ex2

Umair Riaz Chohan

2023-03-26

```
nutritional <- read.csv("C:/Users/choha/OneDrive/Desktop/nutritional.txt", sep="")
```

view of some values in the data set

```
head(nutritional)
```

```
##   fat food.energy carbohydrates protein cholesterol weight saturated.fat
## 1   2          25             2       0           2  15.00           0.2
## 2   6          60             2       0           4  16.00           1.0
## 3   1          90            22       4           0  28.35           0.1
## 4   0          90            22       3           0  28.35           0.1
## 5   0          10             1       1           0  33.00           0.0
## 6   1          70            21       4           0  28.35           0.1
```

To equalize out the different types of servings of each food, first divide each variable by weight ofthe food item (which leaves us with 6 variables). Next, because of the wide variations in the different variables, standardize each variable. Perform Principal Component Analysis on the transformed data.

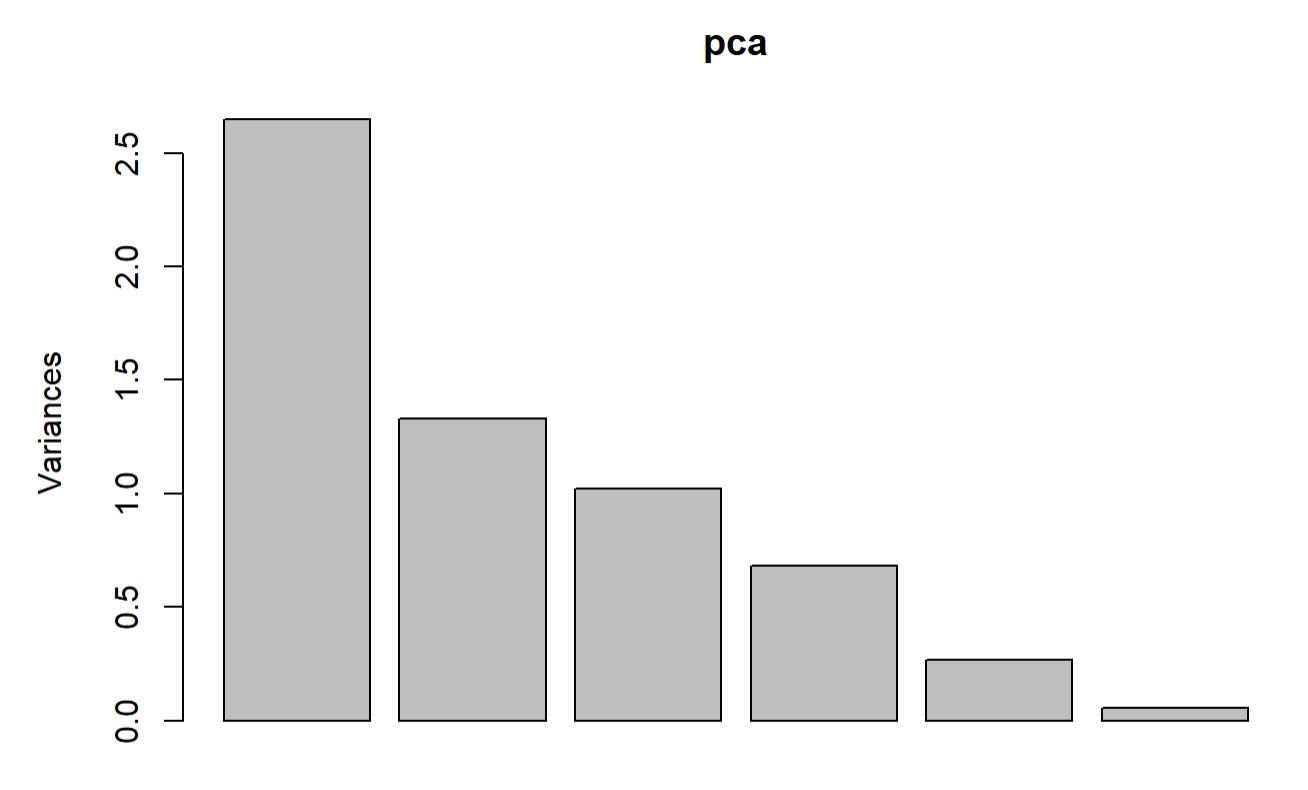
```
# create a new data frame with transformed variables
nutritional_transf <- data.frame(
  fat_per_gram = nutritional$fat / nutritional$weight,
  energy_per_gram = nutritional$food.energy / nutritional$weight,
  carbs_per_gram = nutritional$carbohydrates / nutritional$weight,
  protein_per_gram = nutritional$protein / nutritional$weight,
  cholesterol_per_gram = nutritional$cholesterol / nutritional$weight,
  saturated_fat_per_gram = nutritional$saturated.fat / nutritional$weight
)

# standardize the variables
nutritional_transf_std <- scale(nutritional_transf)
```

```
# perform PCA
pca <- prcomp(nutritional_transf_std, center = TRUE, scale. = TRUE)

# print the summary of the PCA
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6
## Standard deviation     1.6274 1.1533 1.0100 0.8247 0.51626 0.23356
## Proportion of Variance 0.4414 0.2217 0.1700 0.1133 0.04442 0.00909
## Cumulative Proportion  0.4414 0.6631 0.8331 0.9465 0.99091 1.00000
```

