Final Project - Data Analysis on Irish Hospital Discharge Data

Sarosh Farhan (24210969)

Table of contents

Part 1: Analysis

1. Loading the data-set and pre processing

I have chosen the data-set from https://data.cso.ie/table/DHA38 which gives Hospital Discharges of In-Patients, Daycases and Length of Stay (All Diagnoses). I chose this data-set to see how Irish healthcare system works and how patient care is done in this country.
I have data for the years 2017 to 2021.

```
#loading necessary libraries
library(rio)
library(dplyr)
library(lubridate)
library(scales)
library(tidyr)
library(ggplot2)


data = read.csv("hospitalDischargeDataYears.csv")
head(data)
```

```
  Year        Sex     Age.Group  County.and.HSE.Region In.Patients..Number.
1 2017 Both sexes  0 - 14 years Dublin City and County                17593
2 2017 Both sexes  0 - 14 years                Kildare                 3537
3 2017 Both sexes  0 - 14 years               Limerick                 3509
4 2017 Both sexes  0 - 14 years                 Galway                 4518
5 2017 Both sexes 15 - 24 years Dublin City and County                10326
6 2017 Both sexes 15 - 24 years                Kildare                 2189
  Daycases..Number. In.Patients.and.Daycases..Number.
1             12683                             30276
2              2449                              5986
3              2001                              5510
4              2924                              7442
5             11599                             21925
6              1697                              3886
  Average.Length.of.Stay..Rate. In.Patient.Bed.Days..Number.
1                          4.00                        70373
2                          3.81                        13477
3                          3.65                        12809
4                          3.74                        16893
5                          3.32                        34247
6                          3.01                         6593
  Median.Length.of.Stay..Number. Per.1.000.Population..Rate.
1                              2                      115.74
2                              2                      109.58
3                              2                      139.10
4                              1                      137.91
5                              2                      132.31
6                              1                      136.04
```

Renaming the column names for ease of manipulation. I renamed the column by using *colnames()* method and gave easy and descriptive column names for easy understanding and manipulation.

```
# Change column names
colnames(data) <- c("year", "sex", "age_group", "county",
                    "in_patients", "daycases", "in_patient_daycases",
                    "avg_length_stay", "in_patient_bed_days",
                    "median_length_of_stay", "per_1000_population")
```

Showing the data-frame column's datatypes using *str()*

```
str(data, width = 80, strict.width = "wrap")
```

```
'data.frame':   600 obs. of  11 variables:
$ year : int 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 ...
$ sex : chr "Both sexes" "Both sexes" "Both sexes" "Both sexes" ...
$ age_group : chr "0 - 14 years" "0 - 14 years" "0 - 14 years" "0 - 14 years"
   ...
$ county : chr "Dublin City and County" "Kildare" "Limerick" "Galway" ...
$ in_patients : int 17593 3537 3509 4518 10326 2189 2174 2807 24545 4654 ...
$ daycases : int 12683 2449 2001 2924 11599 1697 986 1855 28095 3608 ...
$ in_patient_daycases : int 30276 5986 5510 7442 21925 3886 3160 4662 52640
   8262 ...
$ avg_length_stay : num 4 3.81 3.65 3.74 3.32 3.01 2.92 2.47 3.14 2.85 ...
$ in_patient_bed_days : int 70373 13477 12809 16893 34247 6593 6348 6935 77174
   13247 ...
$ median_length_of_stay: num 2 2 2 1 2 1 1 1 2 2 ...
$ per_1000_population : num 116 110 139 138 132 ...
```

Checking for null values using *is.na()* and removing null values if any. We see that there are no null values in the data.

```
# Count missing values
sum(is.na(data))
```

```
[1] 0
```

Showing summary as a table form for the integer data only. I first filtered out the integer column data using *select()* method and selectinc only column with *is.integer* type and then I displayed the summary using the *knitr::kable()* method.

```
# Select only integer columns
integer_columns <- data %>% select(where(is.integer))

# Display the summary in a neat table
knitr::kable(summary(integer_columns), format = 'html',
             caption = "Summary of Integer Columns in Hospital Data")
```

Table 1: Summary of Integer Columns in Hospital Data

year	in_patients	daycases	in_patient_daycases	in_patient_bed_days
Min. :2017	Min. : 454	Min. : 369	Min. : 878	Min. : 1283
1st Qu.:2018	1st Qu.: 1547	1st Qu.: 1845	1st Qu.: 3966	1st Qu.: 7932
Median :2019	Median : 2934	Median : 4732	Median : 7350	Median : 14413
Mean :2019	Mean : 7826	Mean : 14832	Mean : 22657	Mean : 51585
3rd Qu.:2020	3rd Qu.: 7056	3rd Qu.: 13178	3rd Qu.: 22007	3rd Qu.: 46966
Max. :2021	Max. :149117	Max. :324026	Max. :471236	Max. :1120126

I also changed datatype of column *sex* and *age_group* as factor as these will be easier to manipulate if in factor data type.

```
# Ensuring that categorical variables are
# treated as factors
data$sex <- as.factor(data$sex)
data$age_group <- as.factor(data$age_group)

str(data, width = 80, strict.width = "wrap")
```

```
'data.frame':   600 obs. of  11 variables:
```

```
$ year : int 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 ...
$ sex : Factor w/ 3 levels "Both sexes","Female",..: 1 1 1 1 1 1 1 1 1 1 ...
$ age_group : Factor w/ 10 levels "0 - 14 years",..: 1 1 1 1 2 2 2 2 3 3 ...
$ county : chr "Dublin City and County" "Kildare" "Limerick" "Galway" ...
$ in_patients : int 17593 3537 3509 4518 10326 2189 2174 2807 24545 4654 ...
$ daycases : int 12683 2449 2001 2924 11599 1697 986 1855 28095 3608 ...
$ in_patient_daycases : int 30276 5986 5510 7442 21925 3886 3160 4662 52640
   8262 ...
$ avg_length_stay : num 4 3.81 3.65 3.74 3.32 3.01 2.92 2.47 3.14 2.85 ...
$ in_patient_bed_days : int 70373 13477 12809 16893 34247 6593 6348 6935 77174
   13247 ...
$ median_length_of_stay: num 2 2 2 1 2 1 1 1 2 2 ...
$ per_1000_population : num 116 110 139 138 132 ...
```

2. Exploratory Data Analysis

Now that we have done some pre-processing of our data and cleaned it, we can now do some analysis on our data-set.

