Lab File

Fundamentals of Data Science Lab

Session: Jan - May 2025

Programme: BTech. CS - Data Science

Sem: 4
Batch: 5

Submitted By:

Name: Kshitij Chandrakar
SAP ID: 500124827

Submitted To:

Dr. Sachi Chaudhary

Index

Exp. No.	Objective	Date of Performance	Date of Submission
1	Probability and Statistics using R	7 Feb 2025	
2	Basic data exploration: summary statistics, histograms, scatterplots	14 Feb 2025	
3	Data cleaning: handling missing data, outliers, imputation	21 Feb 2025	

Experiment 1

Question 1: Ask the user to enter a die face (1-6). Compute the probability of rolling that number in a fair die roll

Code

```
die_face <- as.integer(system("read -p 'Enter a Die Number(1-6): ' input; echo $input", intern = TRUE))
print(die_face)
if (die_face >= 1 & die_face <= 6) {
  cat("Probability:", 1/6, "\n")
} else {
  cat("Invalid input.\n")
}
```

Output

Question 2: Ask the user for two numbers (1-6) and compute the probability of rolling both

Code

```r
# 2. Probability of rolling two specific numbers
num1 <- as.integer(system("read -p 'Enter first number (1-6): ' input; echo ", intern=TRUE))
num2 <- as.integer(system("read -p 'Enter second number (1-6): ' input; echo ", intern=TRUE))
cat("Probability of rolling both:", (1/6) * (1/6), "\n")
```

Output

Question 3: Let the user input two sets of numbers (e.g,, evens multiples of 3) and compute the union probability.

Code

```
set2 <- as.integer(strsplit(system("read -p 'Enter second set of numbers (comma-separated, 1-6): ' input; echo ", intern=TRUE), ","))
union_prob <- length(unique(c(set1, set2))) / 6
cat("Union probability:", union_prob, "\n")
```

Output

Question 4: Let the user enter prior probability, sensitivity, and false positive rate to compute the probability of having a disease given a positive test.

Code

```
command = paste("read -p 'Enter prior probability of disease: ' input; echo $input", sep="")

prior <- as.numeric(system("read -p 'Enter prior probability of disease: ' input; echo $input", intern=TRUE))
sensitivity <- as.numeric(system("read -p 'Enter sensitivity (true positive rate):  ' input; echo $input",intern=TRUE))
false_positive <- as.numeric(system("read -p 'Enter false positive rate: ' input; echo $input", intern=TRUE))
posterior <- (sensitivity * prior) / ((sensitivity * prior) + (false_positive * (1 - prior)))
cat("Probability of having disease given a positive test:", posterior, "\n")
```

Output

Question 5: Let the user roll a die n times and compute the probability of each outcome.

Code

```
n <- as.integer(system("read -p 'Enter number of die rolls: ' input; echo ", intern=TRUE))
n <- 3
rolls <- sample(1:6, n, replace = TRUE)
probabilities <- table(rolls) / n
print(probabilities)
```

Output

Question 6: Ask the user for a mean and standard deviation, generate 1000 random normal values, and plot them.