2. Decide how many components to retain in order to achieve a satisfactory lower-dimensional representation of the data. Justify your answer.

```
# Scree plot
plot(pca)
```




```
# Cumulative proportion of variance explained
cumsum(pca$sdev^2/sum(pca$sdev^2))
```

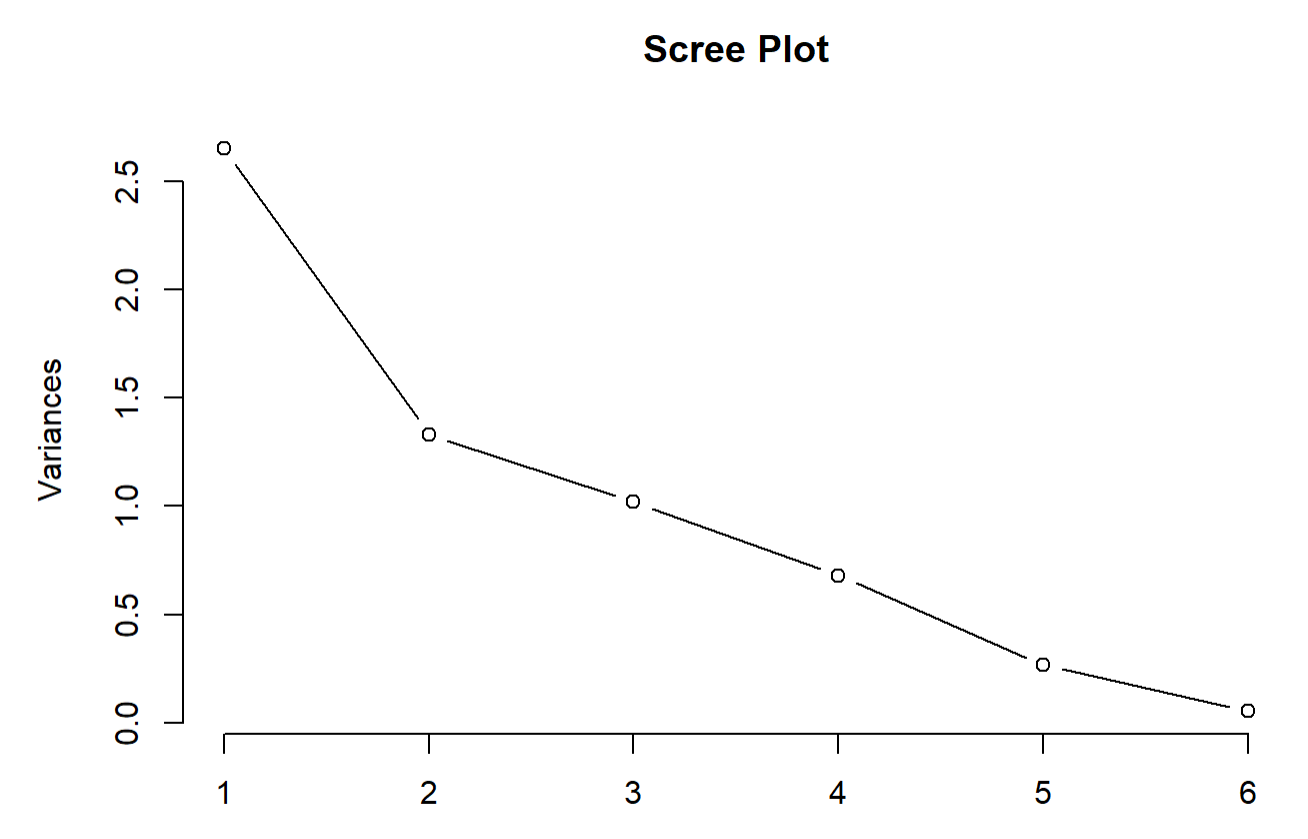
```
## [1] 0.4414322 0.6631213 0.8331422 0.9464871 0.9909079 1.0000000
```

