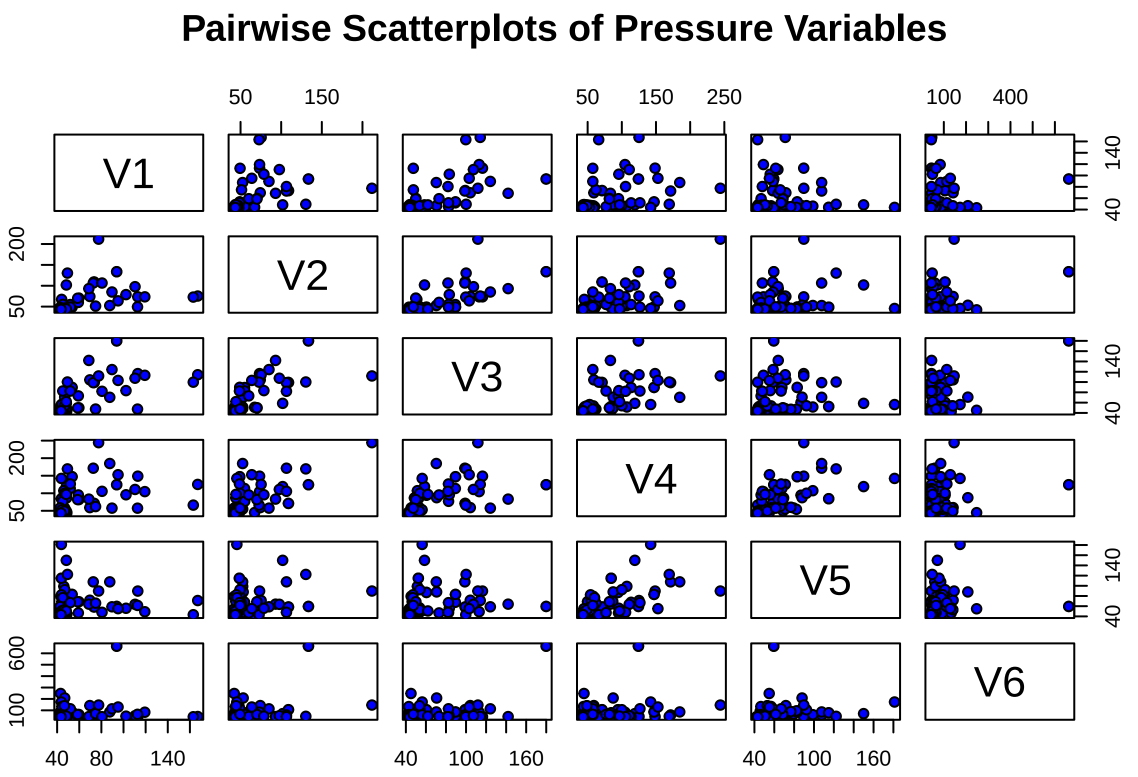
```r
# import necessary libraries
library(caret)
library(MASS)
library(pls)
library(dplyr)
```

Question 1

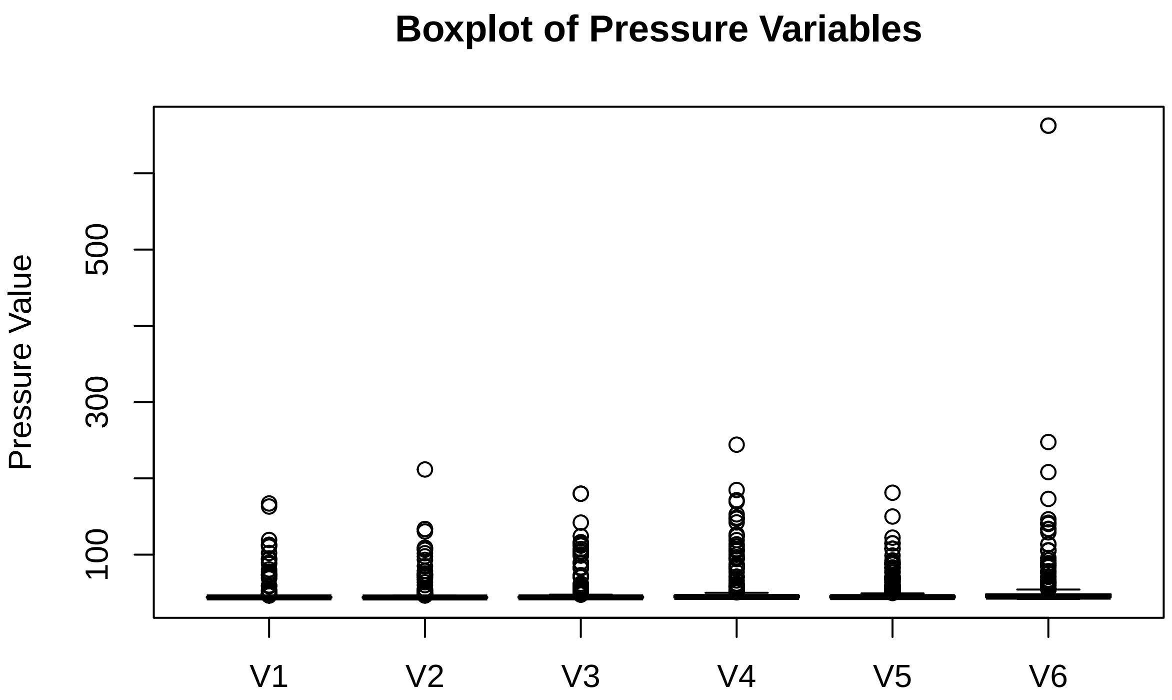
```r
# Set seed and import the data
set.seed(22723265)
df_prime <- read.csv("Pressure_Data.csv")

# Sample a random subset of 400 observations from the dataset.
pressure_data <- df_prime[sample(nrow(df_prime), 400, replace = FALSE), ]

# Removing the index column 'X'
pressure_data <- pressure_data[, -1]
```

Question 2

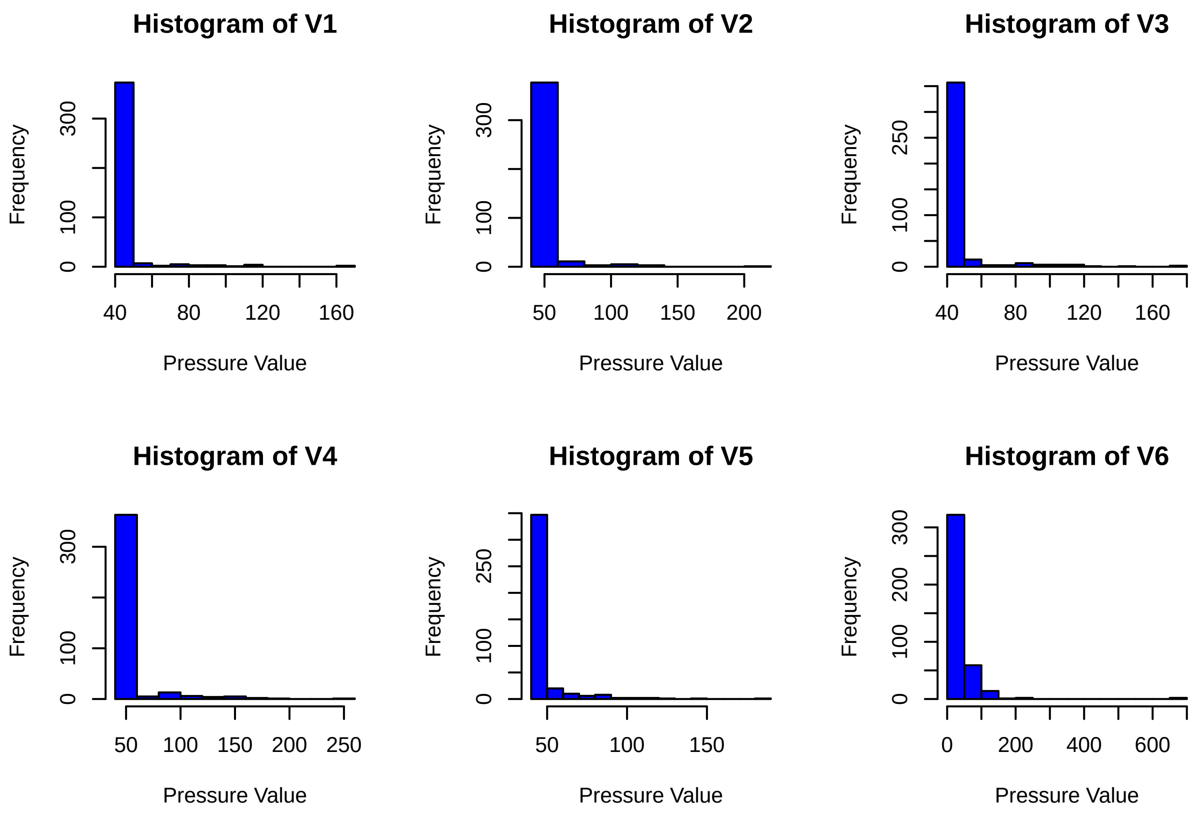
```r
# Remove observations which has missing/NA value from specified variables
pressure_data <- pressure_data[complete.cases(
  pressure_data[,c("Mattress_type","Position", "Posture", "Subject")]), ]

# Convert categorical columns to factors
categorical_vars <- c("Mattress_type", "Position", "Posture", "Subject")
pressure_data[categorical_vars] <- lapply(pressure_data[categorical_vars], as.factor)

# Visualise the the pressure measurement variables V1,..,V144 (subset V1,..,V6)

# Pairwise scatterplots
pairs(pressure_data[, 1:6],  pch = 21, bg = "blue",
      main = "Pairwise Scatterplots of Pressure Variables")
```



```
# Boxplots
boxplot(pressure_data[, 1:6], main = "Boxplot of Pressure Variables",
        ylab = 'Pressure Value', col = "blue")
```



```
# Histograms
par(mfrow = c(2, 3))
for (col in 1:6) {
  hist(pressure_data[, col], main = paste("Histogram of", colnames(pressure_data)[col]),
        xlab = "Pressure Value", col = "blue")
}
```



