CA1

AUTHOR
Sam Ryder, 18317496

Libraries

```r
suppressPackageStartupMessages({
  library(MASS)
  library(pls)
})
```

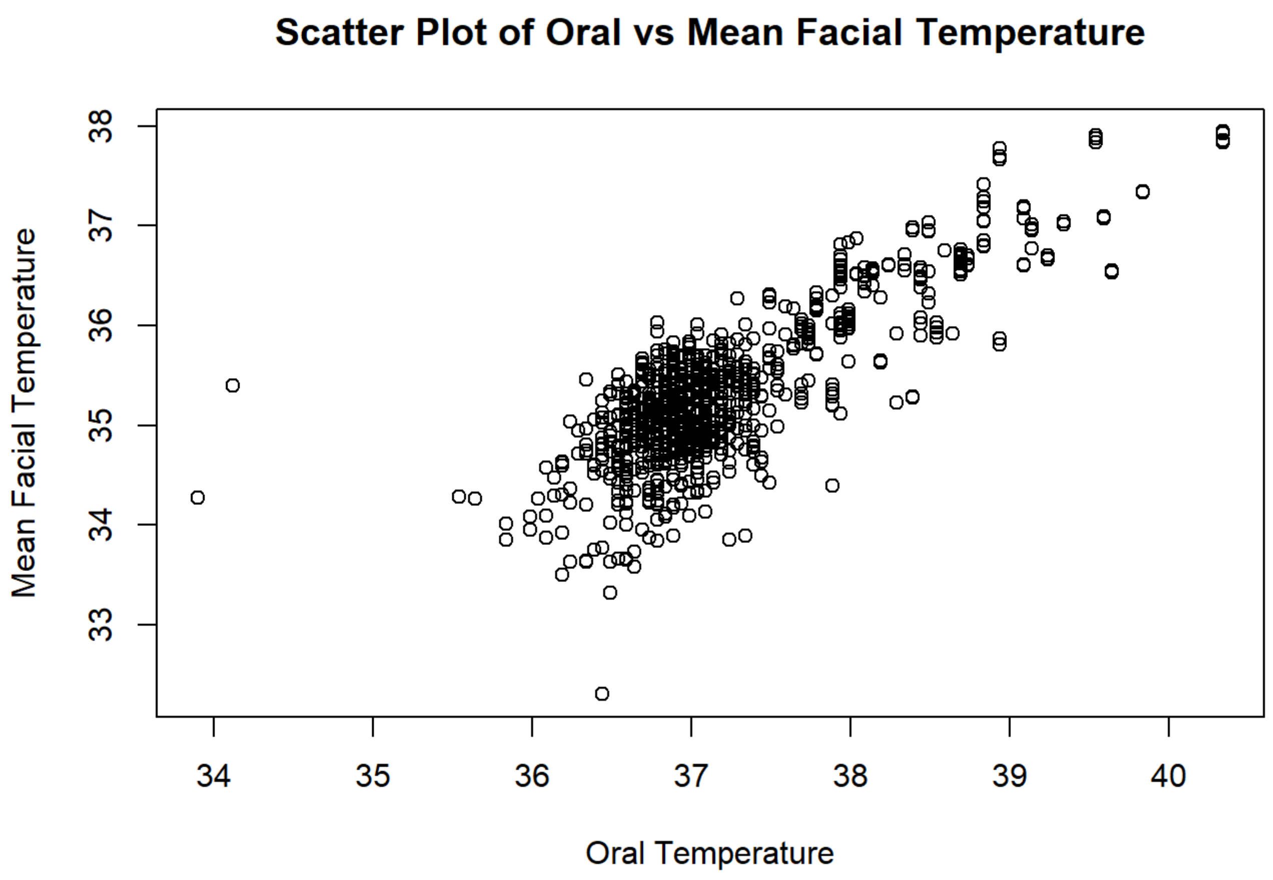
```
Warning: package 'MASS' was built under R version 4.3.2

Warning: package 'pls' was built under R version 4.3.3
```

Data set up

```r
x = read.csv("Temperature_data.csv")
set.seed(18317496)
#Random subset of 1000 observations
random_rows <- sample(nrow(x), 1000)
data_subset <- x[random_rows, ]
```

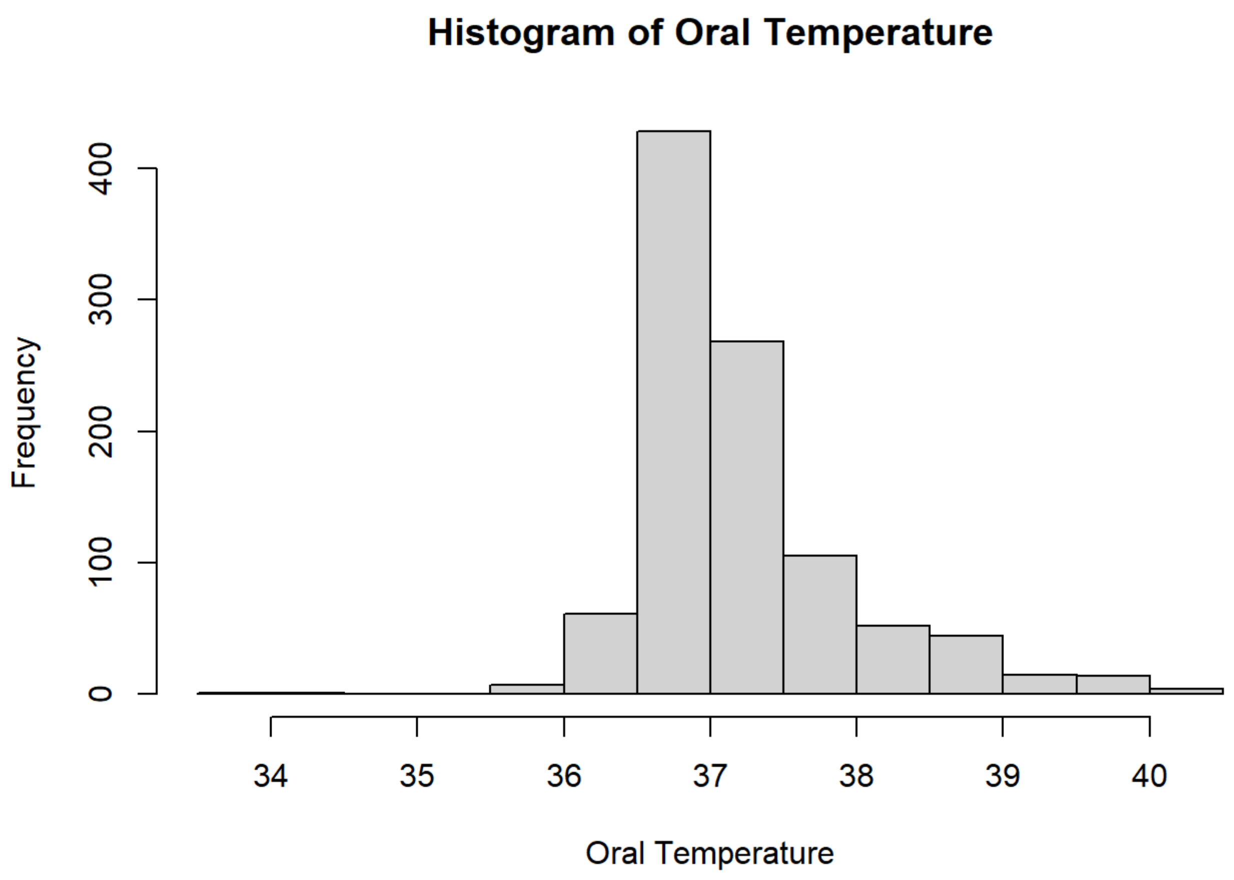
Data cleaning + Visualisation

```r
#Remove data rows with NA for variable oral temp
data <- data_subset[!is.na(data_subset$aveOralM), ]
# Calculate mean facial temperature for each observation
mean_facial_temp <- rowMeans(data[, 1:10])
#Plot oral temperature vs facial temperature
plot(data[, 11], mean_facial_temp,
     xlab = "Oral Temperature", ylab = "Mean Facial Temperature",
     main = "Scatter Plot of Oral vs Mean Facial Temperature",
     xlim = range(data[, 11], na.rm = TRUE))
```