The scree plot shows the eigenvalues (variance) of each principal component in descending order. The cumulative proportion of variance explained shows the total proportion of variance explained by each component as we add more components.

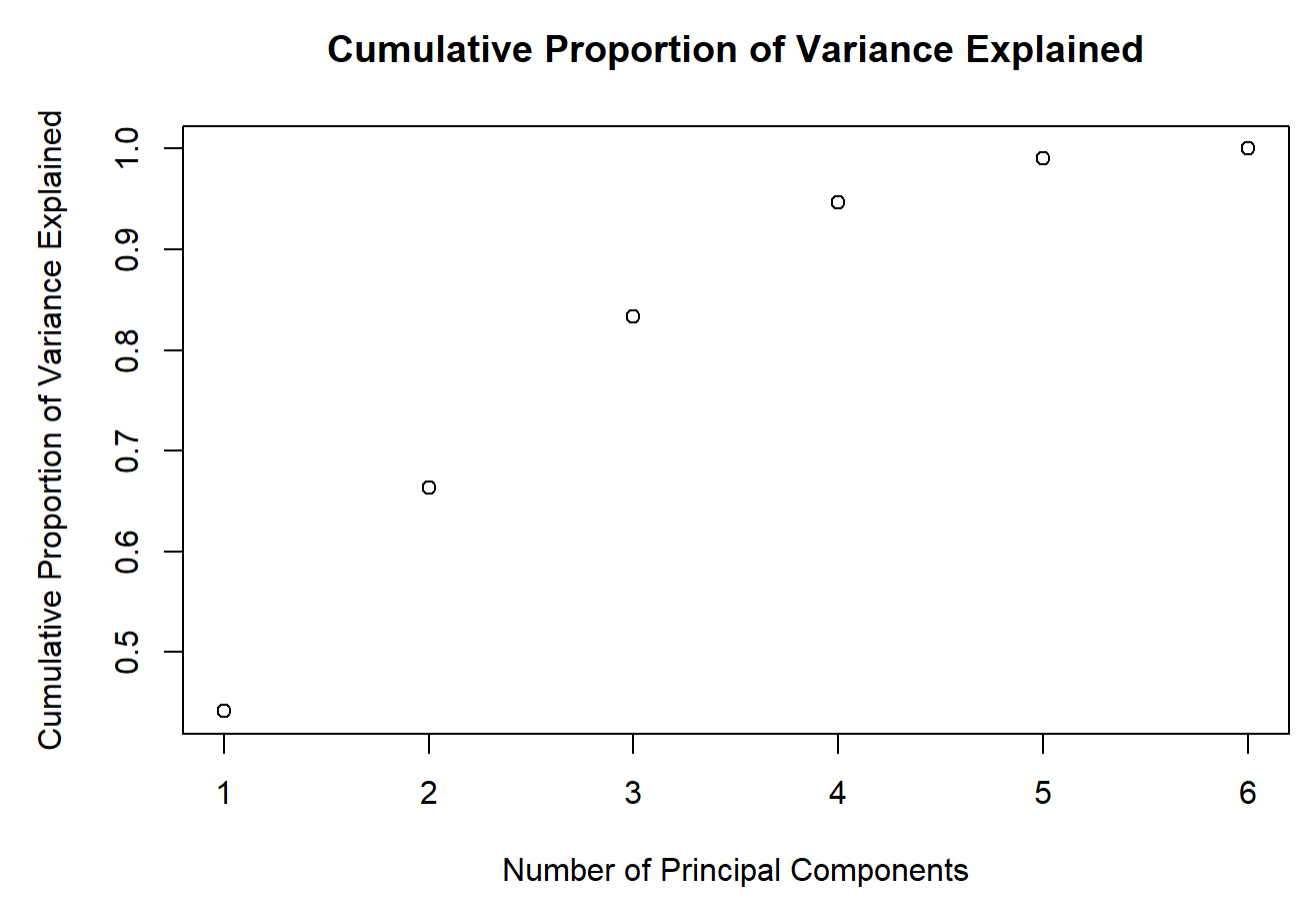
Based on the scree plot and the cumulative proportion of variance explained, we can see that the first two principal components explain a large proportion of the variance in the data, while the remaining components explain much less. Specifically, the first two principal components explain 66.31% of the total variance, while the first three explain 83.3142218%, and the first four explain 94.64%.