a. Tabular statistics

First I want to see summary statistics if inpatient and daycare patient numbers so that we can know how many people are under daycare and how many people gets admitted on an average and also how long they stay before being discharged across the four counties and the 5 year data that we have.

```r
# Summary statistics for inpatient and daycase numbers
summaryStats <- data %>%
  group_by(county) |>
  summarise(
    `Total Inpatients` = sum(in_patients),
    `Total Daycases` = sum(daycases),
    `Average Length of Stay` = mean(avg_length_stay, na.rm = TRUE),
    `Median Length of Stay` = median(median_length_of_stay, na.rm = TRUE)
  )

knitr::kable(summaryStats, format = 'html',
      caption = "Summary statistics for inpatient and daycase numbers")
```

Table 2: Summary statistics for inpatient and daycase numbers

county	Total Inpatients	Total Daycases	Average Length of Stay	Median Length of Stay
Dublin City and County	2885056	6135888	7.910667	3
Galway	701400	1183872	5.840733	2
Kildare	527300	854560	6.247000	2
Limerick	581560	724688	4.770267	2

From the table above we see that Dublin City and County sees the most day cases and has more people admitted in hospital than other county, this is rightly so as Dublin and county is far more greater than all the other counties in Ireland and the data supports it.

b. Visualizations

The best analysis can be done using visualizations and seeing trends in the data to gain insights on how the healthcare infrastructure works in Ireland.

```r
# Total Inpatients by Age Group
ggplot(data = data, aes(x = age_group,
                        y = in_patients,
                        fill = sex)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Total Inpatients by Age Group and Sex",
```

```
       x = "Age Group", y = "Number of Inpatients") +
  theme_minimal()+
  #to center align title
  theme(plot.title = element_text(hjust = 0.5))
```

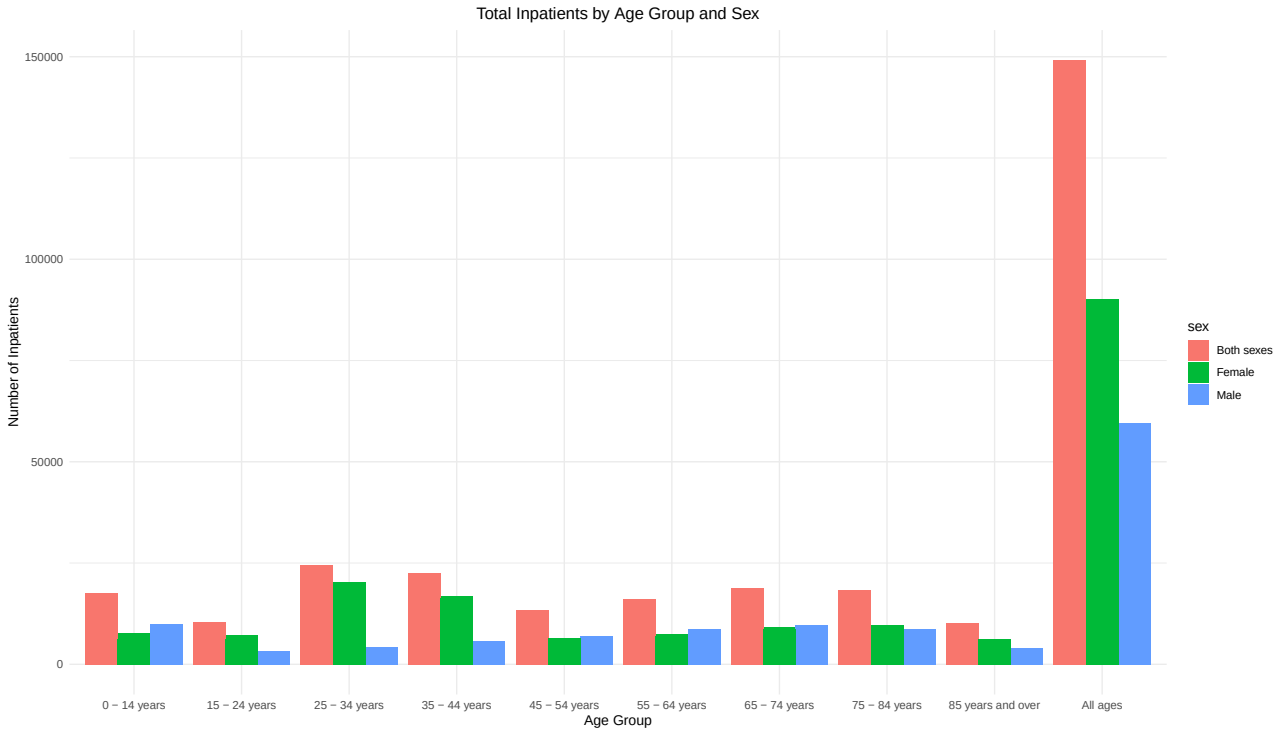

Figure 1: Total Inpatients by Age Group and Sex

From the above graph comparing age-group and sex, we see that in most cases i.r age group, the number of female inpatient is higher than male inpatients.

```
# Average length of stay per county
avg_length_stay <- data %>%
  group_by(county) %>%
  summarise(avg_length_stay = mean(avg_length_stay))

ggplot(avg_length_stay,
       aes(x = reorder(county,
                       -avg_length_stay),
           y = avg_length_stay, fill = county)) +
  geom_bar(stat = "identity") +
  labs(title = "Average Length of Stay by County",
       x = "County", y = "Average Length of Stay (Days)") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5))+
  guides(fill = "none")
```