Code

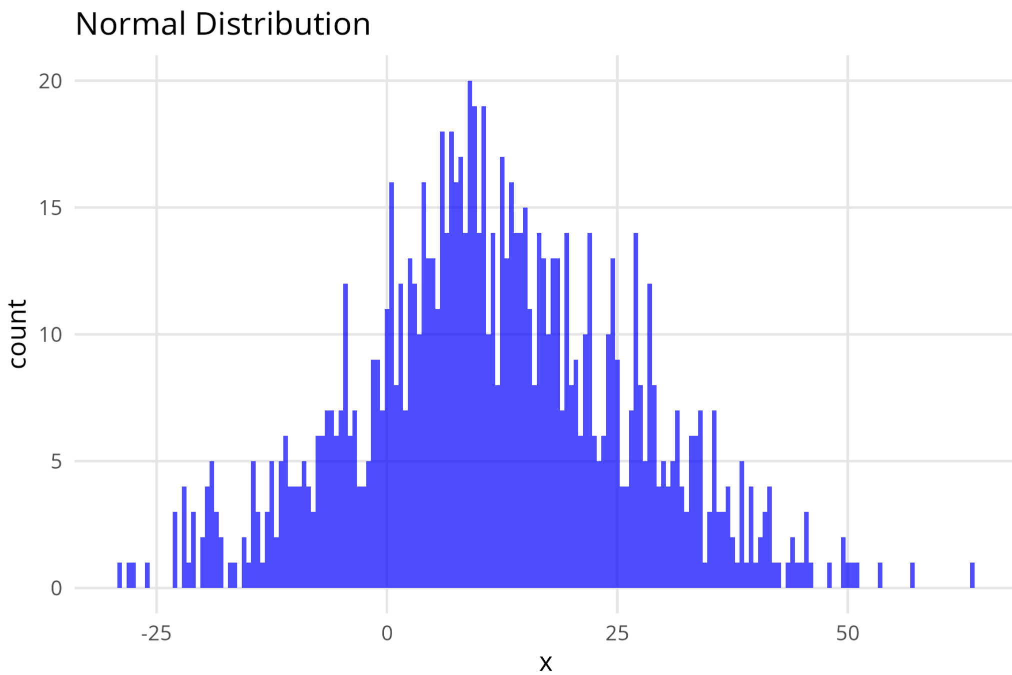
```
# 6. Generate and plot 1000 random normal values
mean_value <- as.numeric(system("read -p 'Enter mean: ' input;
↪  echo $input", intern=TRUE))
sd_value <- as.numeric(system("read -p 'Enter standard
↪  deviation: ' input; echo $input", intern=TRUE))

# Generate random values from normal distribution
random_values <- rnorm(1000, mean_value, sd_value)

# Create the plot
p <- ggplot(data.frame(x = random_values), aes(x)) +
  geom_histogram(binwidth = 0.5, fill = "blue", alpha = 0.7) +
  theme_minimal() +
  ggtitle("Normal Distribution") +
  theme(
    panel.background = element_rect(fill = "white", color =
     ↪  "white"),  # White panel
    plot.background = element_rect(fill = "white", color =
     ↪  "white"),   # White plot area
    panel.grid.major = element_line(color = "gray90"),
     ↪  # Light grid lines
    panel.grid.minor = element_blank()
     ↪  # Remove minor grid lines
  )


# Save the plot
ggsave("normal_distribution.png", plot = p, width = 6, height =
↪  4, dpi = 300)
```

Output



Question 7: Let the user enter the cost of'a game and the outcomes to compute expected value.

Code

```
outcomes <- as.character(system("read -p 'Enter possible outcomes (comma-separated): ' input; echo $input", intern=TRUE))
outcomes <- as.numeric(strsplit(outcomes, ",")[[1]])
probabilities <- system("read -p 'Enter their probabilities (comma-separated): ' input; echo $input", intern=TRUE)
probabilities <- as.numeric(strsplit(probabilities, ",")[[1]])
expected_value <- sum(outcomes * probabilities) - cost
cat("Expected Value:", expected_value, "\n")
```

Output

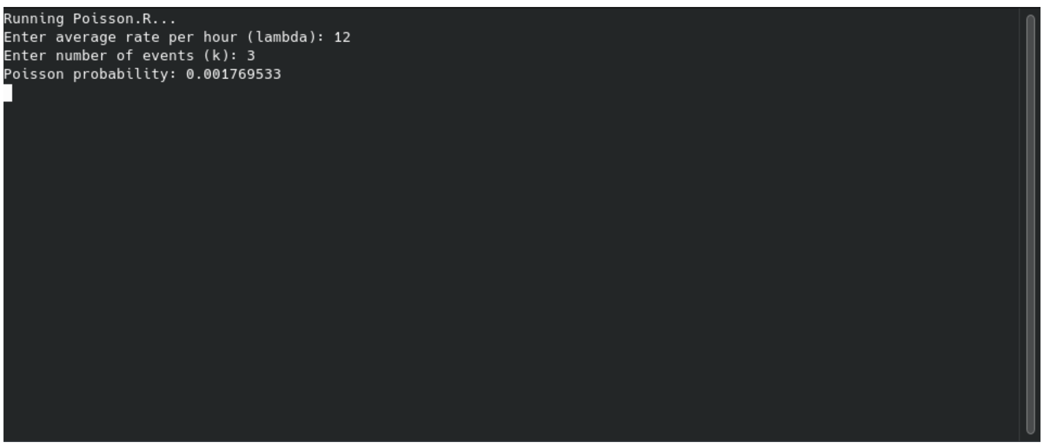
```
Running ExpectedValue.R...
Enter cost of the game: 12
Enter possible outcomes (comma-separated): 3,4,5
Enter their probabilities (comma-separated): 0,0.1,0.9
Expected Value: -7.1
```

Question 8: Let the user enter lambda (average rate per hour) and k (specific number of events) to compute the probability using a Poisson distribution.