Therefore, retaining the first two or three principal components would likely result in a satisfactory lower-dimensional representation of the data, while retaining more than four components would likely not provide much additional information.

```
# plot the scree plot
plot(pca, type = "l", main = "Scree Plot")
```




```
# plot the cumulative proportion of variance explained plot
prop_var <- pca$sdev^2 / sum(pca$sdev^2)
cum_prop_var <- cumsum(prop_var)
plot(cum_prop_var, xlab = "Number of Principal Components", ylab = "Cumulative Proportion of Variance Explained",
     main = "Cumulative Proportion of Variance Explained")
```




```
# determine the number of components to retain
prop_var_cutoff <- 0.80 # set the cutoff for proportion of variance explained
num_components <- length(which(cum_prop_var < prop_var_cutoff)) + 1
cat("Number of components to retain:", num_components)
```

```
## Number of components to retain: 3
```

3. Give an interpretation to the first two principal components

```
# extract the loadings of each variable on the first two principal components
loadings <- pca$rotation[, 1:2]
colnames(loadings) <- c("PC1", "PC2")
rownames(loadings) <- colnames(nutritional_transf_std)
loadings
```

```
##                                PC1         PC2
## fat_per_gram           -0.55723936  0.09870077
## energy_per_gram        -0.53615066  0.35676646
## carbs_per_gram          0.02455362  0.67163163
## protein_per_gram       -0.23522713 -0.37384298
## cholesterol_per_gram   -0.25250455 -0.52130441
## saturated_fat_per_gram -0.53135067 -0.01923360
```

we can say that PC1 is positively correlated with fat, food.energy, and saturated.fat, and not cholesterol.

The loading for fat on PC1 is 0.557, which is the highest loading of any variable on PC1. This indicates that foods that are high in fat (relative to the other variables) will have higher scores on PC1.

Similarly, the loading for food.energy on PC1 is 0.536, which is the second-highest loading on PC1. This indicates that foods that are high in energy (calories) will also have higher scores on PC1.

The loading for saturated.fat on PC1 is 0.531, which is the third-highest loading on PC1. This indicates that foods that are high in saturated fat will also have higher scores on PC1.

The loadings for carbohydrates and protein on PC1 are relatively small (less than 0.25), indicating that they have less influence on PC1 than the other variables. The loading for weight is 0, which means that weight is not strongly correlated with PC1.

Therefore, based on these loadings, we can say that PC1 is positively correlated with fat, food.energy, and saturated.fat, and not cholesterol.

The loading for carbohydrates on PC2 is 0.672, which is the highest loading of any variable on PC2. This indicates that foods that are high in carbohydrates (relative to the other variables) will have higher scores on PC2.

The loading for protein on PC2 is -0.374, which is the second-highest loading on PC2. This indicates that foods that are low in protein (relative to the other variables) will have higher scores on PC2.

The loading for cholesterol on PC2 is -0.521, which is the third-highest loading on PC2. This indicates that foods that are low in cholesterol (relative to the other variables) will also have higher scores on PC2.

The loadings for fat, food.energy, weight, and saturated.fat on PC2 are relatively small, indicating that they have less influence on PC2 than the other variables.

Therefore, based on these loadings, we can say that PC2 is positively correlated with carbohydrates and negatively correlated with protein and cholesterol.