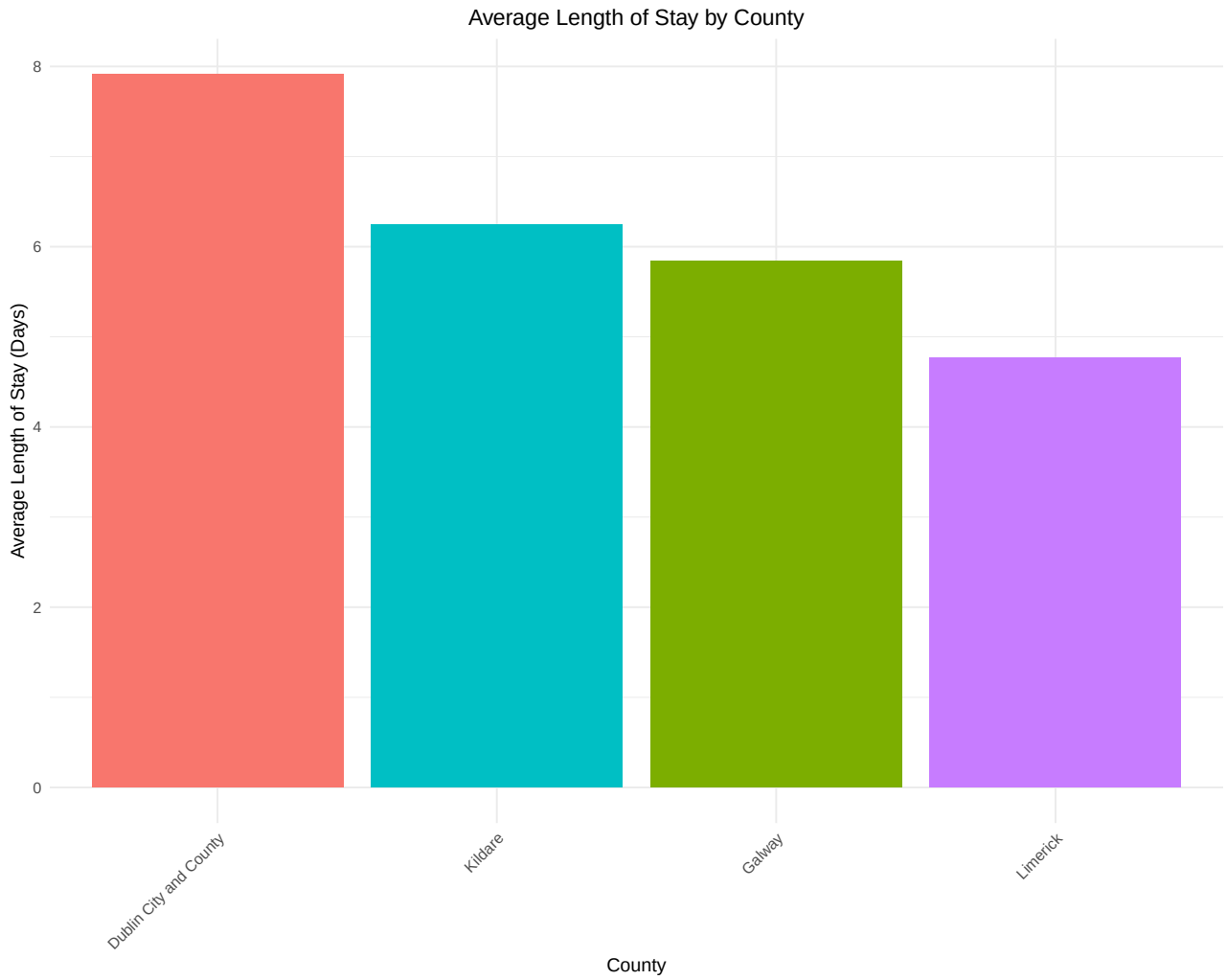

Figure 2: Average Length of Stay by County

The graph illustrates a clear variation in the average length of stay across the four counties: Dublin City and County, Kildare, Galway, and Limerick.

Dublin City and County: This region has the highest average length of stay, indicating that patients in this area tend to spend more time in hospitals compared to the other counties. This could be due to various factors, such as:

1. *Complex cases:* A higher concentration of specialized hospitals may attract more complex cases requiring longer hospital stays.
2. *Population demographics:* A larger elderly population or a higher prevalence of chronic diseases might contribute to longer hospitalizations.
3. *Healthcare system factors:* Differences in healthcare delivery models, resource allocation, or wait times could impact the length of stay.

Kildare, Galway, and Limerick: These counties have relatively shorter average lengths of stay compared to Dublin. This could be attributed to factors such as:

1. *Smaller hospitals:* Smaller hospitals may have fewer resources and capabilities to handle complex cases, leading to shorter stays.
2. *Population demographics***:** A younger population with fewer chronic illnesses might require less hospital care.
3. *Healthcare system factors:* Differences in healthcare policies and practices could influence the length of stay.

```
# Aggregate data by year for key metrics
yearly_trends <- data %>%
  group_by(year) %>%
  summarize(
    in_patients_total = sum(in_patients, na.rm = TRUE),
```

```
    daycases_total = sum(daycases, na.rm = TRUE),
    total_cases = sum(in_patient_daycases, na.rm = TRUE)
  )

# Plot the trends over years
ggplot(yearly_trends, aes(x = year)) +
  geom_line(aes(y = in_patients_total, color = "In-Patients", group=1),
            linewidth = 1) +
  geom_line(aes(y = daycases_total, color = "Daycases", group=1),
            linewidth = 1) +
  geom_line(aes(y = total_cases, color = "Total Cases", group=1),
            linewidth = 1) +
  labs(
    title = "Trends in Hospital Cases Over Years",
    x = "Year",
    y = "Number of Cases",
    color = "Case Type"
  ) +
  theme_minimal()+
  theme(plot.title = element_text(hjust = 0.5))
```

