variables are correlated, as it reduces redundancy by capturing the shared variance in fewer uncorrelated components. In this case, PCA can help identify the main latent factors influencing international currency movements, simplifying the analysis without losing significant information.

2)

```
pca_fx <- prcomp(fx_returns, scale. = TRUE)
plot(pca_fx, type = "l")
```




```
biplot(pca_fx, scale = 0, cex = 0.6)
```



The first plot shows that the first principal component (PC1) explains the majority of the variance in the FX returns data, followed by a steep drop-off. This suggests that PC1 captures a dominant underlying factor driving currency movements.

In the biplot, the currency return variables (red arrows) cluster together and point in similar directions, suggesting strong correlations among them. The length of the arrows indicates which currencies contribute most to PC1 and PC2, such as exjpus and exhkus, which may reflect influential currency movements.

3)

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.4.3
```

```
## Cargando paquete requerido: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.4.3
```

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

```
sp500_returns <- (sp500[2:120, 2] - sp500[1:119, 2]) / sp500[1:119, 2]
```

```
pc_scores <- pca_fx$x[, 1:3]
reg_data <- data.frame(SP500 = sp500_returns, pc_scores)

model_glm <- lm(SP500 ~ ., data = reg_data)
summary(model_glm)
```

```
##
## Call:
## lm(formula = SP500 ~ ., data = reg_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -242.01   -7.42   -0.81    3.96  372.43
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1369     3.9426   0.035    0.972
## PC1           1.5222     1.2370   1.231    0.221
## PC2           1.5502     2.4840   0.624    0.534
## PC3           1.8374     3.3359   0.551    0.583
##
## Residual standard error: 42.82 on 114 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.01904,    Adjusted R-squared:  -0.006777
## F-statistic: 0.7375 on 3 and 114 DF,  p-value: 0.5318
```

```
library(glmnet)

x <- as.matrix(pc_scores)
y <- as.vector(sp500_returns)

complete_cases <- complete.cases(x, y)
x <- x[complete_cases, ]
y <- y[complete_cases]

lasso_fit <- cv.glmnet(x, y, alpha = 1)

lasso_fit$lambda.min
```

```
## [1] 4.853144
```

```
final_lasso <- glmnet(x, y, alpha = 1, lambda = lasso_fit$lambda.min)
coef(final_lasso)
```

```
## 4 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 0.1540827
## PC1         0.0000000
## PC2         .
## PC3         .
```

When regressing S&P 500 returns on the first few FX principal components, we found no statistically significant relationships, and lasso regression selected no components. This implies that the latent factors driving currency movements may not strongly predict US equity performance in our dataset.

4)

```
complete_rows <- complete.cases(fx_returns, sp500_returns)
fx_clean <- fx_returns[complete_rows, ]
sp500_clean <- sp500_returns[complete_rows]
```

```
x_fx <- as.matrix(fx_clean)
y_fx <- as.vector(sp500_clean)

lasso_fx <- cv.glmnet(x_fx, y_fx, alpha = 1)

best_lambda_fx <- lasso_fx$lambda.min


final_lasso_fx <- glmnet(x_fx, y_fx, alpha = 1, lambda = best_lambda_fx)

coef(final_lasso_fx)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                    s0
## (Intercept) 0.1540827
## exalus      0.0000000
## exbzus      .
## excaus      .
## exchus      .
## exdnus      .
## exhkus      .
## exinus      .
## exjpus      .
## exkous      .
## exmaus      .
## exmxus      .
## exnzus      .
## exnous      .
## exsius      .
## exsfus      .
## exslus      .
## exsdus      .
## exszus      .
## extaus      .
## exthus      .
## exukus      .
## exvzus      .
## exeuus      .
```

In this analysis, PCR identified PCs with modest positive associations, though not statistically significant, while lasso excluded all predictors, highlighting the limited predictive power of individual currencies for SP500 performance.

bonus step:

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.4.3
```

```
##
## Adjuntando el paquete: 'pls'
```