Observations Scatterplots: The pairwise scatterplots suggest that there is little to no clear linear relationship between the selected pressure variables. Most of the observations fall into a single cluster, making it difficult to tell whether other distinct groups exist. There are also a few outliers that stand out from the dense region, suggesting that the data might be heavily skewed. Another issue is that the variables are on different scales, which could affect certain types of analysis.

Boxplots: The boxplots indicate that all six selected pressure variables have a high number of outliers, shown as points beyond the whiskers. The median values are quite low relative to the range, meaning most of the data is clustered near the lower end. It's possible that these extreme values come from data collection errors, but they could also simply reflect the natural variation in the data. Either way, they might need further investigation.

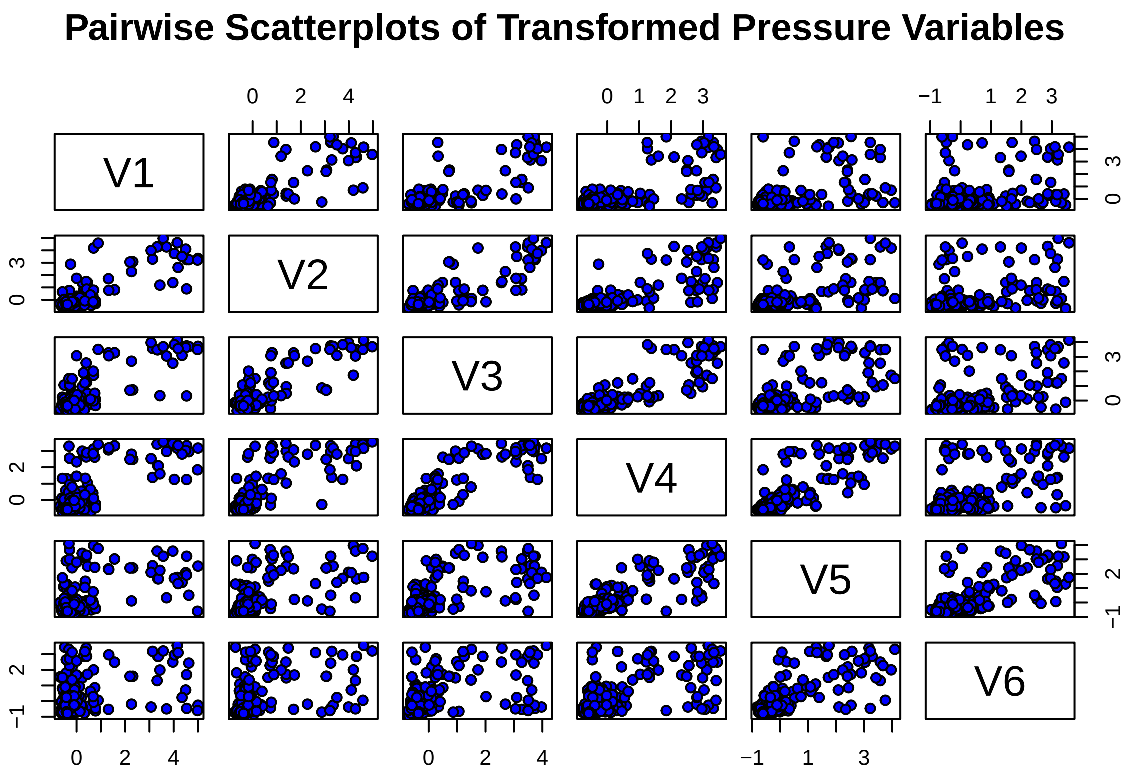
Histograms: The histograms show that the pressure variables are heavily skewed to the right, with most values sitting at the lower end and a few extremely high ones pulling the distribution. This kind of skewness can cause problems in statistical modelling, so a transformation such as a log or Box-Cox transformation might be useful to normalise the data before further analysis.

Potential Issues and Solutions Skewness of Data: Since the histograms reveal significant right skewness, the data isn't normally distributed. If normality is important for certain statistical methods, a log or Box-Cox transformation could help by making the data more balanced. This may improve interpretability and it might also help reveal hidden patterns and make clusters more distinct.

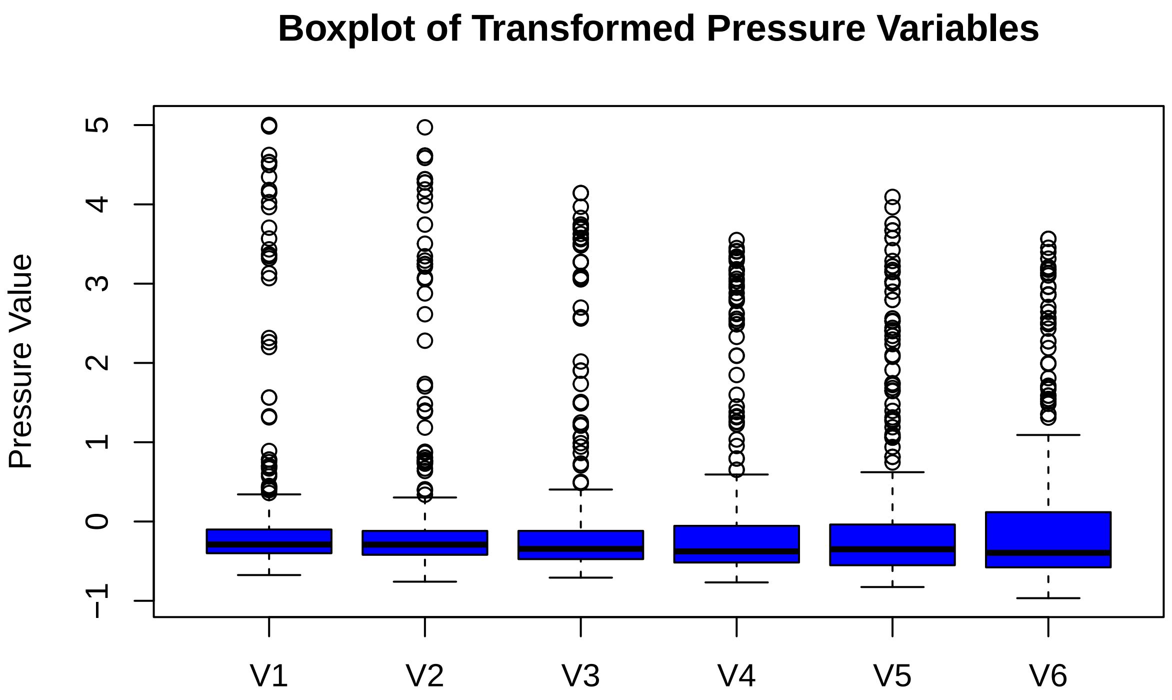
Different Scales Across Variables: The pressure variables (V1–V144) are on very different scales, which could throw off analyses that rely on distances, such as clustering and PCA. To fix this, I'll standardise the data using Z-score normalisation—subtracting the mean and dividing by the standard deviation. This way, all variables contribute equally, and no single feature dominates the results.