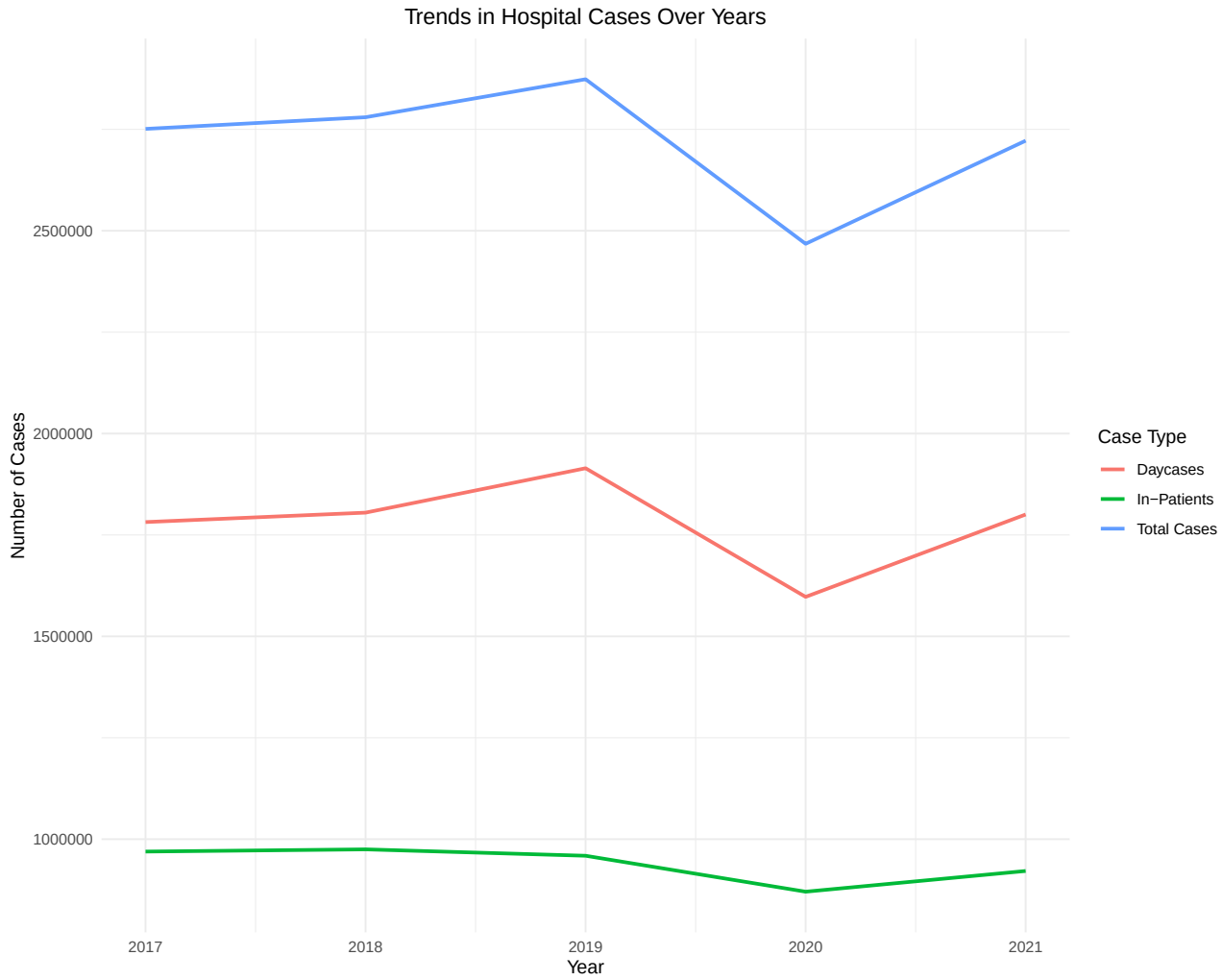

Figure 3: Trends in Hospital Cases Over Years

The graph shows a general increasing trend in the total number of hospital cases from 2017 to 2021. However, there's a notable increase in 2019 in number of daycases, which can be attributed to the COVID-19 pandemic.

We also see the trend picks up after 2020 as second wave hits and number if day-cases increased but this time lesser that

what the increase was in 2019.

```
# Add a COVID period indicator
data <- data %>%
  mutate(Period = ifelse(year < 2019, "Pre-COVID", "COVID"))

# Summarize key metrics by period
period_summary <- data %>%
  group_by(Period) %>%
  summarize(
    total_in_patients = sum(in_patients, na.rm = TRUE),
    total_daycases = sum(daycases, na.rm = TRUE),
    total_cases = sum(in_patient_daycases, na.rm = TRUE),
    avg_length_of_stay = mean(avg_length_stay, na.rm = TRUE)
  )

# Bar plot for cases by period
ggplot(period_summary, aes(x = Period,
                           y = total_cases,
                           fill = Period)) +
  geom_bar(stat = "identity") +
  labs(
    title = "Hospital Cases: Pre-COVID vs COVID Period",
    x = "Period",
    y = "Total Cases"
  ) +
  scale_y_continuous(labels = scales::label_number(big.mark = ","))+
  guides(fill = "none")+
  theme_minimal()+
  theme(plot.title = element_text(hjust = 0.5))
```