Code

```
k <- as.integer(system("read -p 'Enter number of events (k): '
↪  input; echo $input", intern=TRUE))
poisson_prob <- dpois(k, lambda)
cat("Poisson probability:", poisson_prob, "\n")
```

Output



Experiment 2

1. Summary Statistics for a Dataset

- Dataset: Built-in mtcars dataset (Car Specifications)
- Compute summary statistics (mean, median, standard deviation, etc.).
- Understand the distribution of miles per gallon (mpg) and horsepower (hp).

[1]:
```
data("mtcars")
summary(mtcars)
```

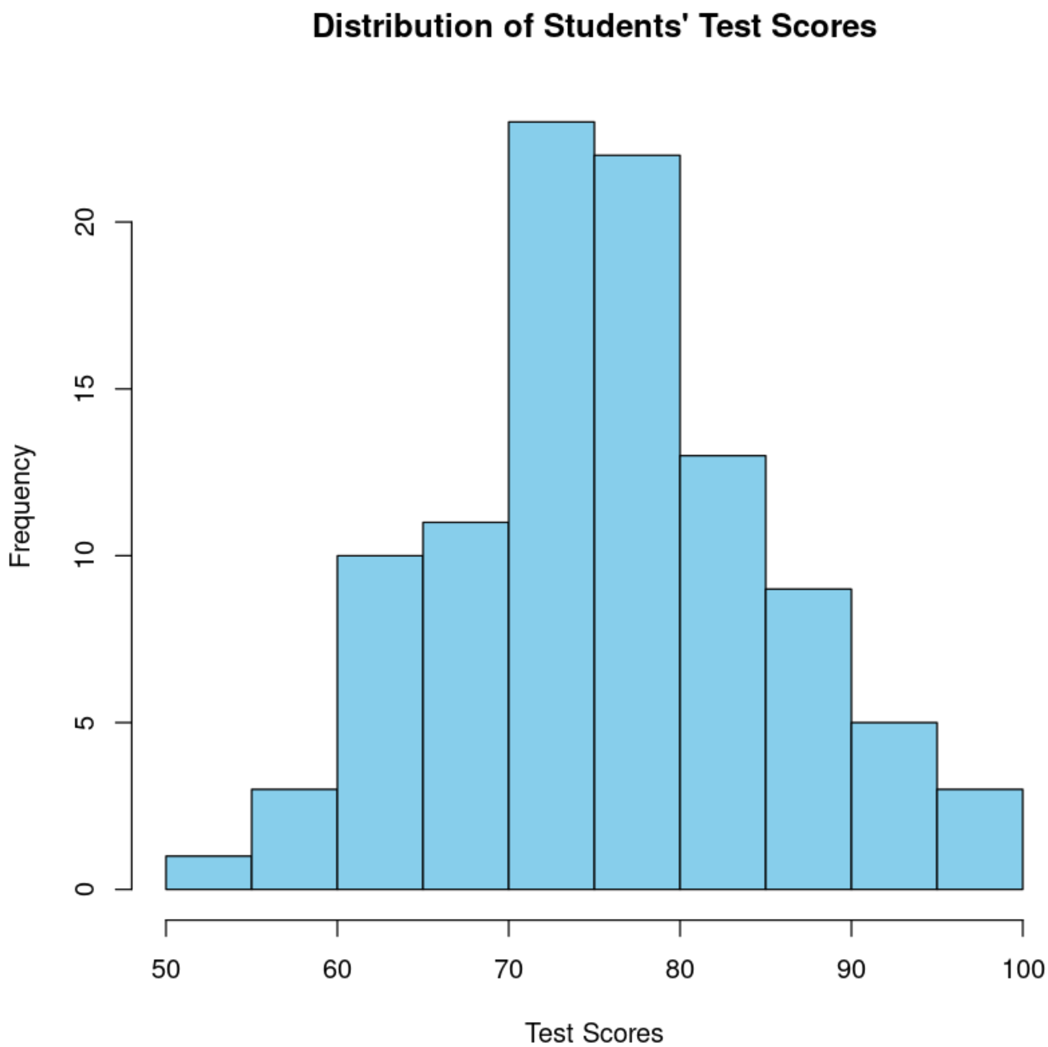
```
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

2. Create a Histogram

- Generate a random dataset of students test scores
- Visualize data distribution using histograms.
- Understand skewness and spread of data.

[2]:
```
# Generate random test scores (normal distribution, mean=75, sd=10)
set.seed(123)  # For reproducibility
scores <- rnorm(100, mean = 75, sd = 10)
# Create a histogram
hist(scores,
     main = "Distribution of Students' Test Scores",
     xlab = "Test Scores",
```