Presence of Outliers: The boxplots show quite a few extreme values, which could be valid variations or measurement errors. While some of these values may carry useful information, they could also cause problems in statistical models by distorting patterns. A Box-Cox transformation could help reduce their influence, making the data more stable. If certain values look completely unrealistic, it might be worth taking a closer look before proceeding.

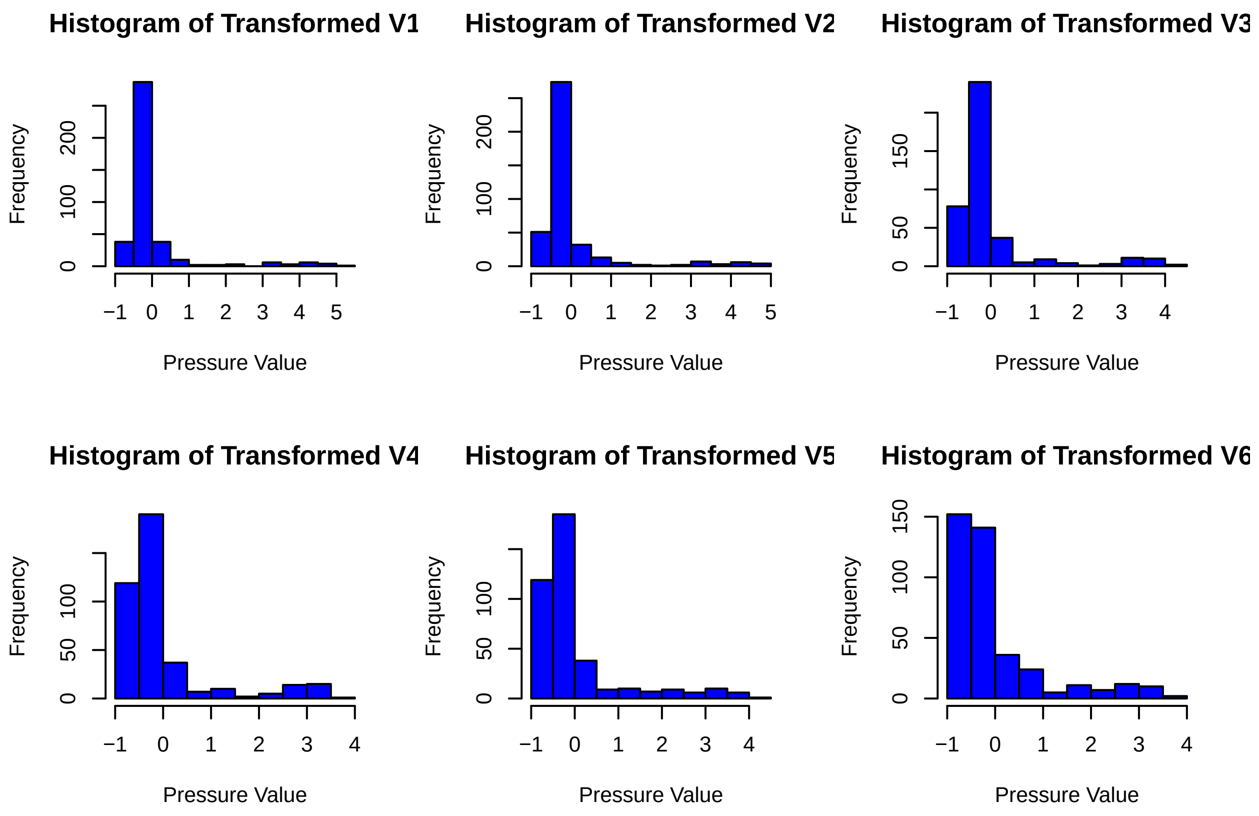
```r
# BoxCoxTrans from caret to each column
pressure_data[, 1:144] <- lapply(pressure_data[, 1:144], function(x) {
  transform <- BoxCoxTrans(x)
  predict(transform, x)
})

# Standardise pressure variables (columns V1 to V144)
pressure_data[, 1:144] <- scale(pressure_data[, 1:144])

# Visualise the transformed and scaled pressure measurement variables V1,..,V144 (subset V1,..,V6)
# Pairwise scatterplots
pairs(pressure_data[, 1:6],
      pch = 21, bg = "blue",
      main = "Pairwise Scatterplots of Transformed Pressure Variables")
```



```r
# Boxplots
boxplot(pressure_data[, 1:6], main = "Boxplot of Transformed Pressure Variables",
        ylab = 'Pressure Value', col = "blue")
```



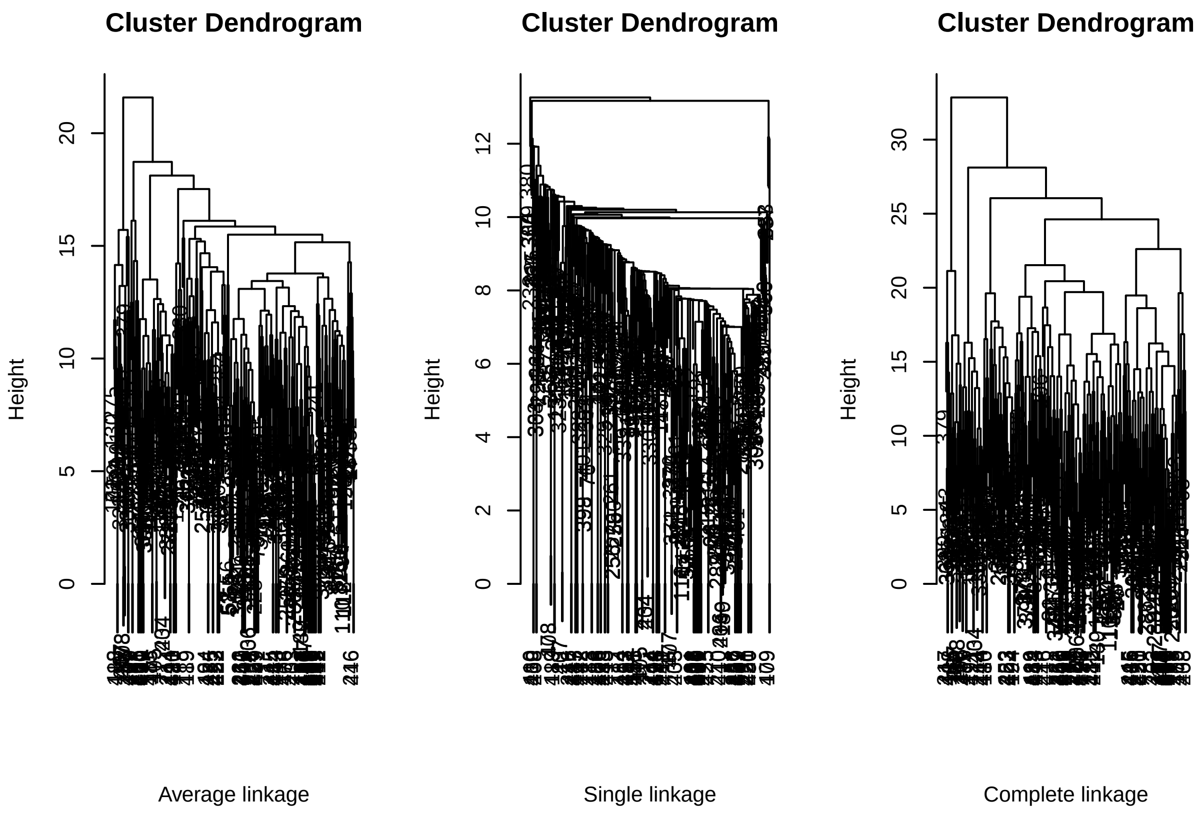
```r
# Histograms
par(mfrow = c(2, 3))
for (col in 1:6) {
  hist(pressure_data[, col], main = paste("Histogram of Transformed", colnames(pressure_data)[col]),
       xlab = "Pressure Value", col = "blue")
}
```



Even after the Box-Cox transformation and standardisation, the data is still somewhat skewed, but it's definitely an improvement. The extreme values don't stand out as much, and the distributions look more balanced overall. Looking at the scatterplots, there also seems to be a bit more separation between some points, suggesting that potential clusters are starting to emerge. While it's not perfect, the transformations have made the data easier to work with and more suitable for further analysis.

Question 3

```r
# Hierarchical clustering
# Compute the Euclidean distance matrix
pressure_dist <- dist(pressure_data[, 1:144], method = "euclidean")

# Plot the Cluster Dendrograms using average, single and complete methods
par(mfrow = c(1, 3))
cl.average = hclust(pressure_dist, method="average")
plot(cl.average, xlab="Average linkage", sub="")

cl.single = hclust(pressure_dist, method="single")
plot(cl.single, xlab = "Single linkage", sub="")

cl.complete = hclust(pressure_dist, method="complete")
plot(cl.complete, xlab = "Complete linkage", sub = "")
```



When deciding the optimal number of clusters from a dendrogram, you should look for a natural "cut" where the branches of the tree remain separate before merging into a single cluster. This is often done by visually identifying a height at which the majority of clusters are distinct before merging into one.