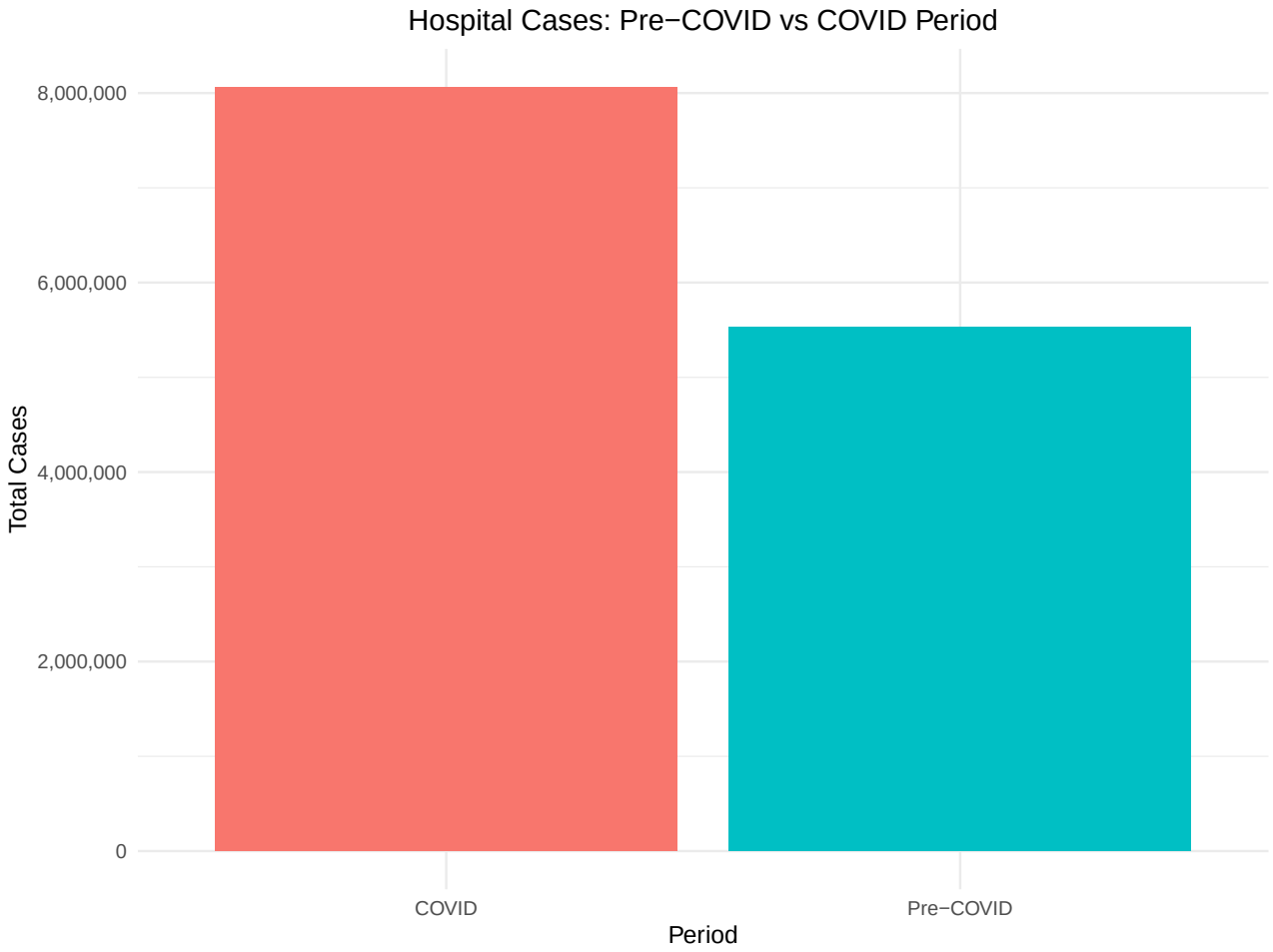

Figure 4: Hospital Cases: Pre-COVID vs COVID Period

The bar chart clearly shows a significant increase in the total number of hospital cases during the COVID period compared to the pre-COVID period, as shown in the above trend.

3. Conclusions

1. We analysed the patient discharge data and saw that women in the four counties selected are more in numbers than men, this might be due to low immunity or women like to frequently go for health checkups than their male counterparts.
2. Dublin and County had the highest average length of stay by patients, this was because Dublin being the capital and having more multi-speciality hospitals can have longer stay for patient for proper nursing and condition improving than other smaller counties.
3. COVID-19 had an impact on the number of daycases and the total cases per year, we saw a huge spike in 2019 and then again in 2021 as COVID came in 2 waves and the data shows the increase suggesting the need for healthcare rose sharply in the pandemic.
4. We also saw the comparison of pre-covid daycases and in covid daycases and we saw a huge increment in the number of people coming for checkups.
5. Overall the healthcare system worked as normal in other years and and COVID-19 really tested the limits of the healthcare systems.

Part 2: R Package

1. Loading the dataset

I will be using a new data-set downloaded from https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

I chose this data because I wanted to learn more about caret library (https://cran.r-project.org/web/packages/caret/index.html) and the classifications algorithms it has to offer.

Since the beginning of the project you must be seeing a pattern of using data-set related to healthcare field, this is a personal choice as I want to do something related to healthcare field.

```
cancer_data <- read.csv("wdbc.data")

# Add column names as the dataset didn't have
# column names, I am just giving dummy column nmaes
num <- 1:32
features <- paste0("feature_", num)
colnames(cancer_data) <- c(features)
colnames(cancer_data)[2] <- "target"
```

After loading the data I have done some pre-processing to add column names as the .data file didn't have column names, after carefully reading the notes on data on the UCI website I was able to separate feature column and target column and hence added header accordingly.

2. Loading the libraries

I am using Caret library for analyzing different classification techniques.

```
# Run the below commands to install the libraries
# Uncomment the code below

#install.packages("caret")
# For SVM (caret requires this)
#install.packages("e1071")

# Load library
library(caret)

# Convert the target variable to a factor
cancer_data$target <- as.factor(cancer_data$target)

# Split the data into training and testing sets
# For reproducibility
set.seed(42)
train_index <- createDataPartition(cancer_data$target,
                                   p = 0.7, list = FALSE)
train_data <- cancer_data[train_index, ]
test_data <- cancer_data[-train_index, ]
```

After loading the libraries I have converted the target column(on the column which we have to do the classification) to factor. I have also added a seed so that I can control the randomness and results are reproducible.

To create the partition for training and test dataset, I have used *createDataPrtition()*method from the Caret librarywith the parameter `p` specifies the proportion of the data to be used for training. In this case, `0.7` means that 70% of the data will be allocated for training the model. The remaining 30% will typically be used for testing or validation.

`list=FALSE` this parameter determines the format of the output. When set to FALSE, the function returns a vector of row indices that can be used to subset the original dataset directly. If it were set to TRUE, it would return a list where each element corresponds to a different partition.

3. Training Random Forest model

Before training the model I used `trainControl()` method and this function from Caret library sets up the parameters for how the model training should be conducted.

`method = "cv"` this specifies the resampling method to be used for model evaluation. In this case, "cv" stands for cross-validation. Cross-validation is a technique used to assess how the results of a statistical analysis will generalize to an independent dataset. It involves partitioning the data into subsets, training the model on some subsets, and validating it on others.

`number = 10` this indicates that 10-fold cross-validation should be used. In 10-fold cross-validation, the dataset is randomly divided into 10 equal-sized folds (subsets). The model is trained on 9 of these folds and validated on the remaining fold. This process is repeated 10 times, with each fold being used as the validation set once. The results are then averaged to provide a more reliable estimate of model performance.

```
# Set up training control
# 10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)

# Train the Random Forest model
rf_model <- train(target ~ ., data = train_data,
                  method = "rf",
                  trControl = train_control)
```

The **`train()`** function from Caret library simplifies the process of fitting a model to a dataset while providing options for resampling and tuning.

`target ~ .` this formula specifies the model to be trained. Here, target is the dependent variable (the outcome you want to predict), and . indicates that all other columns in the train_data dataset should be used as independent variables (features) for predicting the target.

`data = train_data` - this specifies the dataset to be used for training the model. In this case, **`train_data`** is a data frame containing the training data that includes both the features and the target variable.

`method = "rf"` - this specifies the type of model to be trained. Here, **`"rf"`** indicates that a Random Forest model will be used. Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of their predictions for classification tasks or mean prediction for regression tasks.

`trControl = train_control` - this specifies the control parameters for training the model, which were defined earlier using the trainControl() function. The train_control object typically includes settings for resampling (e.g., cross-validation) and other training parameters.