```
     col = "skyblue",
     border = "black",
    )
# Check skewness and spread
# library(moments)
# skewness(scores)  # Measure skewness
```



[3]:
```
summary(scores)    # Summary statistics
dev.off()
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  51.91   70.06   75.62   75.90   81.92   96.87
```


null device: 1

3. Scatterplot to Explore Relationships Dataset: Built-in iris dataset (Flower Measurements) The iris dataset contains sepal and petal lengths and widths of three flower species.

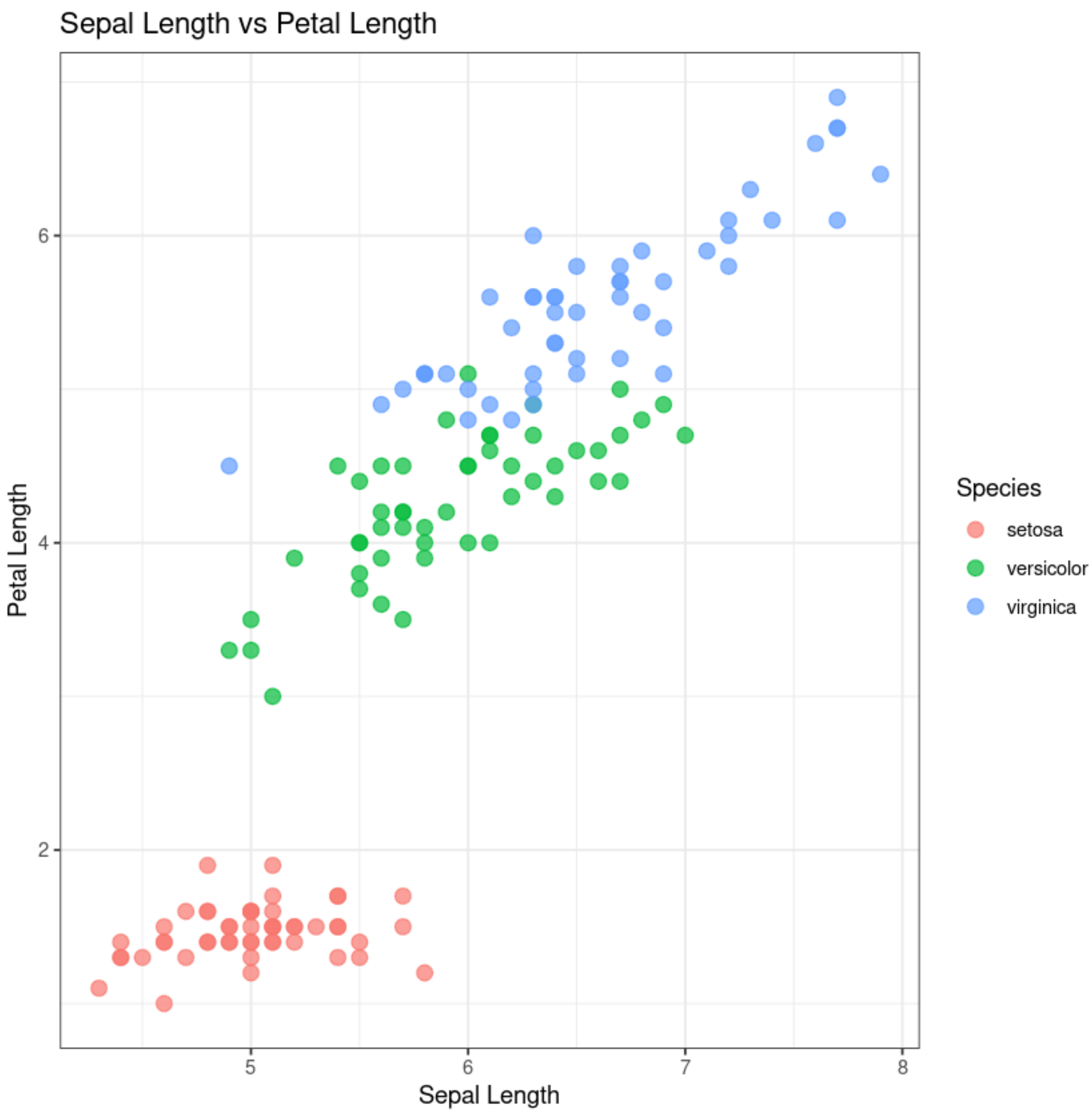
- Create a scatterplot to explore relationships between variables.
- Use colors to distinguish species.

[4]:
```
library(ggplot2)