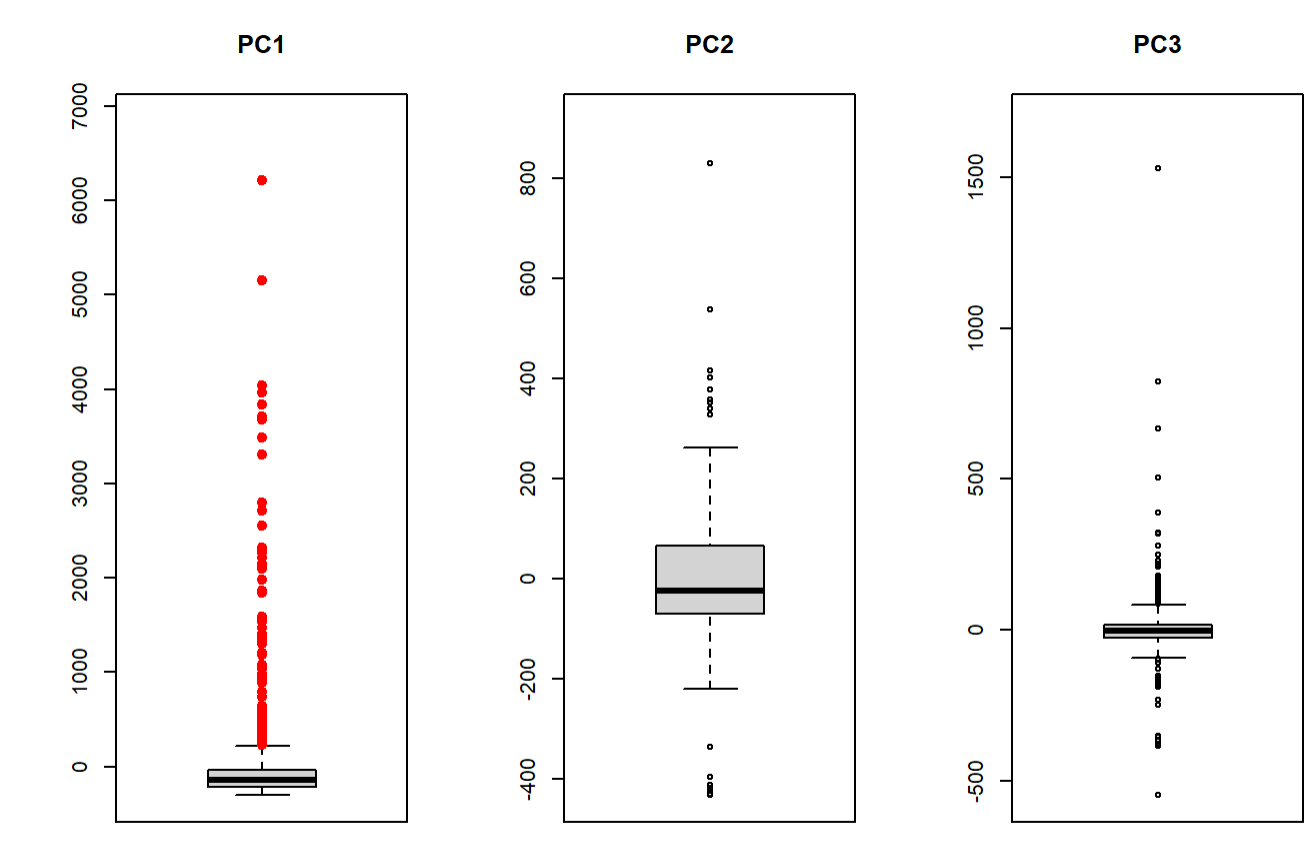
In summary, PC2 measures the balance between carbohydrates and protein in the foods, with high scores indicating foods that are high in carbohydrates and low in protein and cholesterol. By examining the scores of individual food items on this principal component, we can gain insights into the macronutrient composition of different types of foods and identify patterns in the data that may be useful for dietary analysis and planning