4. Predicting and creating Confusion matrix with Random Forest model

I have used `predict()` from stats library, this unction is used to generate predictions from a fitted model. Here, it uses the trained Random Forest model stored in `rf_model`.

I have also visualized the results using the confusion matrix for the classification predictions of the random forest that we have seen above, for this I have used `confisionMatrix()` method, this function is part of the Caret package and is used to compute and display a confusion matrix, which provides insights into the performance of a classification model.

The method is using two argument:

`predictions`: The predicted class labels generated by the model from the previous step.

`test_data$target`: The actual class labels from the test dataset. This is used to compare against the predicted values to evaluate how well the model performed.

```r
# Predict on the test set
predictions <- predict(rf_model, newdata = test_data)

# Confusion matrix
confusion_matrix <- confusionMatrix(predictions, test_data$target)

confusion_matrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction   B   M
         B 104   5
         M   3  58

               Accuracy : 0.9529
                 95% CI : (0.9094, 0.9795)
    No Information Rate : 0.6294
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8985

 Mcnemar's Test P-Value : 0.7237

            Sensitivity : 0.9720
            Specificity : 0.9206
         Pos Pred Value : 0.9541
         Neg Pred Value : 0.9508
             Prevalence : 0.6294
         Detection Rate : 0.6118
   Detection Prevalence : 0.6412
      Balanced Accuracy : 0.9463

       'Positive' Class : B
```

5. Visualizing the results in Caret library

Random Forest models provide a measure of variable importance, which indicates how much each predictor variable contributes to the model's predictions. I just visualized this using the **`varImp()`** function from **`caret`**

```r
importance <- varImp(rf_model, scale = FALSE)

# Plotting the variable importance
plot(importance, main="Feature Importance in Predictions",
     ylab="Features")
```

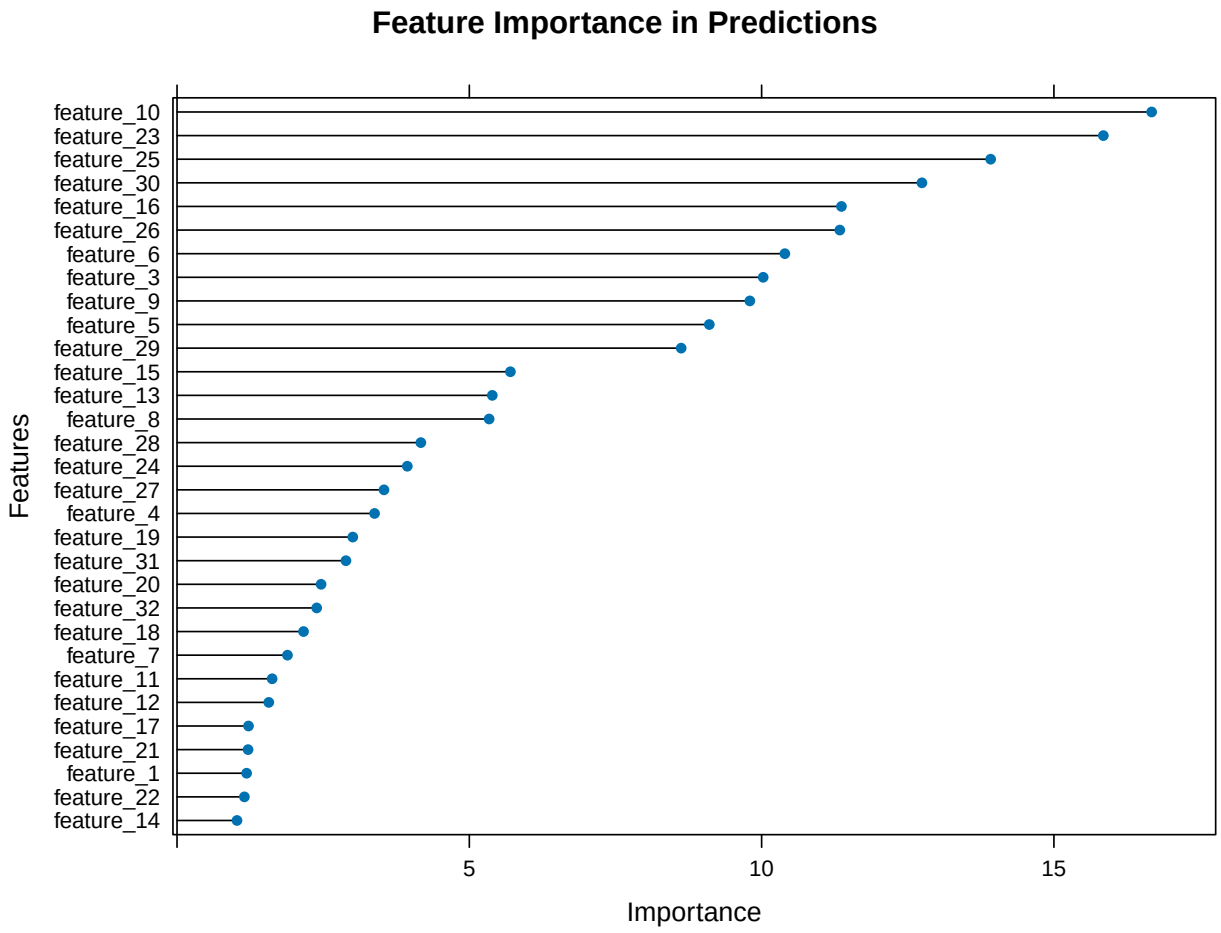

Figure 5: Feature Importance plot

We see that feature_23 is the most important feature in classifying a Benign or Malignant tumor.