# Load the iris dataset
data(iris)

# Scatterplot: Sepal Length vs. Petal Length
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(size = 3, alpha = 0.7) +
  labs(title = "Sepal Length vs Petal Length",
       x = "Sepal Length",
       y = "Petal Length") +
  theme_bw()

# Save the plot
ggsave("iris_scatterplot.png", width = 8, height = 6)
```



4. Boxplot for Detecting Outliers

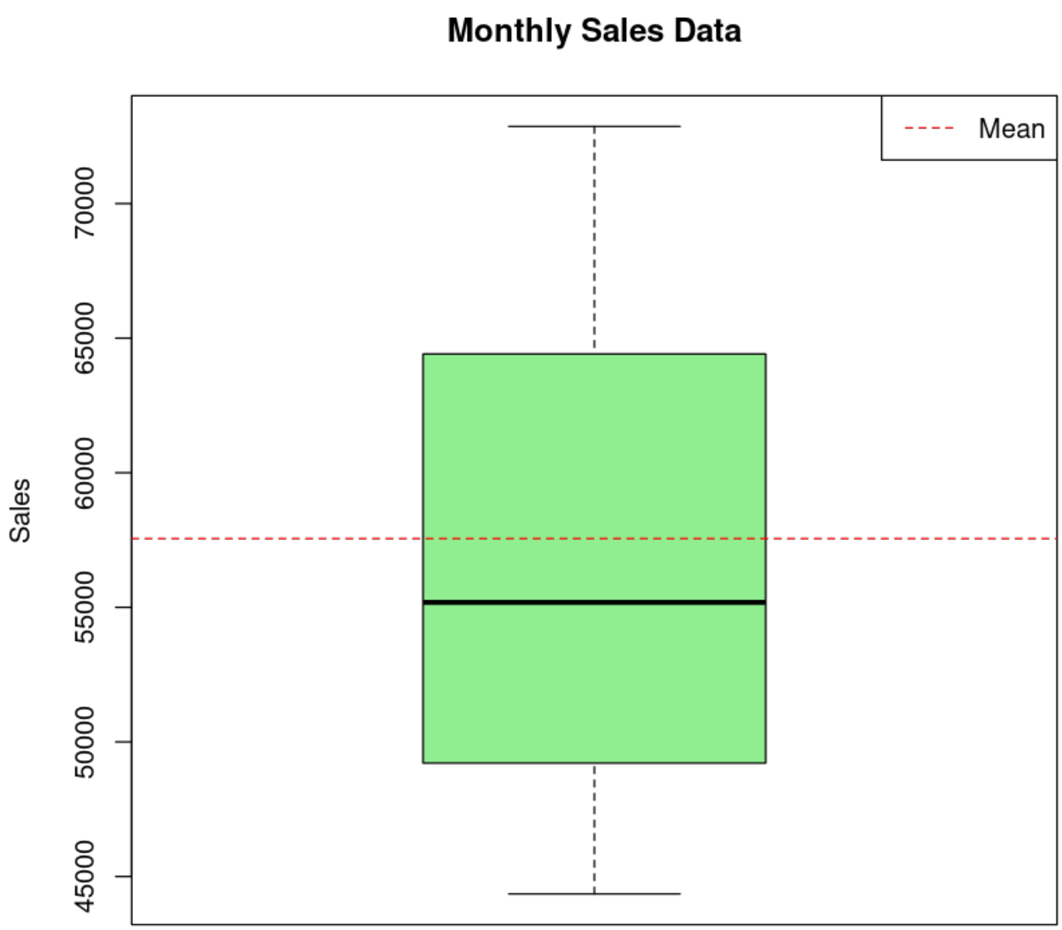
- Dataset: Simulated monthly sales data for a store. Generate random monthly sales data to analyze outliers.
- Create a boxplot to detect outliers.
- Understand quartiles and interquartile range (IQR).

[5]:
```
# Generate random monthly sales data (e.g., 12 months)
set.seed(42)
monthly_sales <- rnorm(12, mean = 50000, sd = 10000)