In single linkage, the clustering appears highly unbalanced, which can indicate chaining effects. It might not be the best method for this dataset.Average linkage and complete linkage appear to provide clearer separation. In complete linkage, the clusters seem more well-defined.

A good cut in a dendrogram is usually chosen at a height where a large jump occurs before the next merge, suggesting that the observations within each cluster are more similar to each other than to those in other clusters. From the plots, k = 3 should be suitable using the Complete linkage method although I will still evaluate with the other methods for comparison.

```
# Use cutree method to perform a cut at k = 3
clusters_single <- cutree(cl.single, k = 3)
table(clusters_single)
```

```
## clusters_single
##   1   2   3
## 395   4   1
```

```
clusters_avg <- cutree(cl.average, k = 3)
table(clusters_avg)
```

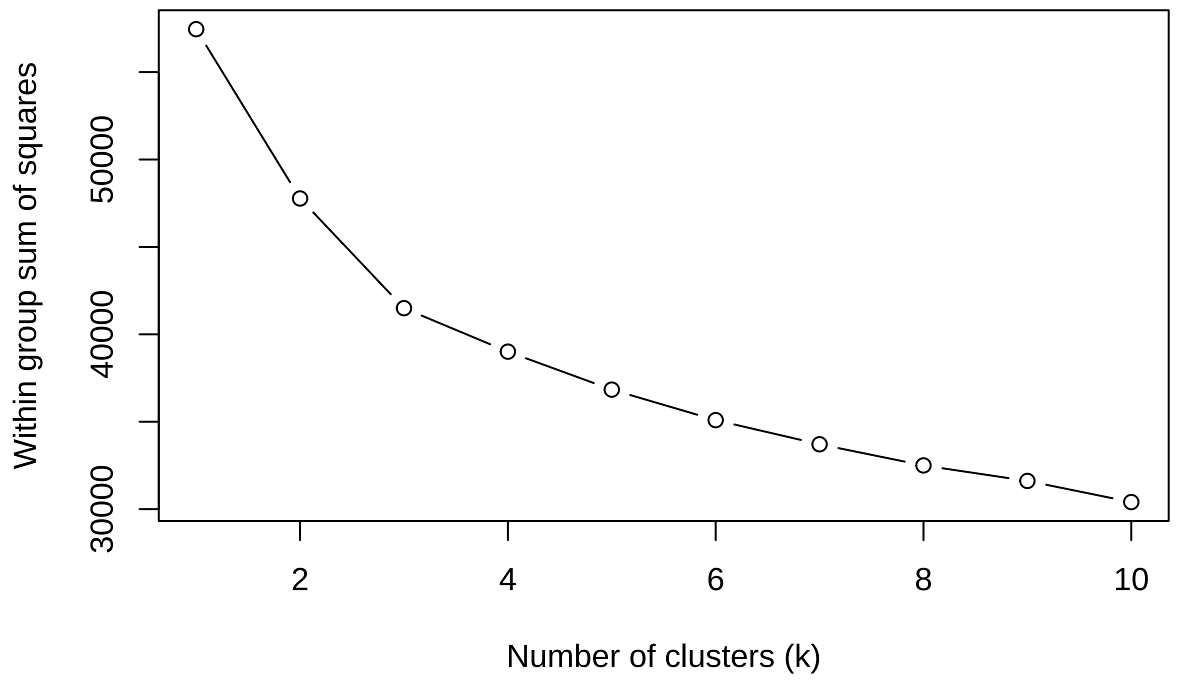
```
## clusters_avg
##   1   2   3
## 358  28  14
```

```
clusters_complete <- cutree(cl.complete, k = 3)
table(clusters_complete)
```

```
## clusters_complete
##   1   2   3
## 338  26  36
```

From inspection of the tables, the complete linkage method appears to have performed the best at separating the data into 3 distinct clusters, however it must be noted that the majority of observations are still being assigned to the first cluster in all three methods, indicating imbalanced classification i.e. the model is still struggling a bit trying to find groupings in the data.

```
# K-means clustering
# Calculate Within Group Sum of Squares (WGSS) for clusters (k)
WGSS <- rep(0, 10)
n <- nrow(pressure_data)
WGSS[1] <- (n - 1) * sum(apply(pressure_data[, 1:144], 2, var))

for (k in 2:10) {
  WGSS[k] <- sum(kmeans(pressure_data[, 1:144], centers = k, nstart = 10)$withinss)
}

# Plot WGSS to identify optimal k using the elbow method.
plot(1:10, WGSS, type = "b", xlab = "Number of clusters (k)", ylab = "Within group sum of squares")
```

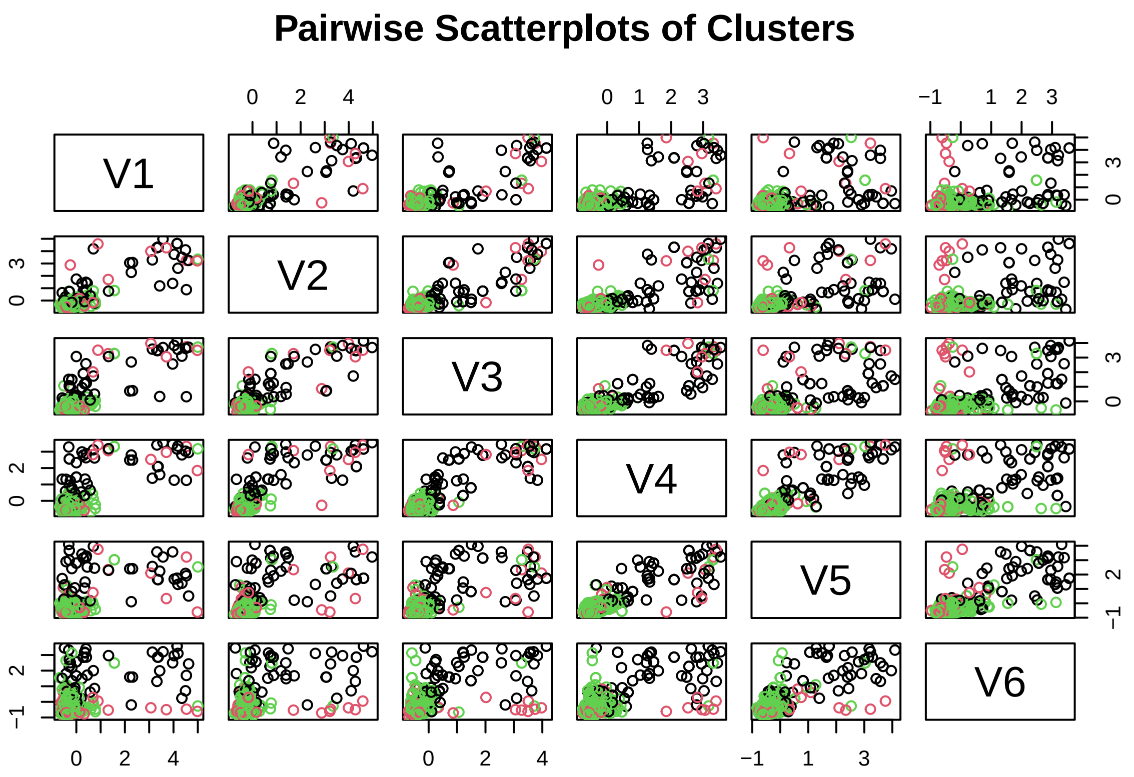


The WGSS plot decreases as number of clusters increases, as expected. To decide on optimal k we choose the point after which adding more clusters provides minimal benefit. A clear elbow point is observed around k = 3. Thus, k = 3 is chosen as optimal.

```r
# Perform k-means clustering with k=3
k <- 3
cl <- kmeans(pressure_data[, 1:144], centers = k, nstart = 10)
table(cl$cluster)
```

```
##
##   1   2   3
##  69  79 252
```

```r
# Visualise clustering results
pairs(pressure_data[, 1:6],  main = "Pairwise Scatterplots of Clusters",  col = cl$cluster)
points(cl$centers[, 1:6], col = 1:k, pch = 8, cex = 2)
```



The clusters contain the following number of observations: Cluster 1: 69 observations Cluster 2: 79 observations Cluster 3: 252 observations The pairwise scatterplots show how the clusters are distributed across the first six pressure variables. There is some overlap between the clusters, which was expected due to the skewness in the data. The cluster sizes are also quite imbalanced, with Cluster 3 being much larger than the others. This suggests that a large portion of the observations share similar characteristics, while the remaining points form smaller, more distinct groups.

```
# Compare k-means clustering with hierarchical clustering
hcl <- cutree(hclust(dist(pressure_data[, 1:144])), k)
pcl <- kmeans(pressure_data[, 1:144], centers = k, nstart = 10)
tab <- table(hcl, pcl$cluster)
tab
```

```
##
## hcl   1   2   3
##   1  53  33 252
##   2  26   0   0
##   3   0  36   0
```

The table compares hierarchical and k-means clustering results:

Hierarchical Cluster 1 (53, 33, 252) overlaps heavily with all three k-means clusters, capturing most of the data. Hierarchical Cluster 2 (26, 0, 0) aligns entirely with k-means Cluster 1, showing strong agreement. Hierarchical Cluster 3 (0, 36, 0) matches k-means Cluster 2, indicating a well-separated group. The K-means method provides finer separation, while hierarchical clustering forms broader groups.