An indirect visualization can be done on confusion matrix that we saw above using ggplot, it is kind of a heat map but with confusion matrix. I just converted the `confusion_matrix` into a dataframe by calling `confusion_matrix$table`, the table component of the confusion matrix contains counts of true positives, false positives, true negatives, and false negatives.

```
# Convert confusion matrix to a data frame for plotting
conf_matrix_df <- as.data.frame(confusion_matrix$table)

# Plotting using ggplot2
ggplot(conf_matrix_df, aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Freq), color = "white") +
  scale_fill_gradient(low = "#F7F7FF", high = "#FE5F55") +
  geom_text(aes(label = Freq), vjust = 1) +
  labs(title = "Confusion Matrix", x = "Actual", y = "Predicted")+
  theme_light()+
  theme(plot.title = element_text(hjust = 0.5))
```

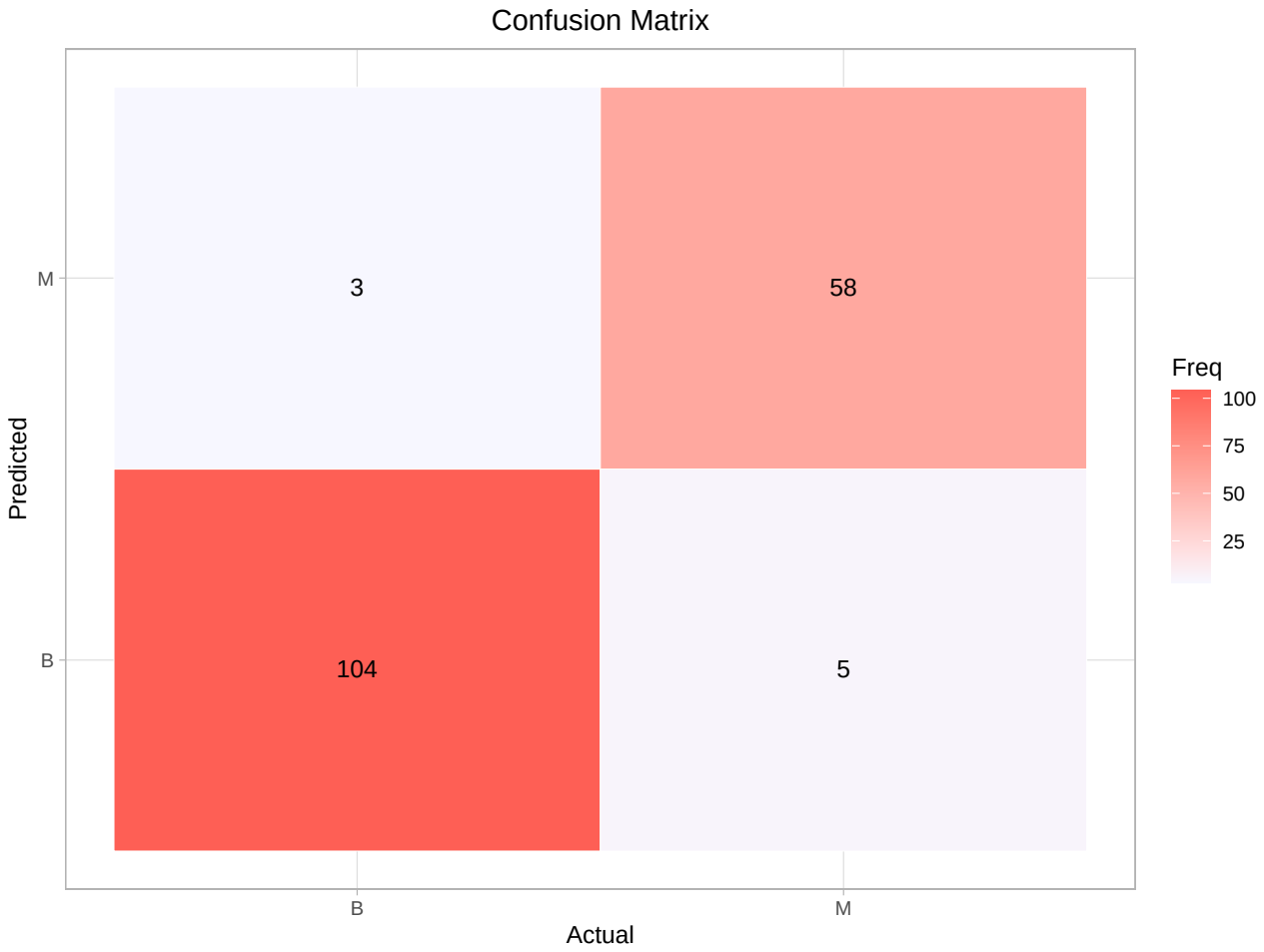

Figure 6: Confusion matrix visualization

Part 3 Functions/Programming

I have chosen to do K Means Clustering classification as the statistical analysis function for this part. I'll use S3 class for simplicity and return the results as an object of S3 class.

1. Custom K Means Clustering function

```
# K-means Clustering Function
kmeans_stat_analysis <- function(data,
                         centers, scale = TRUE) {
  # Scale the data if required
  if (scale) {
    data <- scale(data)
  }

  # Perform K-means clustering
  kmeans_result <- kmeans(data, centers)

  # Create an S3 object to store results
  result <- list(
    # Cluster assignments
    clusters = kmeans_result$cluster,
     # Cluster centers
    centers = kmeans_result$centers,
    # Within-cluster sum of squares
    withinss = kmeans_result$withinss,
    # Total within-cluster SS
    tot_withinss = kmeans_result$tot.withinss,
    # Cluster sizes
    size = kmeans_result$size,
    # Original (or scaled) data
    data = data
  )

  # Assign the S3 class
  class(result) <- "kmeans_stat_analysis"

  return(result)
}
```

In the above custom `kmeans_stat_analysis` function accepts data and number of centeres as input and keeps scale=TRUE by default to scale the data for kmeans clustering.

The result of kmeans clustering is stored in a variable and then we store all statistical metrics related to kmeans in `result` and then this list is assigned to a custom S3 class *kmeans_stat_analysis,* this makes the object compatible with other custom methods created below.

2. Custom `print()` Function

Creating custom `print()` method for S3 object, this method displays the number of clusters, cluster sizes and total within-cluster sum of squares metric for the kmeans statistics.

```
# Print method for K-means Analysis
print.kmeans_stat_analysis <- function(x, ...) {
  cat("K-means Clustering:\n")
  cat("Number of Clusters: ", length(x$size), "\n")
  cat("Cluster Sizes: ", paste(x$size, collapse = ", "), "\n")
}
```

3. Custom `summary()` Function

The `summary()` method provides detailed statistics for each cluster.
I am just showing cluster centers, cluster sizes and within cluster sum of squares.

```
# Summary method for K-means Analysis
summary.kmeans_stat_analysis <- function(object, ...) {
  cat("K-means Clustering Detailed Summary:\n")
  cat("\nCluster Centers:\n")
  print(round(object$centers, 2))

  cat("\nCluster Sizes:\n")
  print(object$size)

  cat("\nWithin-Cluster Sum of Squares:\n")
  print(round(object$withinss, 2))
}
```