# Create a boxplot
boxplot(monthly_sales, col = "lightgreen", main = "Monthly Sales Data",
        ylab = "Sales", outline = TRUE)
```

14

```r
# Add a horizontal line for the mean
abline(h = mean(monthly_sales), col = "red", lty = 2)
legend("topright", legend = "Mean", col = "red", lty = 2)
```



Experiment 3

Question 1

[8]:
```r
library(tidyverse)
```

[9]:
```r
setwd("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/Experiment␣
↪3")
```

[10]:
```r
df <- data.frame(
  ID = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Name = c("Alice", "Bob", NA, "David", "Emma", "Frank", NA, "Hannah",␣
↪"Ian", "Jack"),
  Age = c(25, NA, 30, 29, NA, 35, 40, NA, 50, 27),
  Salary = c(50000, 60000, 55000, NA, 70000, 75000, 80000, 65000, NA,␣
↪72000),
  Score = c(80, 90, NA, 85, 88, 92, NA, 77, 95, Inf)
)
```

[11]:
```r
###############################
# Exploring Inbuilt Functions
###############################
```

[12]:
```r
is.na(df)
```

A matrix: 10 × 5 of type lgl

ID	Name	Age	Salary	Score
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE

[13]:
```r
complete.cases(df)
```

1. TRUE 2. FALSE 3. FALSE 4. FALSE 5. FALSE 6. TRUE 7. FALSE 8. FALSE 9. FALSE 10. TRUE

[14]:
```r
df[complete.cases(df), ]
```

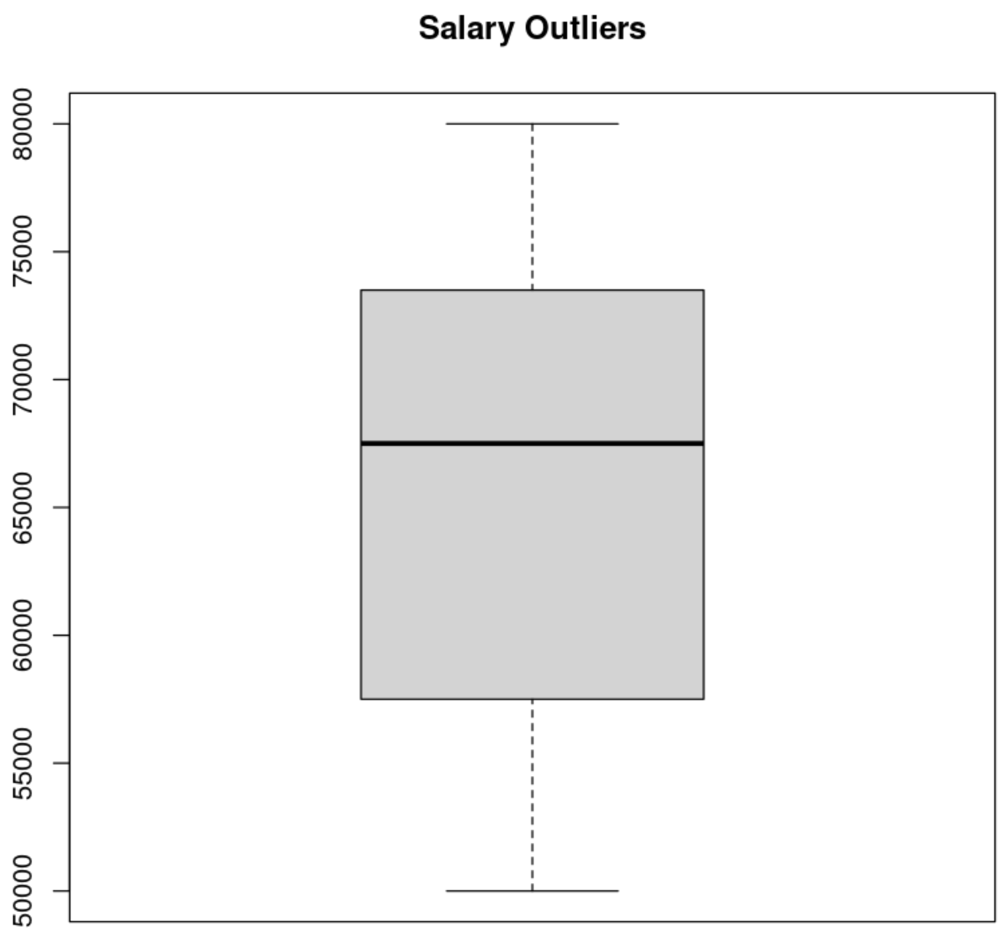
A data.frame: 3 × 5

	ID <dbl>	Name <chr>	Age <dbl>	Salary <dbl>	Score <dbl>
1	1	Alice	25	50000	80
6	6	Frank	35	75000	92
10	10	Jack	27	72000	Inf

[15]:
```
summary(df)
```