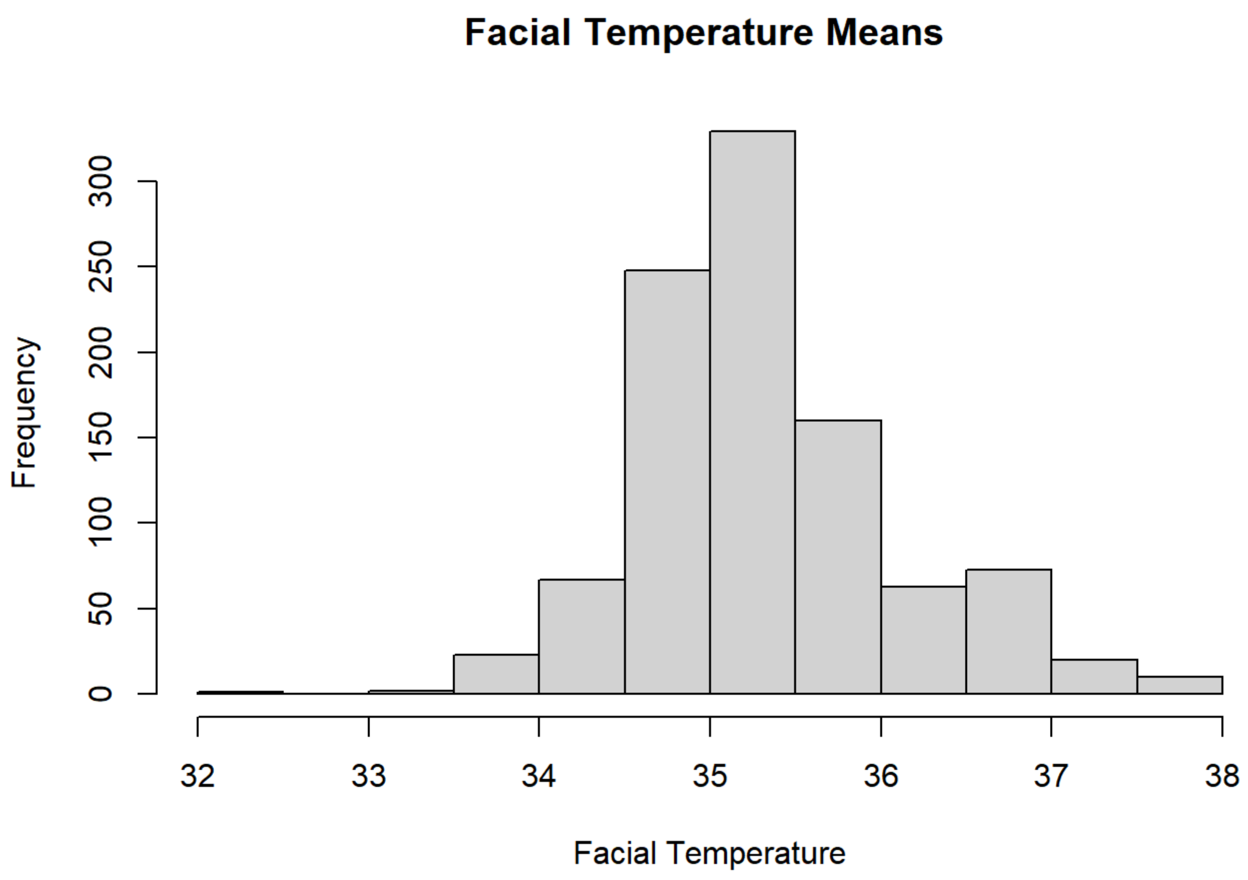
The plot above shows a positive correlation between oral temperature and the mean facial temperature measurement. The majority of measurements for oral temperature are situated between 36 and 38, meanwhile for facial temperature the most common range is 34-36, this indicates the facial temperature is generally lower than the oral temperature. There are however a number of outliers towards high values for both measurements as well as 2 very low measurements for the oral temperature.

```r
#Histogram of oral and facial temp
hist(data$aveOralM, main = "Histogram of Oral Temperature", xlab = "Oral Temperature")
```



The histogram above shows that values for the oral temperature are positively skewed.

```
hist(mean_facial_temp, main = "Facial Temperature Means ", xlab = "Facial Temperature")
```



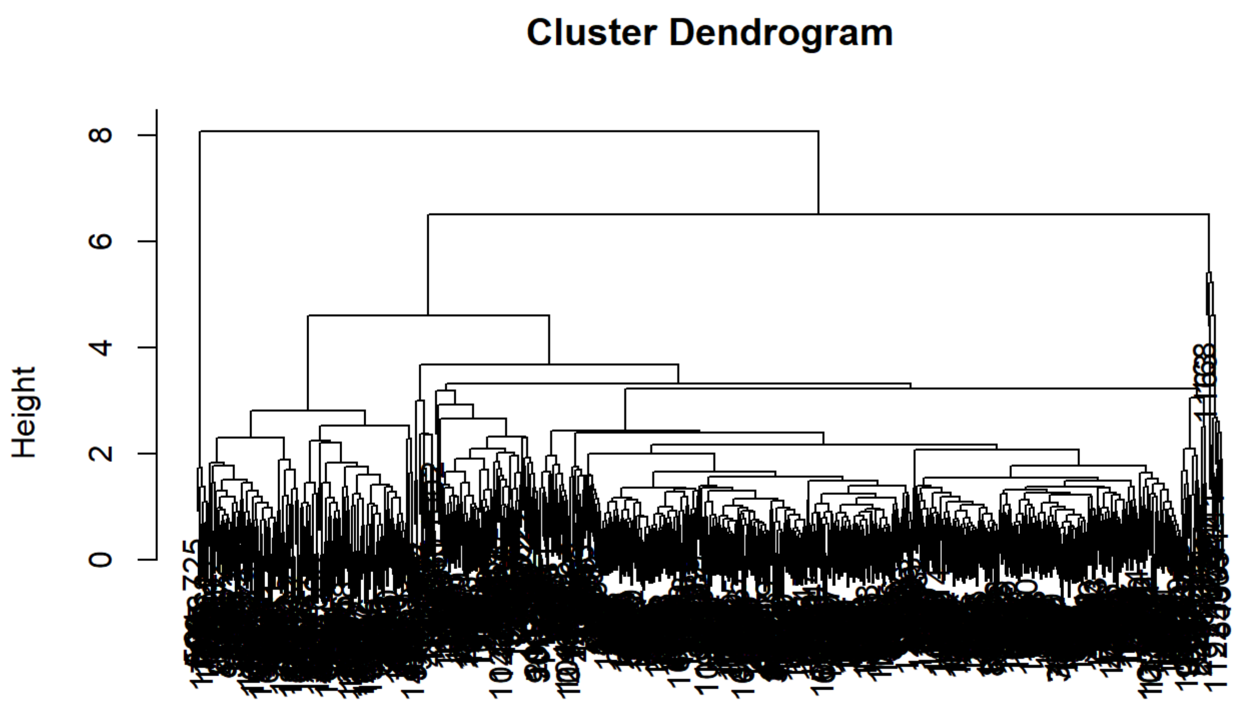
The histogram above shows that values for the facial temperature are relatively evenly distributed with possibly a slight positively skewed.