Given the context of the data, the Posture variable has three categorical groups: left-side, right-side, and supine. These results suggest that the pressure data might also naturally split into three groups. So, when deciding on the number of clusters for k-means or hierarchical clustering, starting with k = 3 seems like a good first guess. If posture affects how people lie on the mattress, it could definitely affect pressure readings.

Question 4

```r
# Perform LDA
lda.res <- lda(Posture ~ .,
               data = pressure_data[, c(1:144, which(colnames(pressure_data) == "Posture"))])

# Take a look at the prior probabilities of each class
lda.res$prior
```

```
##   Left  Right Supine
## 0.1950 0.2075 0.5975
```

```r
# Define number of observations and groups
N <- nrow(pressure_data)
G <- length(table(pressure_data$Posture))

# Create subsets for each posture category
pressure_data.left = subset(pressure_data, Posture == "Left")
pressure_data.right = subset(pressure_data, Posture == "Right")
pressure_data.supine = subset(pressure_data, Posture == "Supine")

# Compute class-specific covariance matrices
cov_left <- cov(pressure_data.left[, 1:144])
cov_right <- cov(pressure_data.right[, 1:144])
cov_supine <- cov(pressure_data.supine[, 1:144])

# Compute pooled covariance matrix
cov_all <- ((cov_left * (nrow(pressure_data.left) - 1)) +
              (cov_right * (nrow(pressure_data.right) - 1)) +
              (cov_supine * (nrow(pressure_data.supine) - 1))) / (N - G)

# Define ldf function
ldf = function(x, prior, mu, covar)
{
  x = matrix(as.numeric(x), ncol=1)
  log(prior) - (0.5*t(mu)%*%solve(covar)%*%mu) + (t(x)%*%solve(covar)%*%mu)
}

# Compute LDA decision function values
id <- 1
dfs <- rep(0, G)
for (g in 1:G) {
  dfs[g] <- ldf(pressure_data[id, 1:144],
                lda.res$prior[g], lda.res$means[g, ], cov_all)
}

# Print decision function values and identify class with highest decision function value
dfs
```

```
## [1] -25.5206600 -18.3831622   0.7914321
```

```r
names(lda.res$prior)[dfs == max(dfs)]
```

```
## [1] "Supine"
```

The output [1] “Supine” indicates that the model predicts the first observation belongs to the “Supine” posture. Now to evaluate the model.

```
# Perform cross-validation for LDA
lda.res.cv <- lda(Posture ~ ., CV = TRUE,
                  data = pressure_data[, c(1:144, which(colnames(pressure_data) == "Posture"))])
conf_matrix <- table(lda.res.cv$class, pressure_data$Posture)
conf_matrix
```

```
##
##           Left Right Supine
##   Left      72     2      9
##   Right      0    72      8
##   Supine     6     9    222
```

```
# Compute classification error rate
error_rate <- 1 - sum(diag(conf_matrix)) / sum(conf_matrix)
cat("LDA Classification Error Rate:", error_rate, "\n")
```

```
## LDA Classification Error Rate: 0.085
```

```
# Compute posterior probabilities
round(exp(dfs) / sum(exp(dfs)), 4)
```

```
## [1] 0 0 1
```

```
# Try Perform QDA
tryCatch({
  qda.res.cv <- qda(Posture ~ ., CV = TRUE,
      data = pressure_data[, c(1:144, which(colnames(pressure_data) == "Posture"))])
}, error = function(e) { cat("QDA failed with error:\n", e$message, "\n")
})
```

```
## QDA failed with error:
##  some group is too small for 'qda'
```

LDA was performed to classify subjects into three posture categories: Left, Right, and Supine using pressure data (V1-V144). The prior probabilities for each class were computed as: Left: 19.5% Right: 20.75% Supine: 59.75%

Cross-validation was performed to assess classification accuracy. The Confusion matrix results show that Supine posture had the highest correct classification rate (222 correctly classified).Some misclassification in Left (9 instances) and Right (8 instances). Overall classification accuracy: 91.5% Error rate: 8.5% The LDA model performed reasonabely well, especially given the imbalance in posture categories.

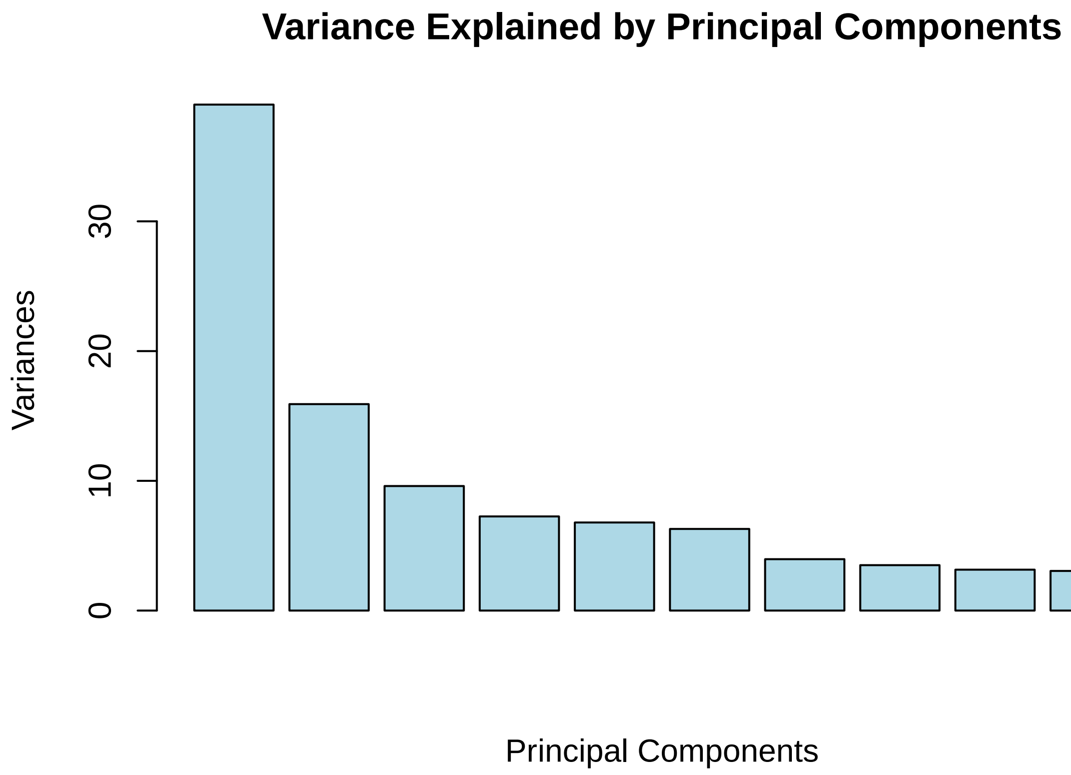
Posterior probability calculations showed that the model confidently classified an observation as Supine.

Attempted to fit QDA, but it failed due to insufficient observations in at least one class. Reason for failure: LDA assumes that all classes share the same covariance structure, while QDA allows each class to have its own. Since one class had too few observations, QDA couldn't properly estimate its covariance matrix, making it impossible to fit the model.