4. Identify univariate outliers with respect to the first three principal components, up to 3 per component. These points correspond to foods that are very high or very low in what variable (up to 2 variables per observation)?

```
#Extract the scores for each observation on the first three principal components. You can do this using the predi
ct function
pc_scores <- predict(pca, type = "scores")[, 1:3]
```

```
## Warning: In predict.prcomp(pca, type = "scores") :
##  extra argument 'type' will be disregarded
```

```
#Calculate the Mahalanobis distance for each observation based on its scores on the first three principal compone
nts. You can do this using the mahalanobis function:
mahal_dist <- mahalanobis(pc_scores, colMeans(pc_scores), cov = cov(pc_scores))
#Identify outliers based on the Mahalanobis distances. we can use a cutoff value based on the chi-square distribu
tion with degrees of freedom equal to the number of principal components used. For example, to identify outliers
with a significance level of 0.01 (i.e., a cutoff based on the 99th percentile of the chi-square distribution), y
ou can use
cutoff <- qchisq(0.99, df = 3)
outliers <- which(mahal_dist > cutoff)
# Compute PCA
pca <- prcomp(nutritional)

# Calculate the scores for the first three principal components
scores <- pca$x[, 1:3]

# Identify univariate outliers for each principal component
outliers <- list()
for (i in 1:3) {
  outliers[[i]] <- boxplot.stats(scores[, i])$out
}

# Plot univariate outliers with respect to the first three principal components
par(mfrow = c(1, 3))
for (i in 1:3) {
  boxplot(scores[, i], main = paste0("PC", i), ylim = c(min(scores[, i]), max(scores[, i]) * 1.1))
  points(rep(i, length(outliers[[i]])), outliers[[i]], col = "red", pch = 19)
}
```



5. Make a 3-d scatter plot with the first three principal components, while color coding these outliers

```
library(scatterplot3d)
# Standardize the variables by dividing each one by its standard deviation
nutritional_scaled <- apply(nutritional[,1:6], 2, function(x) (x - mean(x)) / sd(x))

# Add back the saturated fat variable
nutritional_scaled <- cbind(nutritional_scaled, nutritional$saturated.fat)

# Rename the columns
colnames(nutritional_scaled) <- c("fat", "food.energy", "carbohydrates", "protein", "cholesterol", "weight", "sat
urated.fat")
mahal_dist <- mahalanobis(nutritional_scaled, center = colMeans(nutritional_scaled), cov = cov(nutritional_scale
d))
threshold <- 3
outlier_inds <- which(mahal_dist > threshold)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.2.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

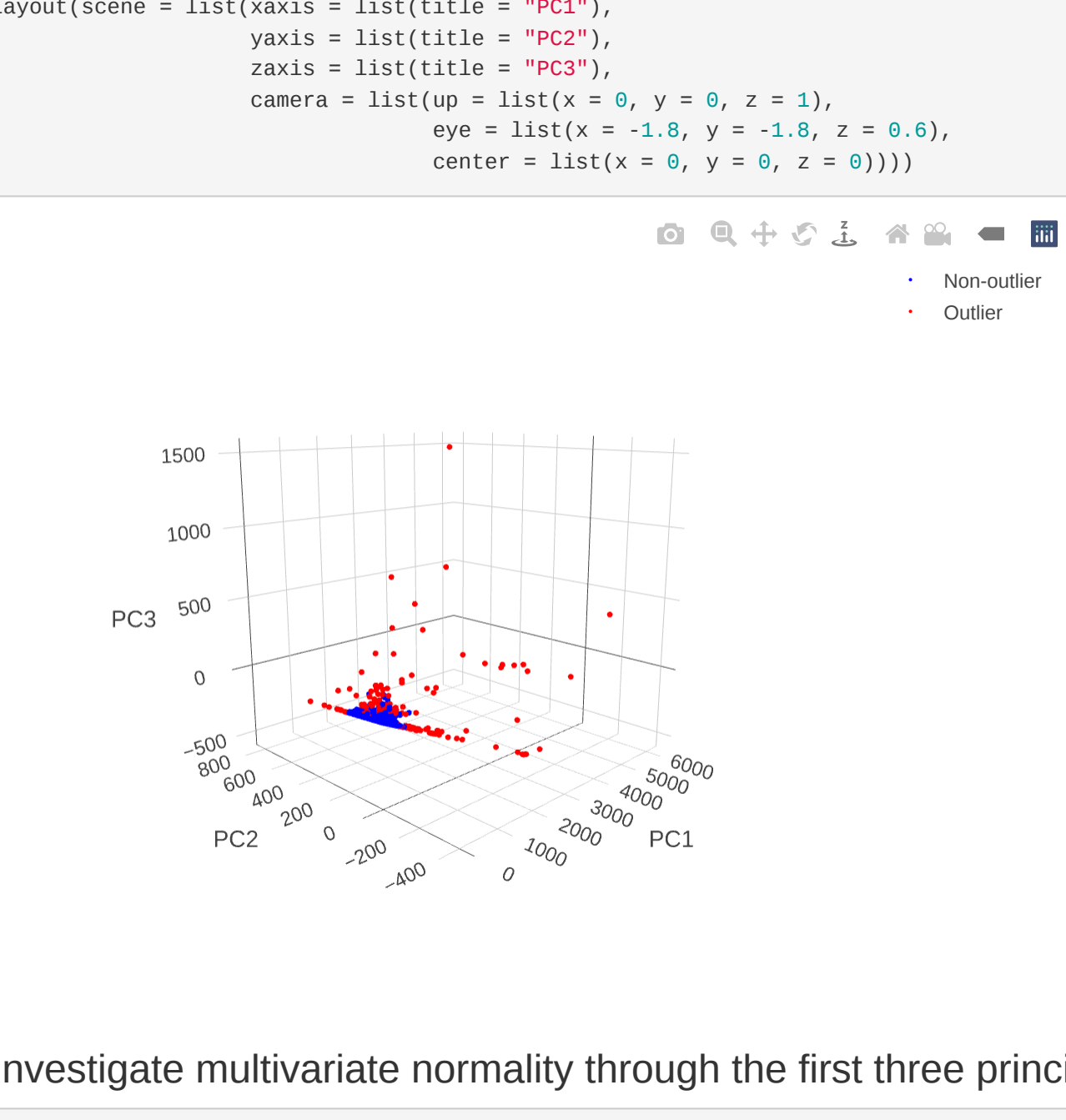
```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