Outliers are removed as they may otherwise negatively affect later analysis.

```
#Remove observations 4 standard deviations below mean
mean_oral_temp <- mean(data$aveOralM)
sd_oral_temp <- sd(data$aveOralM)
lower_bound <- mean_oral_temp - 4 * sd_oral_temp
data <- data[data$aveOralM >= lower_bound, ]
```

Q3: Clustering

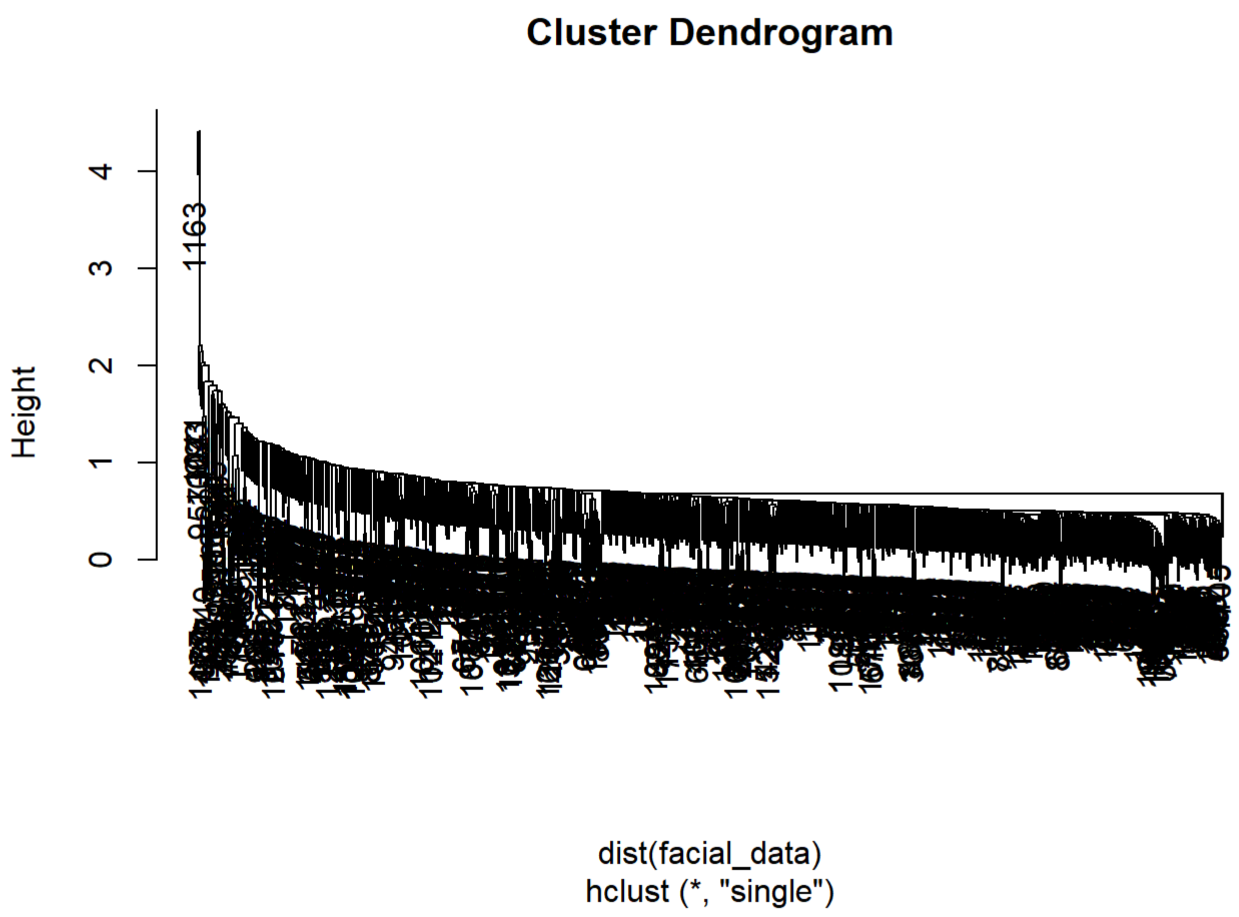
```
#Hierarchical clustering on facial temp measurements
facial_data <- data[, 1:10]
facial_data <- as.matrix(facial_data)
#Average linkage
cl.average = hclust(dist(facial_data), method="average")
plot(cl.average, xlab="Average linkage", sub="")
```



Average linkage

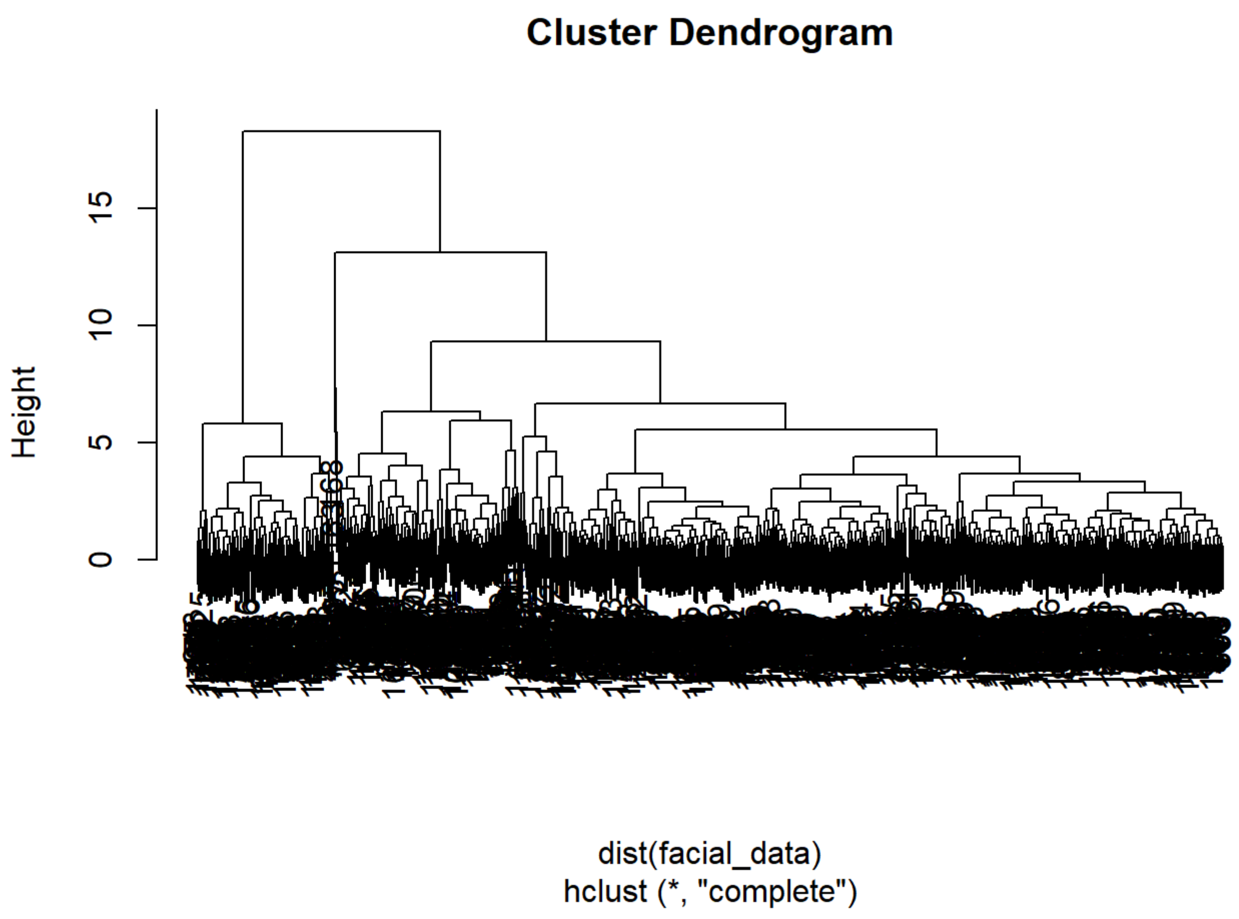
When using an average linkage approach it would appear 3 clusters is optimum, as there is a clear gap until these lines diverge again on the graph.

```
#Single linkage
cl.single = hclust(dist(facial_data), method="single")
plot(cl.single)
```



A single linkage approach does not appear to be suitable in this case as there are no distinct clusters or hierarchical structure.

```
#Complete linkage
cl.complete = hclust(dist(facial_data), method="complete")
plot(cl.complete)
```