4. Custom `plot()` function

The plot function plots the scatter plot for PCA1 and PCA2 of the k-means clustering, I chose only these two analysis, as combined, these two gives the most information on the k-means clustering by dimensionality reduction and we get very good estimates from these.

```
# Plot method for K-means Analysis
plot.kmeans_stat_analysis <- function(x, ...) {
  library(ggplot2)

  # Convert data into a data frame for ggplot
  plot_data <- as.data.frame(x$data)
  plot_data$Cluster <- as.factor(x$clusters)

  # Use the first two principal components for visualization
  # due to dimesionality reduction
  pca_result <- prcomp(plot_data[, -ncol(plot_data)])
  plot_data$PC1 <- pca_result$x[, 1]
  plot_data$PC2 <- pca_result$x[, 2]

  # Scatter plot of clusters
  ggplot(plot_data, aes(x = PC1, y = PC2, color = Cluster)) +
    geom_point(size = 3, alpha = 0.7) +
    labs(
      title = "K-means Clustering Visualization",
      x = "Principal Component 1",
      y = "Principal Component 2"
    ) +
    theme_minimal()+
    theme(plot.title = element_text(hjust = 0.5))
}
```

5. Demonstrating the functions

I will be using the Breast Cancer dataset that I used in Part 2: R Package

a. Calling the custom `kmeans_stat_analysis()` method

```
# Remove the diagnosis column for clustering
cancer_data_new <- cancer_data[, -2]

# Perform K-means Clustering
# For reproducibility
set.seed(42)
kmeans_result <- kmeans_stat_analysis(cancer_data_new, centers = 2)
```

I did some preprocessing of data by just removing the target column from the dataset as K-means is an unsupervised learning method.

b. Calling the custom `print()` method

```
# Print the K-means summary
print(kmeans_result)
```

```
K-means Clustering:
Number of Clusters:  2
Cluster Sizes:  375, 193
```

c. Calling the `summary()` method

```
# Show detailed summary
summary(kmeans_result)
```

```
K-means Clustering Detailed Summary:

Cluster Centers:
  feature_1 feature_3 feature_4 feature_5 feature_6 feature_7 feature_8
1     -0.04     -0.48     -0.24     -0.50     -0.48     -0.31     -0.52
2      0.07      0.93      0.47      0.97      0.92      0.60      1.02
  feature_9 feature_10 feature_11 feature_12 feature_13 feature_14 feature_15
1     -0.58      -0.58      -0.31      -0.15      -0.43      -0.02      -0.43
2      1.12       1.14       0.59       0.28       0.83       0.04       0.83
  feature_16 feature_17 feature_18 feature_19 feature_20 feature_21 feature_22
1      -0.40      -0.02      -0.37      -0.33      -0.40      -0.07      -0.23
2       0.77       0.05       0.71       0.65       0.77       0.14       0.46
  feature_23 feature_24 feature_25 feature_26 feature_27 feature_28 feature_29
1      -0.52      -0.25      -0.53      -0.50      -0.31      -0.49      -0.53
2       1.00       0.49       1.03       0.97       0.61       0.95       1.04
  feature_30 feature_31 feature_32
1      -0.58      -0.30      -0.33
2       1.12       0.58       0.65

Cluster Sizes:
[1] 375 193

Within-Cluster Sum of Squares:
[1] 5300.05 6836.07
```

d. Calling the `plot()` method

```
# Plot the clustering result
plot(kmeans_result)
```

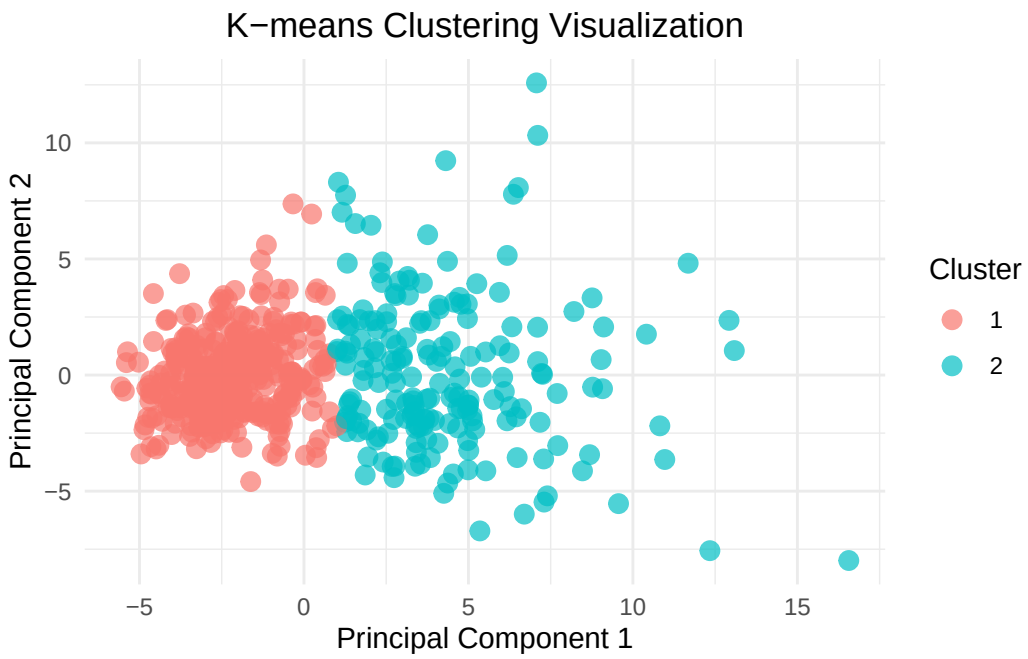

Figure 7: Scatter plot of PCA1 and PCA2 using custom plot method

References

1. Analysis data-set - https://data.cso.ie/table/DHA38
2. Breast Cancer data-set - https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic
3. Caret package - https://cran.r-project.org/web/packages/caret/index.html
4. Caret documentation - https://topepo.github.io/caret/
5. Learning on Random Forest
    a. https://rpubs.com/phamdinhkhanh/389752
    b. https://www.geeksforgeeks.org/building-a-randomforest-with-caret/
    c. https://montilab.github.io/BS831/articles/docs/Classification_Caret.html
6. Confusion matrix resource referred - https://www.digitalocean.com/community/tutorials/confusion-matrix-in-r
7. S3 Class documentation - https://rstudio-education.github.io/hopr/s3.html