```
       ID            Name                  Age            Salary
 Min.   : 1.00   Length:10          Min.   :25.00   Min.   :50000
 1st Qu.: 3.25   Class :character   1st Qu.:28.00   1st Qu.:58750
 Median : 5.50   Mode  :character   Median :30.00   Median :67500
 Mean   : 5.50                      Mean   :33.71   Mean   :65875
 3rd Qu.: 7.75                      3rd Qu.:37.50   3rd Qu.:72750
 Max.   :10.00                      Max.   :50.00   Max.   :80000
                                    NA's   :3       NA's   :2
     Score
 Min.   :77.00
 1st Qu.:83.75
 Median :89.00
 Mean   :  Inf
 3rd Qu.:92.75
 Max.   :  Inf
 NA's   :2
```

[16]:
```
# Boxplot to detect outliers
boxplot(df$Salary, main = "Salary Outliers", horizontal = FALSE)
```



[17]:
```
# Identify outliers using IQR
Q1 <- quantile(df$Salary, 0.25, na.rm = TRUE)
Q3 <- quantile(df$Salary, 0.75, na.rm = TRUE)
IQR <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR
outliers <- df$Salary[df$Salary < lower_bound | df$Salary > upper_bound]
print(outliers)
```

```
[1] NA NA
```

[18]:
```
iqr_value <- IQR(df$Salary, na.rm=TRUE)
print(iqr_value)
```

```
[1] 14000
```

[19]:
```
df_clean <- na.omit(df)
print(df_clean)
```

```
   ID  Name Age Salary Score
1   1 Alice  25  50000    80
6   6 Frank  35  75000    92
10 10  Jack  27  72000   Inf
```

[20]:
```
df$Age[is.na(df$Age)] <- mean(df$Age, na.rm = TRUE)
df$Salary[is.na(df$Salary)] <- mean(df$Salary, na.rm = TRUE)
df$Score[is.na(df$Score)] <- mean(df$Score, na.rm = TRUE)
print(df)
```

```
   ID   Name      Age Salary Score
1   1  Alice 25.00000  50000    80
2   2    Bob 33.71429  60000    90
3   3   <NA> 30.00000  55000   Inf
4   4  David 29.00000  65875    85
5   5   Emma 33.71429  70000    88
6   6  Frank 35.00000  75000    92
7   7   <NA> 40.00000  80000   Inf
8   8 Hannah 33.71429  65000    77
9   9    Ian 50.00000  65875    95
10 10   Jack 27.00000  72000   Inf
```

Experiment 3

Question 2

[3]:
```
library(tidyverse)
```

```
 Attaching core tidyverse packages
 tidyverse 2.0.0
 dplyr     1.1.4     readr     2.1.5
 forcats   1.0.0     stringr   1.5.1
 ggplot2   3.5.1     tibble    3.2.1
 lubridate 1.9.4     tidyr     1.3.1
 purrr     1.0.4
 Conflicts

tidyverse_conflicts()
 dplyr::filter() masks stats::filter()
 dplyr::lag()    masks stats::lag()
 Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

[4]:
```
setwd("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/Experiment␣
↪3")
```

[5]:
```
df_mean <- data.frame(
  ID = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Name = c("Alice", "Bob", NA, "David", "Emma", "Frank", NA, "Hannah",␣
↪"Ian", "Jack"),
  Age = c(25, NA, 30, 29, NA, 35, 40, NA, 50, 27),
  Salary = c(50000, 60000, 55000, NA, 70000, 75000, 80000, 65000, NA,␣
↪72000),
  Score = c(80, 90, NA, 85, 88, 92, NA, 77, 95, Inf)
)
```

Identify missing data (is.na(df), sum(is.na(df))).