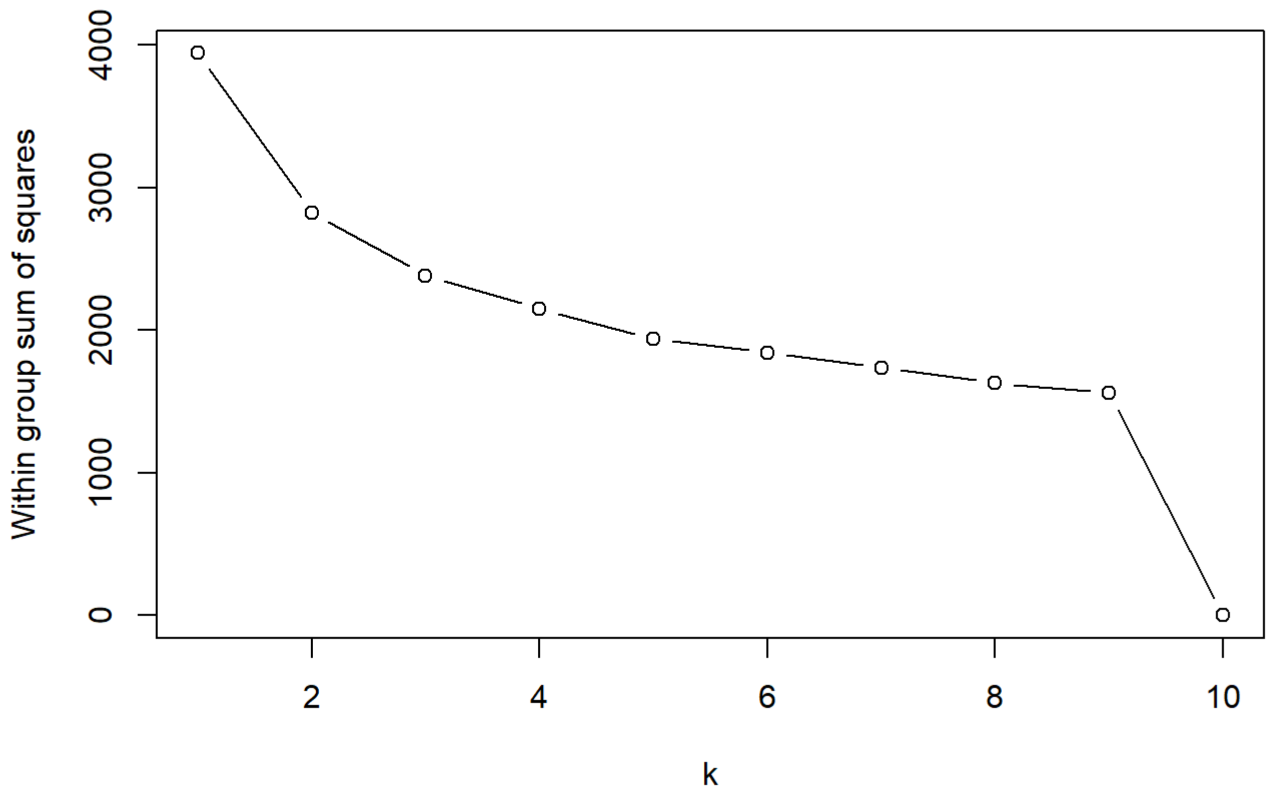


Complete Linkage does not seem to be a suitable approach in this case as there are no distinct clusters or hierarchical structure.

```r
#3. kmeans clustering
facial_data_imputed <- facial_data
for (i in 1:ncol(facial_data)) {
  col_mean <- mean(facial_data[, i], na.rm = TRUE)
  facial_data_imputed[is.na(facial_data[, i]), i] <- col_mean
}

WGSS <- numeric(10)
# Perform k-means clustering for each value of k from 2 to 10
for (k in 2:10) {
  kmeans_result <- kmeans(facial_data_imputed, centers = k)
  WGSS[k - 1] <- sum(kmeans_result$withinss)
}
WGSS
```

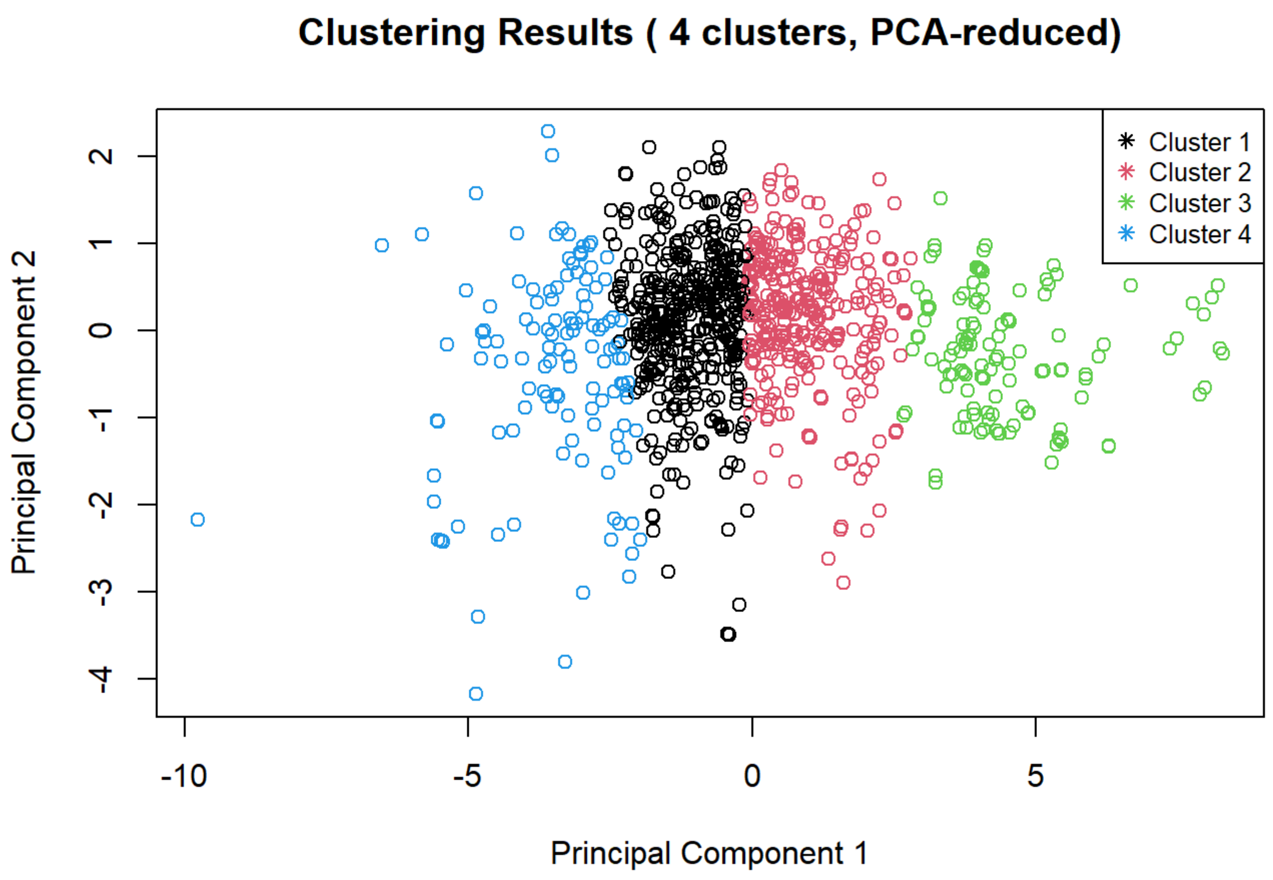
```
[1] 3940.719 2824.552 2380.678 2149.166 1938.343 1840.604 1734.529 1629.186
[9] 1560.565    0.000
```

```r
# Plot WGSS vs k
plot(1:10, WGSS, type="b", xlab="k", ylab="Within group sum of squares")
```



When reading the graph above it would appear the optimal number of clusters is 4. This is due to the fact the error curve flattens from a steep descent to a more gradual decline at this point. The kmeans clustering algorithm is run with this in mind.

```
#Clustering with optimal k(4)
optimal_k <- 4
kmeans_result <- kmeans(facial_data_imputed, centers = optimal_k)
# Plot kmeans clusters using pca to visualize
facial_data_pca <- prcomp(facial_data_imputed)
plot(facial_data_pca$x[,1:2], col = kmeans_result$cluster,
     main = paste("Clustering Results (", optimal_k, "clusters, PCA-reduced)"),
     xlab = "Principal Component 1", ylab = "Principal Component 2")

# Add cluster centers to the plot
points(kmeans_result$centers[, 1:3], col = 1:optimal_k, pch = 8, cex = 2)

# Add legend
legend("topright", legend = paste("Cluster", 1:optimal_k), col = 1:optimal_k, pch = 8, cex = 0
```



The kmeans clustering algorithm is run on data that has gone through principal component analysis to allow for easier visualisation. The algorithm seems to have performed reasonably well as the graph shows clear distinction between the 4 clusters.