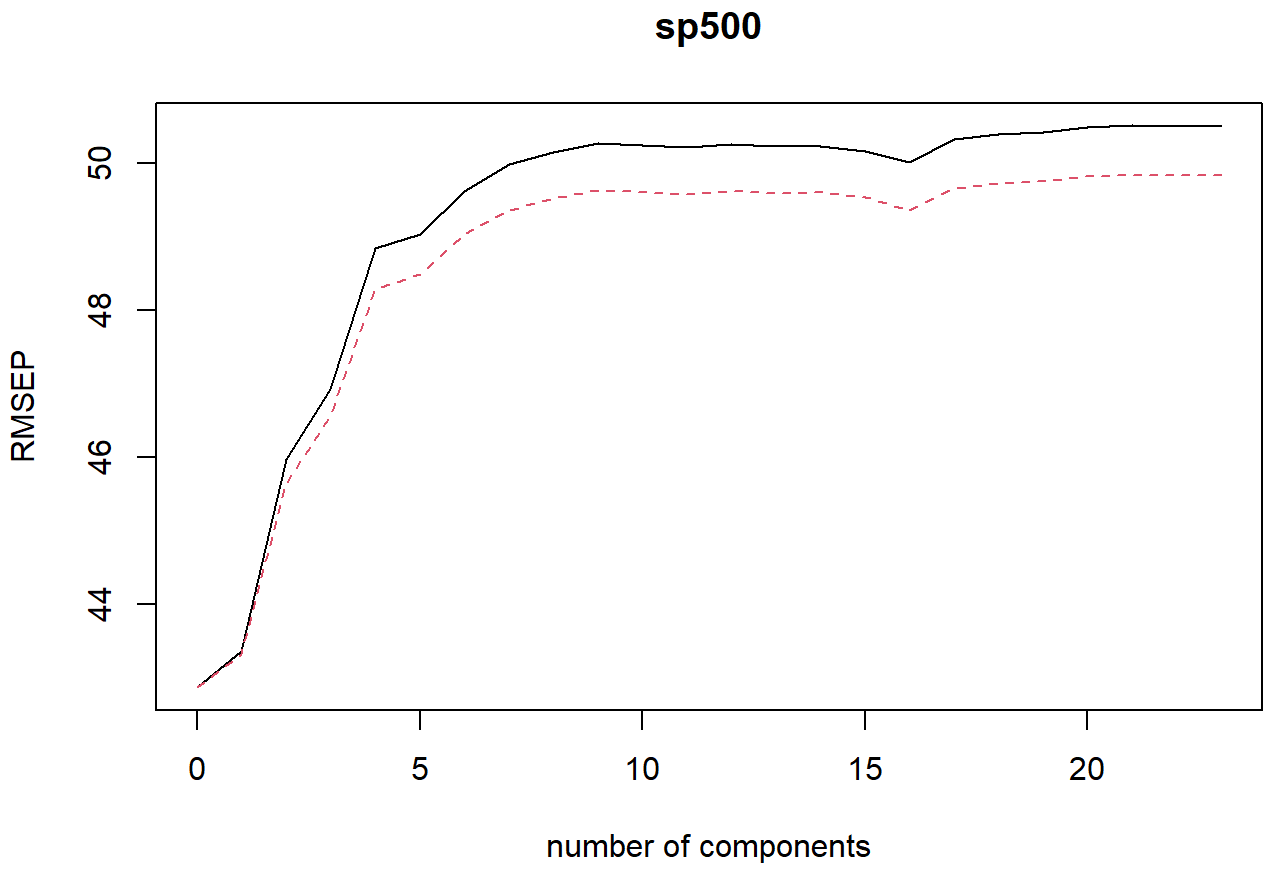
```
## The following object is masked from 'package:corrplot':
##
##     corrplot
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
df_pls <- data.frame(sp500 = sp500_clean, fx_clean)
pls_model <- plsr(sp500 ~ ., data = df_pls, scale = TRUE, validation = "CV")
summary(pls_model)
```

```
## Data:    X dimension: 118 23
##  Y dimension: 118 1
## Fit method: kernelpls
## Number of components considered: 23
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           42.86    43.36    45.97    46.92    48.83    49.03    49.62
## adjCV        42.86    43.31    45.64    46.55    48.28    48.48    49.03
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       49.98    50.15    50.27     50.23     50.21     50.25     50.22
## adjCV    49.36    49.52    49.63     49.59     49.58     49.61     49.58
##        14 comps  15 comps  16 comps  17 comps  18 comps  19 comps  20 comps
## CV        50.23     50.16     50.01     50.33     50.38     50.42     50.49
## adjCV     49.59     49.52     49.35     49.65     49.72     49.75     49.82
##        21 comps  22 comps  23 comps
## CV        50.51     50.50     50.50
## adjCV     49.84     49.83     49.83
##
## TRAINING: % variance explained
##        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X        41.56    49.16    58.03    61.46    66.27    68.52    71.72    75.36
## sp500     4.34    15.48    17.70    19.77    20.16    20.39    20.48    20.52
##        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X        78.93     80.56     83.12     84.90     87.09     89.13     90.74
## sp500    20.53     20.56     20.56     20.57     20.58     20.59     20.63
##        16 comps  17 comps  18 comps  19 comps  20 comps  21 comps  22 comps
## X         91.78     93.60     94.79     95.87     97.14     98.13     99.34
## sp500     20.96     21.29     21.68     21.86     21.92     21.92     21.92
##        23 comps
## X        100.00
## sp500     21.92
```

```
validationplot(pls_model, val.type = "RMSEP")
```




```
coef(pls_model, ncomp = 4)
```

```
## , , 4 comps
##
##              sp500
## exalus  -0.4690802
## exbzus  -7.1763294
## excaus   0.5734220
## exchus   1.7478521
## exdnus  -4.8319188
## exhkus   5.7328431
## exinus  -6.9186602
## exjpus   2.2762952
## exkous   9.3628962
## exmaus   4.1996987
## exmxus   8.2372158
## exnzus  -5.1809033
## exnous   7.1713769
## exsius   3.9506058
## exsfus  -5.3390199
## exslus  -3.7147324
## exsdus   4.7595326
## exszus  -5.7006735
## extaus   5.1672782
## exthus  -2.5871283
## exukus -11.5287025
## exvzus  -0.4076978
## exeuus  -4.3492303
```

```
preds <- predict(pls_model, ncomp = 4)
r2_pls <- cor(preds, sp500_clean)^2
r2_pls
```

```
## [1] 0.1977322
```

Unlike PCA, which captured overall variance in currency returns, and lasso, which identified no predictive covariates, PLS successfully extracted components from currency returns that jointly predict nearly 20% of S&P 500 return variance. The analysis reveals that appreciation of emerging market currencies such as the Mexican peso and South Korean won tend to align with positive US equity returns, whereas movements in the British pound and Indian rupee show a negative relationship.