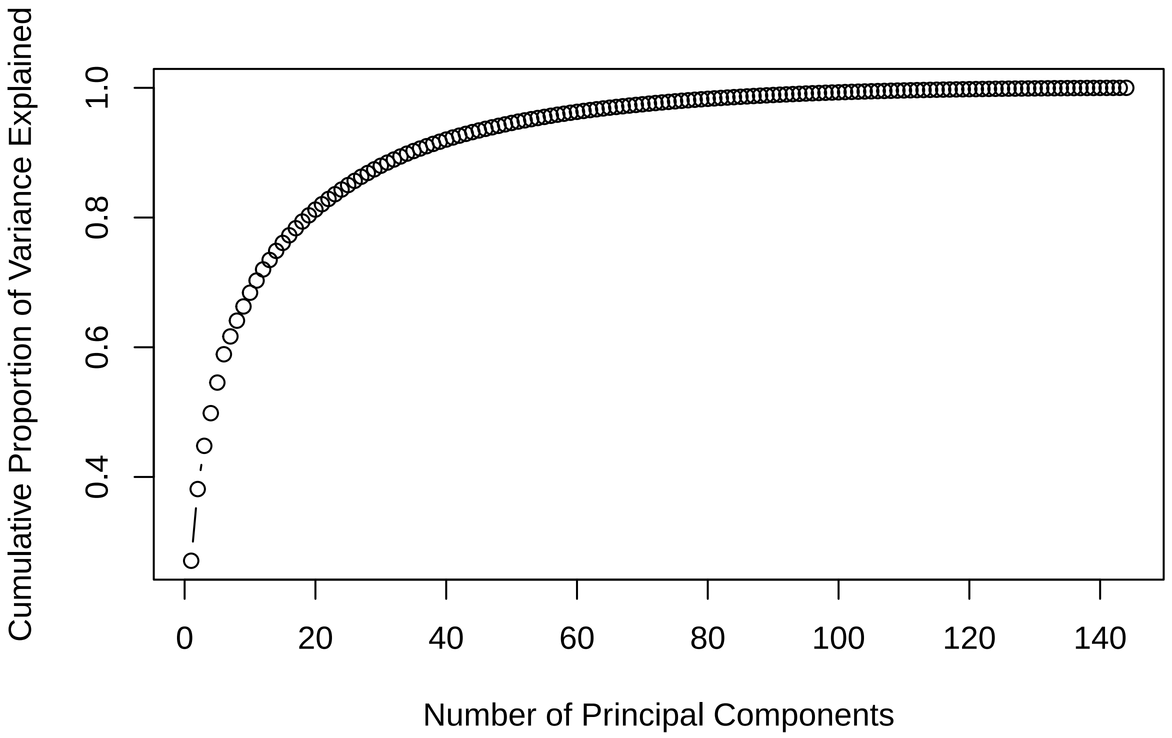
Question 5

```
# Perform Principal Component Analysis (PCA)
fit = prcomp(pressure_data[, 1:144])
# summary(fit) was not added as 144 leads to quite a bulky summary

# plot the fit
```

```r
plot(fit, main = "Variance Explained by Principal Components",
     xlab = "Principal Components", col = "lightblue"
)
```

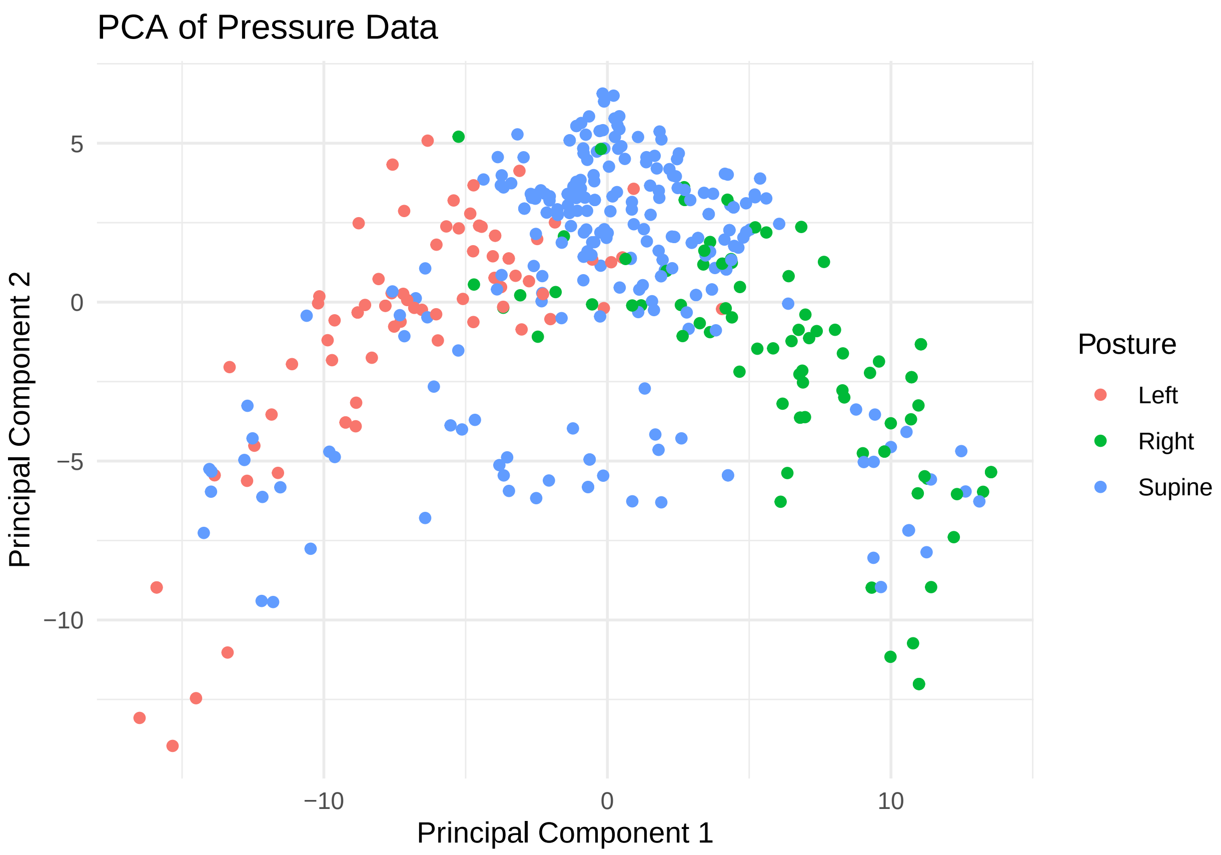


```r
# plot the Cumulative Proportion of Variance Explained
plot(cumsum(fit$sdev^2 / sum(fit$sdev^2)),
     type="b", xlab="Number of Principal Components",
     ylab="Cumulative Proportion of Variance Explained")
```



The bar plot suggests that the first two principal components explain a large portion of the variance, which could justify using just those for lower-dimensional representation or visualisation. The cumulative variance plot, however, shows that around 20-30 components are needed to explain 95% of the variance, which is typically considered a threshold for sufficient data representation in dimensionality reduction. Hence about 25 principle components would be required to represent the data sufficiently.

```r
# Compute PCA scores
pca_scores <- predict(fit)

# Dataframe for plotting 1st and 2nd pcs
pca_df <- data.frame(PC1 = pca_scores[,1],
                     PC2 = pca_scores[,2], Posture = pressure_data$Posture)

# Scatter plot of PC1 vs PC2 with Posture as color
ggplot(pca_df, aes(x = PC1, y = PC2, color = Posture)) +
  geom_point() + labs(title = "PCA of Pressure Data",
            x = "Principal Component 1", y = "Principal Component 2") +
  theme_minimal()
```



The scatter plot of Principal Component 1 (PC1) vs. Principal Component 2 (PC2), coloured by Posture, gives a good idea of how well these two components separate different sleeping postures.