When comparing the best performing hierarchical clustering model (Average) and a kmeans clustering algorithm the kmeans approach is prefferred. This is because there are clear boundaries between values of different classes, with 4 distinct clusters versus relatively non-distinct hierarchical clusters.

Q4: Discriminant Analysis

For discriminant analysis, the data is split into training and testing datasets. Linear and Quadratic Discriminant Analyses models are then fit.

```
#DA data
da_data <- data[,1:12]
n <- nrow(da_data)
train_indices <- sample(1:n, 0.7 * n)  # 70% for training
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]
train_predictors <- train_data[, 1:11]
train_target <- train_data[, 12]
test_predictors <- test_data[, 1:11]
test_target <- test_data[, 12]
#Check for missing values
complete_cases <- complete.cases(test_predictors)
test_predictors_complete <- test_predictors[complete_cases, ]
```

```
target_complete <- test_target[complete_cases]
#LDA/QDA fitting
lda_res <- lda(train_target  ~ ., data = train_predictors)
qda_res <- qda(train_target  ~ ., data = train_predictors)
#LDA/QDA Predictions
lda_predictions <- predict(lda_res, test_predictors_complete)
qda_predictions <- predict(qda_res, test_predictors_complete)
# Summary of LDA results
table(lda_predictions$class, target_complete)
```

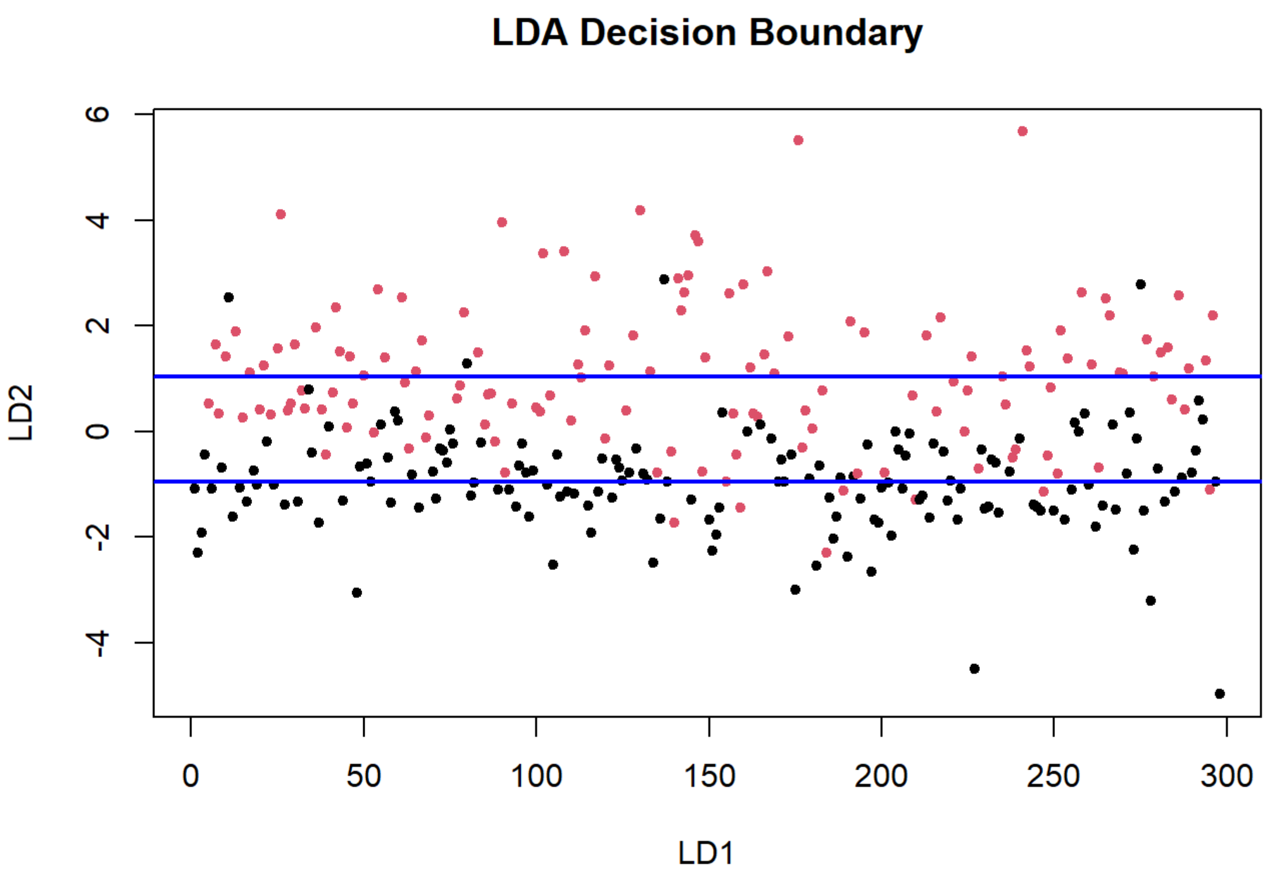
```
        target_complete
         Female Male
  Female    148   33
  Male       10  107
```

```
# Summarize QDA predictions
table(qda_predictions$class, target_complete)
```

```
        target_complete
         Female Male
  Female    146   36
  Male       12  104
```

The tables above show that the LDA approach performed better with a higher proportion of correctly identified cases. It is then used to plot the decision boundary.

```
#LDA plot
# Get discriminant scores for each class
scores <- predict(lda_res, test_predictors_complete)$x
# Plot data points
plot(scores[,1],col = as.factor(target_complete), pch = 20,
     main = "LDA Decision Boundary", xlab = "LD1", ylab = "LD2")
# Get coefficients of the linear discriminant function
coefficients <- lda_res$scaling
# Define the equation of the decision boundary (hyperplane)
intercept <- lda_res$scaling[1] * lda_res$means[, 1] / lda_res$scaling[2] - lda_res$means[, 2]
# Get the average LD2 value for each class
mean_ld2 <- tapply(scores[, 1], as.factor(target_complete), mean)
abline(h = mean_ld2, col = "blue", lwd=2)
```



The plot above shows a reasonably well performing model. The boundaries on either side have done a pretty good job of splitting the data with the classification boundary containing relatively equal measures of both classes.

Q5&6: PCA

```
#5 PCA
pca_data <- data[,1:11]
any(is.na(pca_data))
```

```
[1] TRUE
```

```
# Remove rows with missing values
pca_data <- pca_data[complete.cases(pca_data), ]
#Fit pca
fit = prcomp(pca_data[,1:11])
fit
```

```
Standard deviations (1, .., p=11):
 [1] 2.46286960 0.92984030 0.54891937 0.49362195 0.42591164 0.32873963
 [7] 0.31004165 0.23566249 0.17756561 0.09192315 0.07160642

Rotation (n x k) = (11 x 11):
                  PC1        PC2          PC3          PC4            PC5
Max1R13_1   0.3011891 -0.2062789  0.037709798  0.179355777  0.1057933996
```

```
T_RC1       0.2915470 -0.1987285  0.008193256  0.067302003  0.2141642475
T_LC1       0.2717518 -0.1749278 -0.022233172 -0.040970672  0.3031445513
RCC1        0.3116073 -0.1976376  0.027829000  0.227105309  0.0934183565
canthiMax1  0.2786257 -0.1956715 -0.002202995 -0.003558138  0.2663706632
T_FHCC1     0.3543140  0.4384718 -0.526375108  0.068809708 -0.0008399217
T_FHLC1     0.3067869  0.3942812 -0.385112762 -0.060277319 -0.0030026526
T_FHTC1     0.3329579  0.5651515  0.736718569 -0.131878762  0.0222912292
T_OR1       0.2573996 -0.2378811 -0.078767924 -0.724635029 -0.5228274367
aveAllR13_1 0.3432540 -0.1824521  0.128381210  0.503206449 -0.6458899607
aveOralM    0.2472036 -0.2302624  0.080089051 -0.325516625  0.2834351831
                    PC6         PC7          PC8          PC9          PC10
Max1R13_1    0.047509067  0.12315870  0.063545629 -0.508098717  6.998674e-01
T_RC1        0.069075503  0.16278433 -0.007377596 -0.490151811 -4.220127e-01
T_LC1        0.188675932  0.12736187 -0.527262501  0.543487388  3.005698e-01
RCC1         0.093838767  0.17089885  0.754744535  0.438362511 -5.416524e-02
canthiMax1   0.101711192  0.14192838 -0.261262047 -0.033479772 -4.773810e-01
T_FHCC1     -0.569436802  0.27002932 -0.043188428  0.035825233  8.663533e-03
T_FHLC1      0.617756602 -0.45375695  0.075999497 -0.070747014 -3.617912e-03
T_FHTC1     -0.009342389  0.09487945 -0.001632324  0.002742253  6.736017e-06
T_OR1        0.076619666  0.24977493  0.059296586 -0.011249193  2.452729e-02
aveAllR13_1 -0.094362317 -0.27910822 -0.242875717  0.077770794 -9.828555e-02
aveOralM    -0.466345777 -0.68165963  0.099277799  0.021326533  2.392461e-02
                    PC11
Max1R13_1   -2.291494e-01
T_RC1        6.126795e-01
T_LC1        2.931477e-01
RCC1        -3.870835e-03
canthiMax1  -6.966702e-01
T_FHCC1     -4.221861e-05
T_FHLC1     -1.253984e-02
T_FHTC1     -2.614454e-04
T_OR1       -2.627201e-03
aveAllR13_1  2.265010e-02
aveOralM     1.171511e-02
```

```r
names(fit)
```

```
[1] "sdev"     "rotation" "center"   "scale"    "x"
```