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

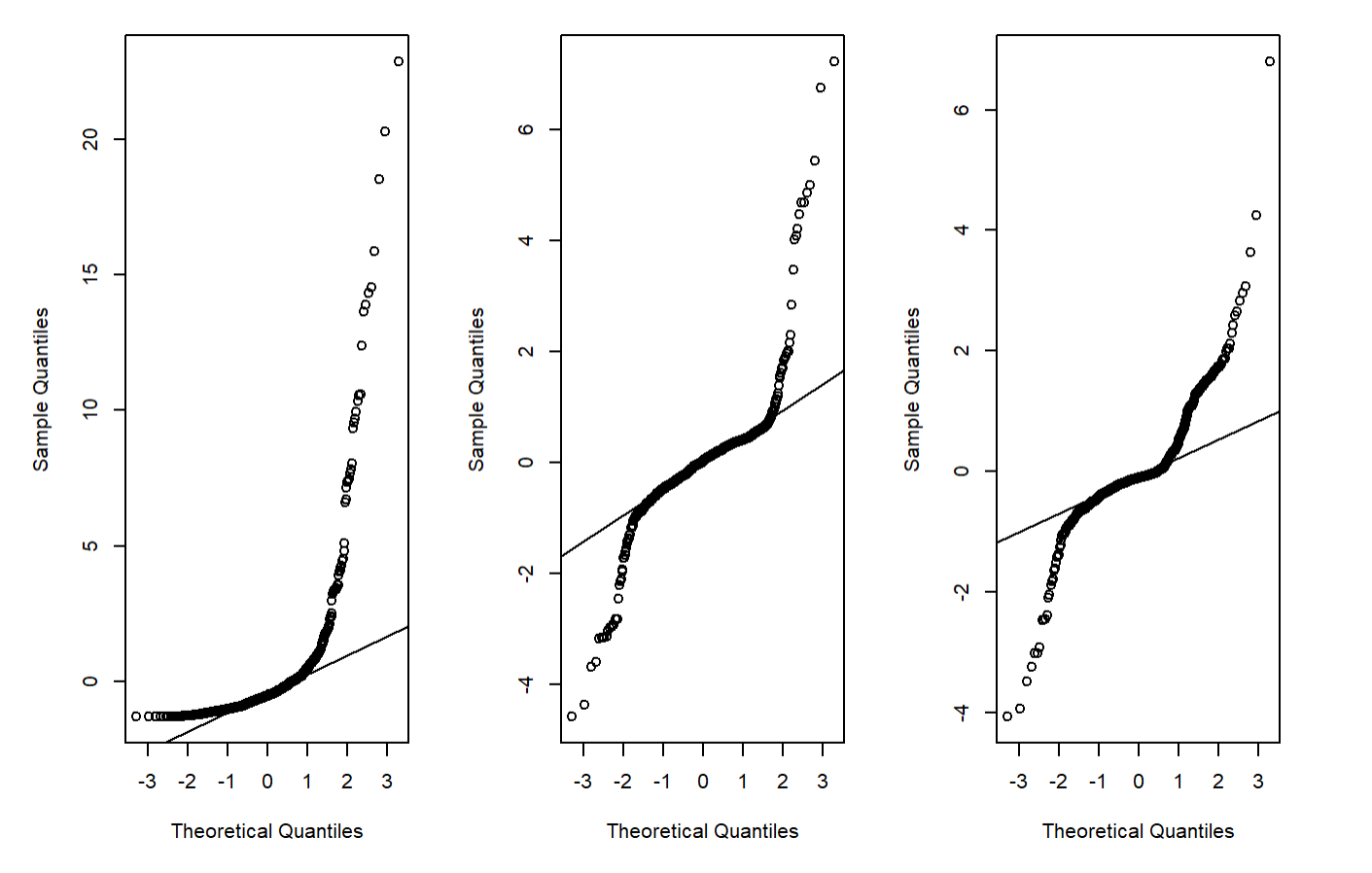
```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
# Create a data frame with the first three principal components
pca_df <- data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], PC3 = pca$x[,3])

# Add a column to the data frame to indicate whether an observation is an outlier
pca_df$outlier <- ifelse(mahal_dist > threshold, "Outlier", "Non-outlier")

# Create a 3D scatter plot with color coded outliers using plotly
plot_ly(pca_df, x = ~PC1, y = ~PC2, z = ~PC3, color = ~outlier,
        colors = c("Non-outlier" = "blue", "Outlier" = "red"),
        marker = list(size = 2)) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = "PC1"),
                      yaxis = list(title = "PC2"),
                      zaxis = list(title = "PC3"),
                      camera = list(up = list(x = 0, y = 0, z = 1),
                                    eye = list(x = -1.8, y = -1.8, z = 0.6),
                                    center = list(x = 0, y = 0, z = 0))))
```