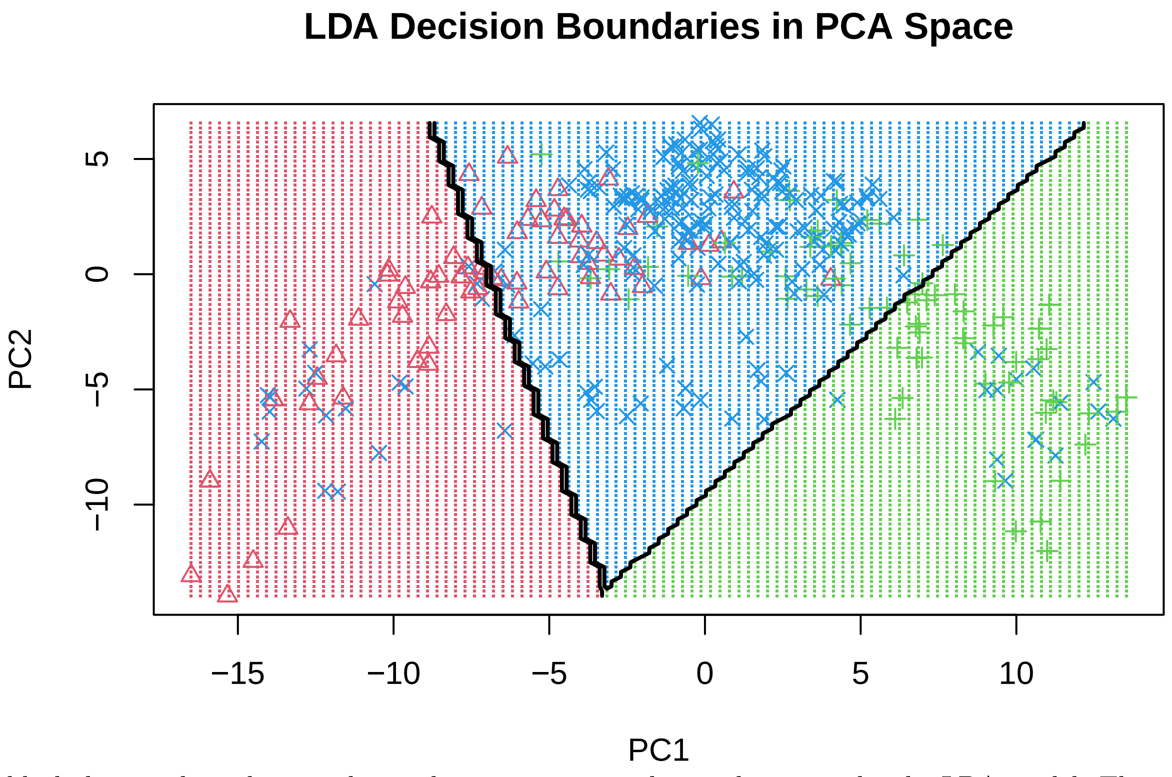
PC1 captures most of the variation between postures, which makes sense as it's the first principal component. "Left" (red) and "Right" (green) postures are spread out along PC1, meaning this component does a good job of distinguishing between side-sleeping postures. "Supine" (blue), however, is more clustered in the middle, suggesting PC1 doesn't influence it as much. PC2 provides some additional separation, but not as clearly. There's still some overlap, particularly between Supine (blue) and the other two postures, meaning PC2 alone doesn't fully separate them.

Overall: PC1 mainly differentiates Left vs. Right sleeping postures. PC2 adds some variation but doesn't fully separate Supine from the others. The first two PCs provide useful separation, but more components may be needed for a more accurate classification.

Question 6

```r
# Extract PC1 and PC2 from the PCA transformation
pca_df <- data.frame(PC1 = pca_scores[,1],
                     PC2 = pca_scores[,2], Posture = pressure_data$Posture)

# Train a new LDA model using PC1 and PC2
lda_pca <- lda(Posture ~ PC1 + PC2, data = pca_df)

# Define boundary function
boundary <- function(model, data, class = NULL, predict_type = "class",
                     resolution = 100, showgrid = TRUE, ...) {
```

```r
  if(!is.null(class)) cl <- data[,class] else cl <- 1
  data <- data[,1:2]
  k <- length(unique(cl))
  plot(data, col = as.integer(cl)+1L, pch = as.integer(cl)+1L, ...)
  # make grid
  r <- sapply(data, range, na.rm = TRUE)
  xs <- seq(r[1,1], r[2,1], length.out = resolution)
  ys <- seq(r[1,2], r[2,2], length.out = resolution)
  g <- cbind(rep(xs, each=resolution), rep(ys, time = resolution))
  colnames(g) <- colnames(r)
  g <- as.data.frame(g)
  ### guess how to get class labels from predict
  p <- predict(model, g, type = predict_type)
  if(is.list(p)) p <- p$class
  p <- as.factor(p)
  if(showgrid) points(g, col = as.integer(p)+1L, pch = ".")
  z <- matrix(as.integer(p), nrow = resolution, byrow = TRUE)
  contour(xs, ys, z, add = TRUE, drawlabels = FALSE,
          lwd = 2, levels = (1:(k-1))+.5)
  invisible(z)
}

# Call boundary function using the newly trained model
boundary(model = lda_pca, data = pca_df, class = "Posture",
         main = "LDA Decision Boundaries in PCA Space")
```



The black decision boundaries indicate the regions assigned to each posture by the LDA model. The coloured points represent the actual posture data in the reduced 2D space. The boundaries are linear, as expected from LDA, but some misclassification is visible, particularly around the class transitions.

Question 7

```r
# Extract PC1 and PC2 from the PCA transformation
pca_df <- data.frame(PC1 = pca_scores[,1],
                     PC2 = pca_scores[,2], Subject = pressure_data$Subject)

# Train a new LDA model using PC1 and PC2, make predictions and get metrics
lda_pca <- lda(Subject ~ PC1 + PC2, data = pca_df)
lda_predictions <- predict(lda_pca)$class
lda_full_metrics <- confusionMatrix(lda_predictions, pca_df$Subject)

# Extract overall and per-class statistics
lda_metrics <- list(
  overall = lda_full_metrics$overall,   # Overall statistics
  byClass = lda_full_metrics$byClass[, c("Sensitivity", "Specificity")]
)
print(lda_metrics)
```

```
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   1.675000e-01   4.855784e-02   1.322317e-01   2.077754e-01   1.325000e-01
## AccuracyPValue  McnemarPValue
##   2.597326e-02   4.078045e-14
##
## $byClass
##           Sensitivity Specificity
## Class: S1  0.62000000   0.7057143
## Class: S2  0.06000000   0.9542857
## Class: S3  0.09615385   0.8563218
## Class: S4  0.03773585   0.9164265
## Class: S5  0.20000000   0.8457143
## Class: S6  0.16326531   0.8945869
## Class: S7  0.17021277   0.8753541
## Class: S8  0.00000000   1.0000000
```

```r
# Train QDA model using PC1 and PC2, make predictions and get metrics
qda_pca <- qda(Subject ~ PC1 + PC2, data = pca_df)
qda_predictions <- predict(qda_pca)$class
qda_full_metrics <- confusionMatrix(qda_predictions, pca_df$Subject)

# Extract overall and per-class statistics
qda_metrics <- list(
  overall = qda_full_metrics$overall,   # Overall statistics
  byClass = qda_full_metrics$byClass[, c("Sensitivity", "Specificity")]
)

# Print the modified output
print(qda_metrics)
```

```
## $overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   2.300000e-01   1.199183e-01   1.896388e-01   2.744081e-01   1.325000e-01
## AccuracyPValue  McnemarPValue
##   7.734087e-08   4.059290e-15
##
```

```
## $byClass
##            Sensitivity Specificity
## Class: S1  0.60000000   0.7857143
## Class: S2  0.54000000   0.7428571
## Class: S3  0.05769231   0.8965517
## Class: S4  0.11320755   0.9538905
## Class: S5  0.22000000   0.8885714
## Class: S6  0.16326531   0.9373219
## Class: S7  0.12765957   0.9263456
## Class: S8  0.02040816   0.9886040
```

PCA Approach: Since the data was standardised before applying PCA, we used the correlation matrix rather than the covariance matrix. The correlation matrix is just a scaled version of the covariance matrix, where each covariance value is divided by the product of the standard deviations of the respective variables.

Using the correlation matrix was more appropriate because the pressure data consists of multiple sensor readings which are on different scales. If we used the covariance matrix, the variables with higher variance would have dominated the principal components, potentially skewing the results. By using the correlation matrix, we ensured that all variables contributed equally to the PCA.

Model Performance Comparison:

LDA Performance: Accuracy: 16.75% (95% CI: 13.22% – 20.78%) LDA struggled to classify subjects accurately, with a great deal of misclassification. Sensitivity and specificity varied across subjects, with Class S1 performing the best, but most other classes had low sensitivity. Kappa statistic (0.0486) suggests the model was only slightly better than random guessing.