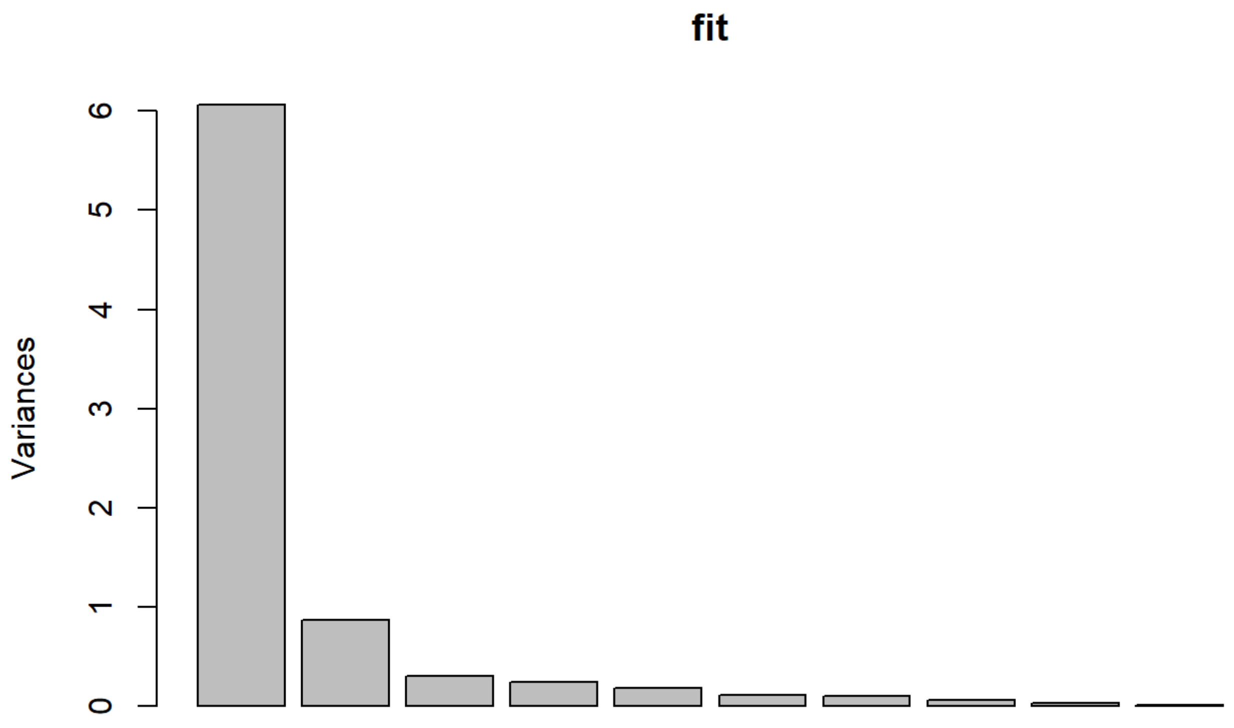
```r
summary(fit)
```

```
Importance of components:
                          PC1    PC2     PC3    PC4     PC5     PC6     PC7
Standard deviation     2.4629 0.9298 0.54892 0.4936 0.42591 0.32874 0.31004
Proportion of Variance 0.7619 0.1086 0.03785 0.0306 0.02278 0.01357 0.01207
Cumulative Proportion  0.7619 0.8705 0.90832 0.9389 0.96171 0.97529 0.98736
                           PC8     PC9    PC10    PC11
Standard deviation     0.23566 0.17757 0.09192 0.07161
Proportion of Variance 0.00698 0.00396 0.00106 0.00064
Cumulative Proportion  0.99433 0.99829 0.99936 1.00000
```

```r
round(fit$rotation, 2)
```

```
             PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8   PC9  PC10  PC11
Max1R13_1   0.30 -0.21  0.04  0.18  0.11  0.05  0.12  0.06 -0.51  0.70 -0.23
T_RC1       0.29 -0.20  0.01  0.07  0.21  0.07  0.16 -0.01 -0.49 -0.42  0.61
T_LC1       0.27 -0.17 -0.02 -0.04  0.30  0.19  0.13 -0.53  0.54  0.30  0.29
RCC1        0.31 -0.20  0.03  0.23  0.09  0.09  0.17  0.75  0.44 -0.05  0.00
canthiMax1  0.28 -0.20  0.00  0.00  0.27  0.10  0.14 -0.26 -0.03 -0.48 -0.70
T_FHCC1     0.35  0.44 -0.53  0.07  0.00 -0.57  0.27 -0.04  0.04  0.01  0.00
T_FHLC1     0.31  0.39 -0.39 -0.06  0.00  0.62 -0.45  0.08 -0.07  0.00 -0.01
T_FHTC1     0.33  0.57  0.74 -0.13  0.02 -0.01  0.09  0.00  0.00  0.00  0.00
T_OR1       0.26 -0.24 -0.08 -0.72 -0.52  0.08  0.25  0.06 -0.01  0.02  0.00
aveAllR13_1 0.34 -0.18  0.13  0.50 -0.65 -0.09 -0.28 -0.24  0.08 -0.10  0.02
aveOralM    0.25 -0.23  0.08 -0.33  0.28 -0.47 -0.68  0.10  0.02  0.02  0.01
```

```
#PC variance
plot(fit)
```



The plot above displays the depreciating nature of the cumulative variance contained in the Priciniple components. The first Principle Component contains a very large proportion of the variance in the data, followed by the second principle component containing a reasonable proportion of data variance. Values then depreciate from then with regularity as expected. As the proportion explained begins to level off after 3 Principle Components, it is brought forward in the analysis.