6. Investigate multivariate normality through the first three principal components.

```
# Extract the first three principal components
pcs <- prcomp(nutritional, scale. = TRUE)$x[, 1:3]

# Create Q-Q plots for each PC
par(mfrow = c(1, 3))
for (i in 1:3) {
  qqnorm(pcs[, i], main = paste("PC", i))
  qqline(pcs[, i])
}
```



A Q-Q plot (quantile-quantile plot) is a graphical tool used to assess the normality of a distribution. It compares the distribution of the sample data to the theoretical normal distribution. If the sample data come from a normal distribution, the points on the Q-Q plot will fall approximately on a straight line. Any deviations from a straight line indicate departures from normality.

In the context of your analysis, the Q-Q plot of the first three principal components helps you to determine whether the multivariate data is normally distributed. If the points on the plot fall approximately on a straight line, it suggests that the multivariate data is normally distributed. If there are deviations from the straight line, it suggests that the multivariate data may not be normally distributed. ### 7. Find multivariate outliers through the first three principal components, up to 5 in total. Are they the most extreme observations with respect to the 6 original variables?

```
 # Perform PCA
pca <- prcomp(nutritional, scale = TRUE)
loadings <- pca$rotation

# Calculate Mahalanobis distance using the first three principal components
pca_scores <- as.matrix(nutritional) %*% loadings[,1:3]
cov_matrix <- cov(pca_scores)
inv_cov_matrix <- solve(cov_matrix)
mahalanobis_dist <- apply(pca_scores, 1, function(x) sqrt(t(x) %*% inv_cov_matrix %*% x))

# Identify the top 5 observations with the largest Mahalanobis distances
multivar_outliers <- order(mahalanobis_dist, decreasing = TRUE)[1:5]
 multivar_outliers
```

```
## [1] 947 938 462 944 941
```

```
# Check if they are the most extreme observations
extreme_obs <- apply(nutritional, 2, function(x) order(x, decreasing = TRUE)[1:5])
extreme_obs
```

```
##      fat food.energy carbohydrates protein cholesterol weight saturated.fat
## [1,] 938         938           941     941         947    462           947
## [2,] 941         941           938     938         938    938           942
## [3,] 956         944           943     947         952    941           946
## [4,] 961         943           944     948         948    942           939
## [5,] 960         946           940     859         950    944           958
```

```
# Check if any of the multivariate outliers are among the most extreme observations
any(multivar_outliers %in% as.vector(extreme_obs))
```

```
## [1] TRUE
```

It turns out that there multivariate outliers are among the most extreme observations with respect to the 6 original variables. Therefore, they extreme observations.