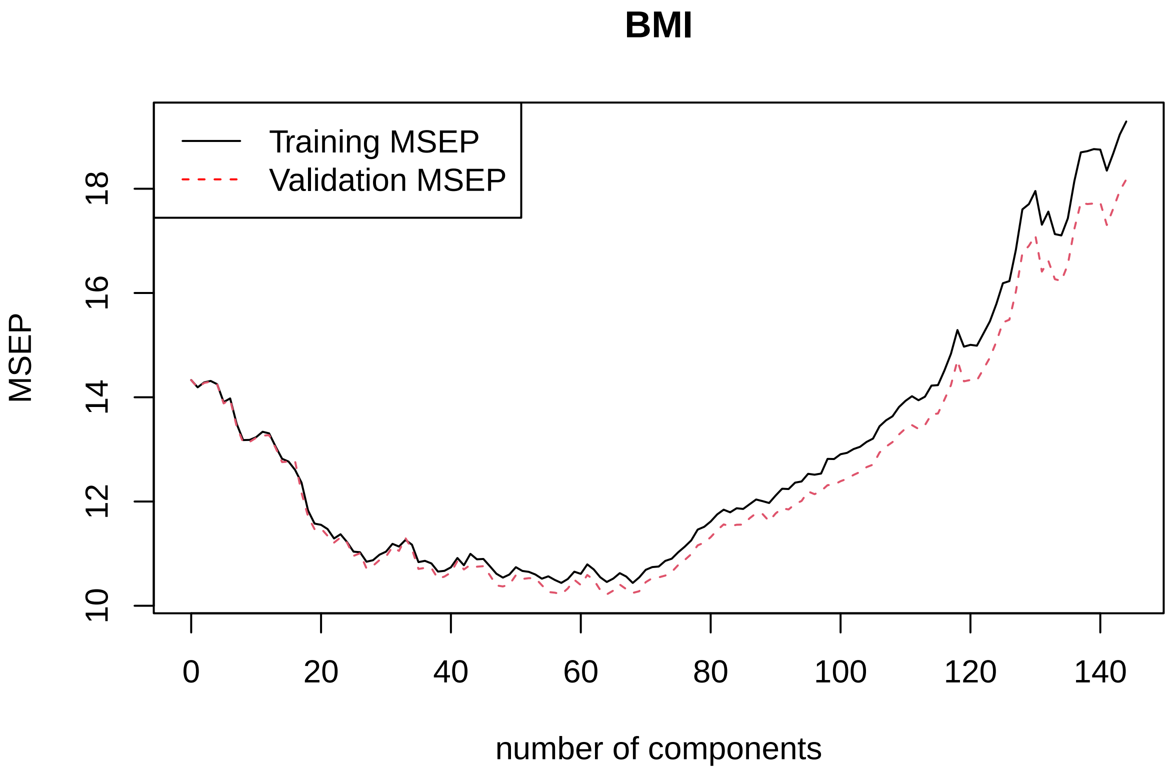
QDA Performance: Accuracy: 23.00% (95% CI: 18.96% – 27.44%) QDA performed slightly better than LDA, with improved accuracy and better class separation. Some classes (e.g. S7 and S8) still had very low sensitivity, meaning they were rarely predicted correctly. Kappa statistic (0.1199) is still low but suggests QDA provided better class separation than LDA.

Conclusion: Overall, QDA worked better than LDA, but both models had pretty low accuracy. Using PCA with the correlation matrix was definitely the right choice because of the differences in variable scales. Even after reducing dimensionality, classifying subjects remains difficult. Perhaps we might need more features or maybe even a different classification approach.

Question 9

```r
# import the subject info data
subject_info_df <- read.csv("Subject_Info_Data.csv")

# Compute BMI from height and weight ( Weight kg divided by (Height m)^2 )
subject_info_df <- subject_info_df %>%
  mutate(Height.m = Height.cm / 100, BMI = Weight.kg / (Height.m^2))

# Subject number is in the wrong format e.g. 2 -> S2
subject_info_df <- subject_info_df %>%
  mutate(Subject = paste0("S", Subject.Number))

# Merge BMI into pressure dataset using Subject no.
pressure_data <- merge(pressure_data, subject_info_df[, c("Subject","BMI")],
                       by.x = "Subject", by.y = "Subject")

# Define predictors (V1 - V144) and response (BMI)
X <- pressure_data[, 2:145] # predictors X
```

```r
Y <- pressure_data$BMI # response Y

# test and train set 80:20 split
train_indices <- sample(1:nrow(pressure_data), 0.8 * nrow(pressure_data))

X_train <- X[train_indices, ]
Y_train <- Y[train_indices]
X_test <- X[-train_indices, ]
Y_test <- Y[-train_indices]

# Merge BMI into the training dataset
train_data <- as.data.frame(cbind(BMI = Y_train, X_train))

# Fit PCR model with cross-validation
pcr_model <- pcr(BMI ~ ., data = train_data, scale = TRUE, validation = "CV")

# Did not include summary(pcr_model) as output is far too
# bulky with cross validation and all variables

# Plot variance explained to find optimal number components
validationplot(pcr_model, val.type = "MSEP")
legend("topleft", legend = c("Training MSEP", "Validation MSEP"),
       col = c("black", "red"), lty = c(1, 2))
```



Because a lot of the variables are highly correlated I used cross-validation (CV) in training to improve generalisation and hopefully reduce risk of overfitting.

MSEP (Mean Squared Error of Prediction) measures how well a model predicts new data. It is calculated as the average squared difference between predicted and actual values. Lower MSEP values indicate better model performance, while higher values suggest poor predictive accuracy.

From the MSEP plot we can see: 0-20 components: MSEP decreases rapidly, indicating improved model performance. 20-70 components: MSEP stabilises, suggesting this is the optimal range. 70 and above components: MSEP increases, indicating overfitting. Hence the optimal number of components should be around 40-60.

```r
# MSEP values from the model
msep <- MSEP(pcr_model)$val[1,1,]

# Select the number of components with lowest MSEP
num_pcs <- which.min(msep)
cat("Optimal Number of Components:",  num_pcs,"\n")
```

```
## Optimal Number of Components: 69
```

```r
# Define the test dataframe and predict responses (BMI)
test_data <- as.data.frame(X_test)
Y_pred <- predict(pcr_model,newdata = test_data,  ncomp =num_pcs)

# RMSE (Root Mean Squared Error)
pcr_rmse <- sqrt(mean((Y_pred - Y_test)^2) )
cat("PCR Test RMSE:",round(pcr_rmse,  3), "\n")
```

```
## PCR Test RMSE: 2.984
```

The MSEP plot indicated that increasing the number of components initially improved performance, but beyond 70 components, validation error increased due to overfitting. The optimal number of components retrieved was 69.

Model Performance Evaluation: The RMSE of 2.984 means that, on average, the predicted BMI values deviate by about 2.98 BMI units from the actual values. Given that BMI in the dataset ranges between 22 and 32, the model performs reasonably well, though there is still some prediction error. While it generalises fairly well to new data, performance could be enhanced with better feature selection, alternative dimensionality reduction techniques, or a more advanced regression model.

Conclusion: PCR was useful for handling lots of variables and reducing overfitting. Still, accuracy could be improved. Trying something like Random Forest Regression might help, especially since it can deal with non-linear patterns better.

Question 8

(i) Purpose of PCR

Principal Components Regression (PCR) is a dimensionality reduction technique designed to handle multicollinearity in regression models. When predictor variables are highly correlated, regression methods can lead to unstable coefficient estimates and overfitting. PCR solves this problem by converting the original variables into independent components (PCs) that serve as new predictors.

(ii) How PCR Works

PCR consists of two main steps. The first is Principal Component Analysis (PCA), where the original predictors are transformed into a new set of orthogonal (uncorrelated) variables called principal components (PCs). These PCs are ranked in descending order by the amount of variance they explain in the original dataset. The second step is Regression on Principal Components, where instead of using all the original variables, a smaller set of PCs is selected. An ordinary least squares (OLS) regression is then fitted using these selected PCs as predictors.

(iii) Key Choices in PCR

Choosing the right number of components (k) is key to making PCR work well. If too few PCs are used, the model might miss important patterns (underfitting). On the other hand, using too many can introduce noise, making the model struggle with new data (overfitting). The optimal number of components is usually determined through cross-validation or by examining the cumulative variance explained. The scaling of predictors is another important choice to be made. If the predictors have different scales, when the covariance matrix is calculated, predictors with larger magnitudes may dominate as they will have larger variances. It's usually a good idea to standardise them (set mean = 0, variance = 1) before running PCA. Finally, validation methods, such as cross-validation, are used to assess how many PCs should be retained for the best predictive performance.

(iv) Advantages and Disadvantages of PCR

PCR has several advantages, especially in high-dimensional datasets where multicollinearity is a big issue. By transforming correlated predictors into independent PCs, it helps prevent regression models from overfitting by removing unnecessary variance. It also makes the model far more simple by reducing the number of predictors while still keeping most of the data's variability. However, PCR does have disadvantages too. Because principal components mix different variables together, it’s harder to interpret them. Another drawback is that PCR chooses components based on variance, but this doesn't always mean they are the best predictors for the target variable. As a result, important information may be discarded if it isn’t captured within the first few principal components.

Conclusion

PCR is a useful method for regression in datasets with high multicollinearity, such as pressure measurements predicting BMI. Choosing the right number of components is important to avoid underfitting or overfitting. While PCR is effective for dimensionality reduction, it does not always guarantee the best predictive performance, especially when relationships between predictors and the response variable are not well represented in the selected components.