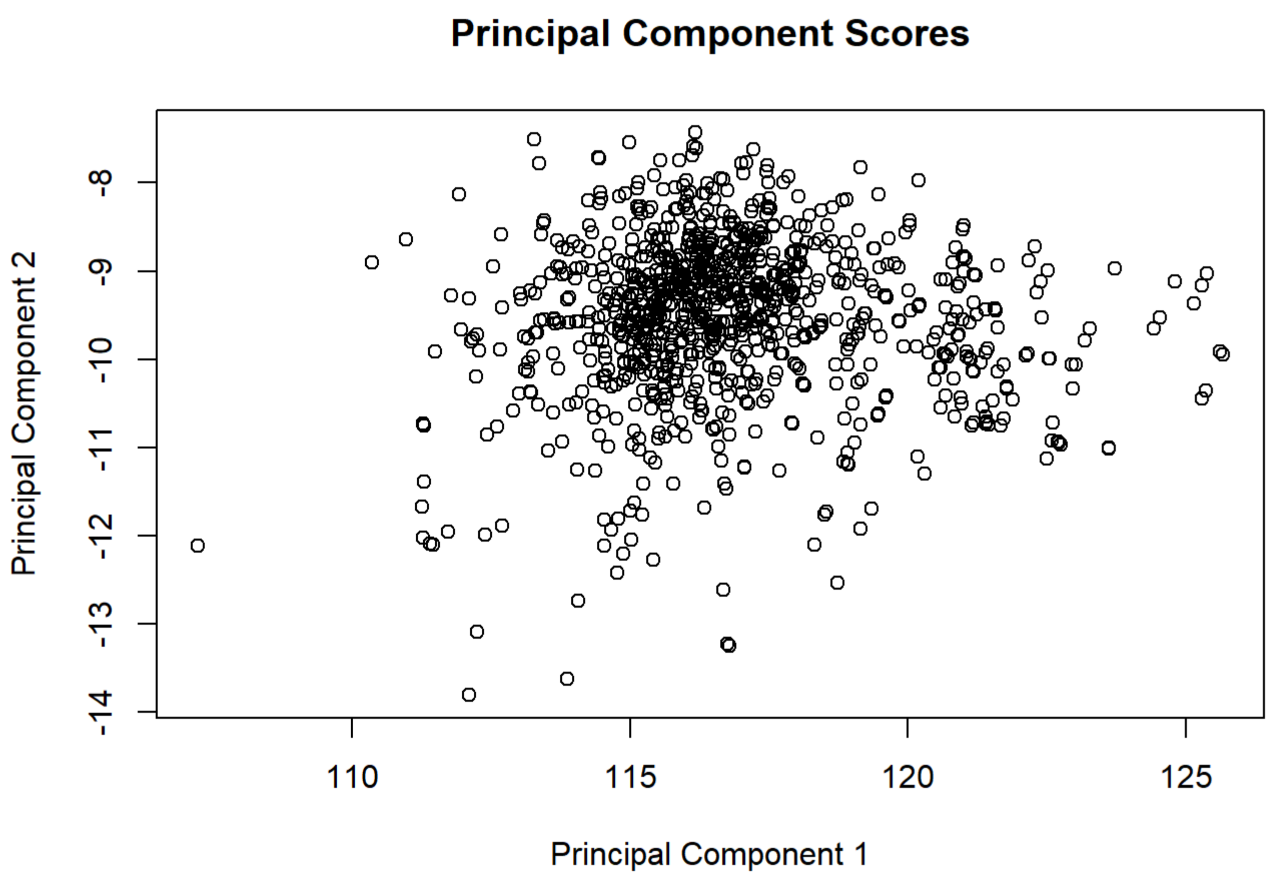
```
#3PCs
num_components <- 3
selected_components <- fit$rotation[, 1:num_components]
selected_components <- t(selected_components)
# Calculate principal component scores
```

```
pca_data <- as.matrix(pca_data)
#Derive principal componenet scores from first principle
principal_scores <- pca_data %*% t(selected_components)
head(principal_scores)
```

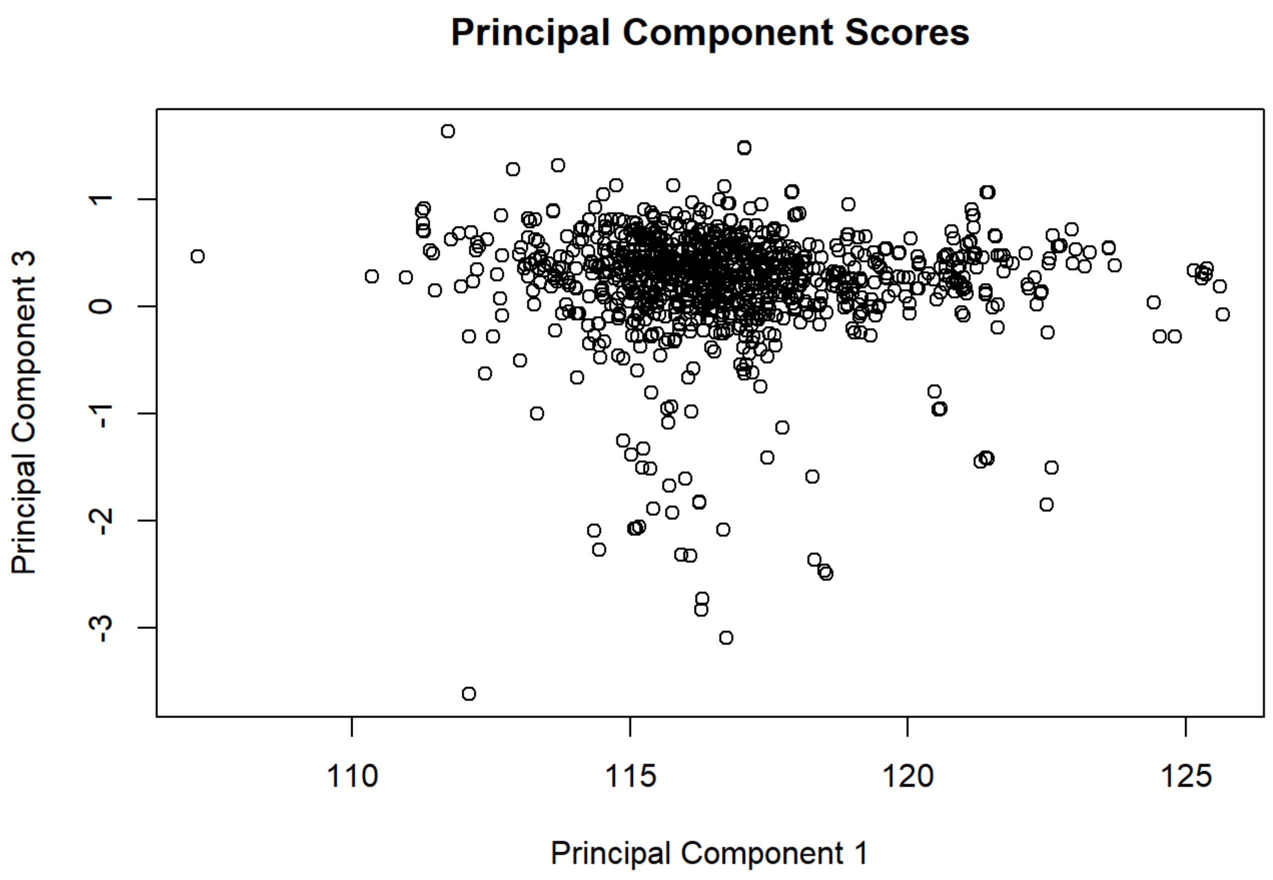
```
          PC1        PC2         PC3
464  115.4046  -9.255988  0.32576295
138  120.8088  -8.898441  0.21730823
980  116.7791 -13.244634  0.96001515
216  121.2164  -9.051382  0.49642333
436  116.6728  -7.965189 -0.05483104
1192 115.7207  -9.141344  0.51958598
```

The sample dataframe above shows the depreciating nature of the principle components. As the principle components increase the contribution to the variance decreases.

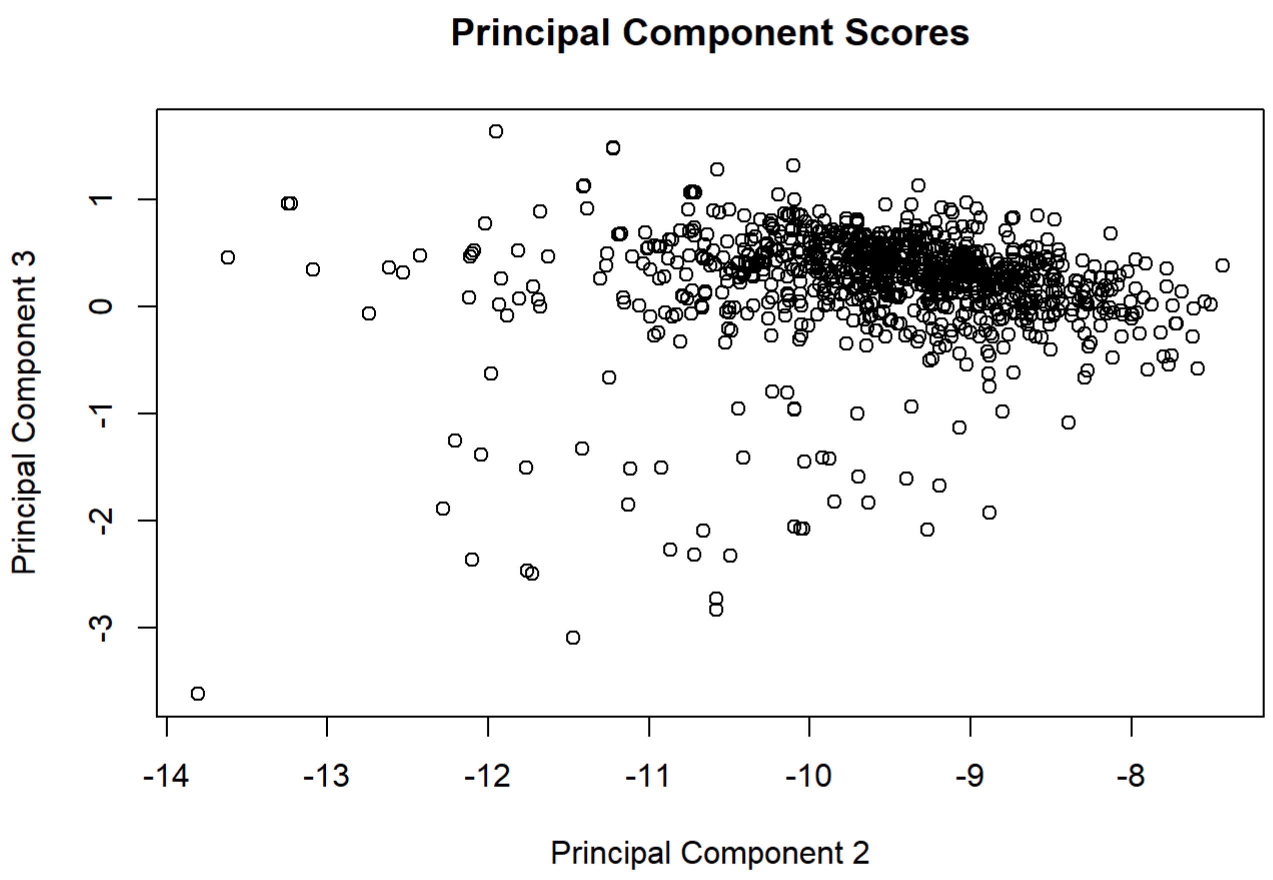
```
# Plot the principal component scores
plot(principal_scores[, 1], principal_scores[, 2],
     xlab = "Principal Component 1", ylab = "Principal Component 2",
     main = "Principal Component Scores")
```



```
plot(principal_scores[, 1], principal_scores[, 3],
     xlab = "Principal Component 1", ylab = "Principal Component 3",
     main = "Principal Component Scores")
```



```r
plot(principal_scores[, 2], principal_scores[, 3],
     xlab = "Principal Component 2", ylab = "Principal Component 3",
     main = "Principal Component Scores")
```



Question 7: PCR research

Purpose:

Principle component analysis addresses the fact that ‘often a small number of principal components suffice to explain most of the variability in the data, as well as the relationship with the response’ (James et al., 2021). PCA therefore allows us to reduce the dimensionality of the data while retaining its variability. Dimension reduction is often a necessary step because sometimes ‘there are many, possibly correlated, predictor variables, and relatively few samples’ (Larose and Larose, 2019). This problem, if unaddressed can lead to slow processing times and machine learning models aversely affected by data collinearity. PCA addresses the problem of large feature spaces and collinearity by ‘using a smaller set of uncorrelated linear combination of these variables called components’ (Mevik and Wehrens, 2007).

How PCR works:

Principle components regression focuses on the predictors with high variance, as these are likely to be the most informative in machine learning tasks. It then ‘discards’ those predictors that do not fulfil the high variability threshold (Hastie, Tibshirani and Friedman, 2017). Ultimately, the PCR approach aims to replace the original predictors with a reduced number of principal components which encapsulate data variability but allow for faster processing times and use these components as predictors in a linear regression model (James et al., 2021). PCR works by decomposing X into Orthogonal scores (T) and loadings (P) where X = TP. Y is then regressed on the scores T rather than X (Larose and Larose, 2019). This ultimately means that the k Principal Components contain as much information as the original m variables in a reduced feature space and therefore can be used in regression. Once these Principal

Components are formed, the first Principal Component explains the most data variance followed by the 2nd which is completely uncorrelated with the first and so on (Mevik and Wehrens, 2007).

Choices made when using the method:

There are a number of choices that need to be made when performing PCA. Firstly, data needs to be standardised prior to PCA as principle components are dependent on 'the scaling of the inputs' (Hastie, Tibshirani and Friedman, 2017). When performing PCA, outlining a suitable number of components is an important step and can lead to 'substantial improvement' in the Least Squares score (James et al., 2021). Picking the number of components can be done by using cross-validation and out of bag predictions. Generally a good number of components is highlighted by the fact that after it 'the cross-validation error does not show a significant decrease' (Mevik and Wehrens, 2007). This remains a somewhat subjective practice, but using the above approach can help come to a suitable conclusion.

Advantages and Disadavantages of method:

The main advantage commonly associated with PCR are, the reduction of data dimensions while retaining the data variability and, the creation of uncorrelated predictors (Mevik and Wehrens, 2007). It focuses on the more valuable high variance directions while discarding the rest (Hastie, Tibshirani and Friedman, 2017). It can also speed up data processing in subsequent analysis.

PCR can struggle when the first number of Principle Components fail to capture a satisfactory level of variance and thereby failing to reduce data dimensions or being left with a dataset that loses significant amount of variance that was in initial dataset (James et al., 2021). It can be difficult to select a suitable number of components as there are no 'generally applicable tests available' (Larose and Larose, 2019), this means the number of components selected is somewhat subjective and therefore may not be optimal.

Question 8

A PCR model is initialized and missing values in the test data set are imputed to preserve data structure and size.

```r
#Define PCR model
formula <- aveOralM ~ .
# Impute missing values in test_data
missing_cols <- colnames(test_data)[apply(test_data, 2, anyNA)]
for (col in missing_cols) {
  median_val <- median(test_data[[col]], na.rm = TRUE)
  test_data[[col]][is.na(test_data[[col]])] <- median_val
}
# Fit the PCR model using the training data
set.seed(123)
train_data_subset <- train_data[sample(nrow(train_data), nrow(test_data)), ]
pcr_model <- pcr(formula, data = train_data_subset, scale = TRUE)
```

Once the PCR model has been trained, predictions can be made using the unseen test data. Once this model has been fit, model performance metrics are calculated.

```r
# Predict Oral Temperature for the test set
test_predictions <- predict(pcr_model, newdata = test_data)

# Evaluate the model performance
test_predictions_single <- test_predictions[, 1, 1]
rsquared <- cor(test_data$aveOralM, test_predictions_single)^2
mse <- mean((test_data$aveOralM - test_predictions_single)^2)
rmse <- sqrt(mse)
```

```r
# Print evaluation metrics
print(paste("Mean Squared Error (MSE):", mse))
```

```
[1] "Mean Squared Error (MSE): 0.147639239278487"
```

```r
print(paste("Root Mean Squared Error (RMSE):", rmse))
```

```
[1] "Root Mean Squared Error (RMSE): 0.384238518733465"
```

```r
print(paste("R-squared:", rsquared))
```

```
[1] "R-squared: 0.707018273394008"
```

R^2 of 70% suggests the model explains a significant amount of the variability in the data.

While understanding that lower values are desired for MSE and RMSE scores, without comparing to other models it is difficult to judge whether the model has performed well given these scores in isolation.

Research References

1. Hastie, T., Tibshirani, R. and Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction Second Edition*. [online] Springer.

2. James, G., Witten, D., Hastie, T. and Tibshirani, R. (2021). *INTRODUCTION TO STATISTICAL LEARNING : with applications in r.* S.L.: Springer-Verlag New York.

3. Larose, C.D. and Larose, D.T. (2019). *Data science using Python and R*. Hoboken: Wiley.

4. Mevik, B.-H. and Wehrens, R. (2007). The pls Package: Principal Component and Partial Least Squares Regression in R. *Journal of Statistical Software*, 18(2).