[6]:
```
# i. Identify missing data
print(is.na(df_mean))  # Identify missing values
print(sum(is.na(df_mean)))  # Count total missing values
```

```
         ID  Name   Age Salary Score
 [1,] FALSE FALSE FALSE  FALSE FALSE
 [2,] FALSE FALSE  TRUE  FALSE FALSE
 [3,] FALSE  TRUE FALSE  FALSE  TRUE
 [4,] FALSE FALSE FALSE   TRUE FALSE
```
20

```
 [5,] FALSE FALSE  TRUE  FALSE FALSE
 [6,] FALSE FALSE FALSE  FALSE FALSE
 [7,] FALSE  TRUE FALSE  FALSE  TRUE
 [8,] FALSE FALSE  TRUE  FALSE FALSE
 [9,] FALSE FALSE FALSE   TRUE FALSE
[10,] FALSE FALSE FALSE  FALSE FALSE
[1] 9
```

Remove missing rows (na.omit(df))

[7]:
```r
df_mean_no_na <- na.omit(df_mean)
print(df_mean_no_na)
```

```
   ID  Name Age Salary Score
1   1 Alice  25  50000    80
6   6 Frank  35  75000    92
10 10  Jack  27  72000   Inf
```

Replace NA with zero (df[is.na(df)] <- 0).

[8]:
```r
df_mean_zero <- df_mean
df_mean_zero[is.na(df_mean_zero)] <- 0
print(df_mean_zero)
```

```
   ID   Name Age Salary Score
1   1  Alice  25  50000    80
2   2    Bob   0  60000    90
3   3      0  30  55000     0
4   4  David  29      0    85
5   5   Emma   0  70000    88
6   6  Frank  35  75000    92
7   7      0  40  80000     0
8   8 Hannah   0  65000    77
9   9    Ian  50      0    95
10 10   Jack  27  72000   Inf
```

Replace NA with column mean (df*Age*[*is.na*(*df*Age)] <- mean(df$Age, na.rm=TRUE)).

[9]:
```r
df_mean_mean <- df_mean

df_mean$Age[is.na(df_mean$Age)] <- mean(df_mean$Age, na.rm = TRUE)
df_mean$Salary[is.na(df_mean$Salary)] <- mean(df_mean$Salary, na.rm = 
 ↪TRUE)
df_mean$Score[is.na(df_mean$Score)] <- mean(df_mean$Score, na.rm = TRUE)

print(df_mean_mean)
```

```
   ID   Name Age Salary Score
1   1  Alice  25  50000    80
2   2    Bob  NA  60000    90
3   3   <NA>  30  55000    NA
4   4  David  29     NA    85
5   5   Emma  NA  70000    88
6   6  Frank  35  75000    92
7   7   <NA>  40  80000    NA
8   8 Hannah  NA  65000    77
9   9    Ian  50     NA    95
10 10   Jack  27  72000   Inf
```

Remove Inf and NaN (df$Score[is.infinite(df$Score) | is.nan(df$Score)] <- NA)

[10]:
```
df_mean_clean <- df_mean
df_mean_clean$Score[is.infinite(df_mean_clean$Score) | is.
 ↪nan(df_mean_clean$Score)] <- NA
print(df_mean_clean)
```

```
   ID   Name      Age Salary Score
1   1  Alice 25.00000  50000    80
2   2    Bob 33.71429  60000    90
3   3   <NA> 30.00000  55000    NA
4   4  David 29.00000  65875    85
5   5   Emma 33.71429  70000    88
6   6  Frank 35.00000  75000    92
7   7   <NA> 40.00000  80000    NA
8   8 Hannah 33.71429  65000    77
9   9    Ian 50.00000  65875    95
10 10   Jack 27.00000  72000    NA
```

Use tidyverse's replace_na() for selective column handling.

[11]:
```
df_mean_tidy <- df_mean %>%
  mutate(
    Age = replace_na(Age, mean(Age, na.rm = TRUE)),
    Salary = replace_na(Salary, median(Salary, na.rm = TRUE))
  )
print(df_mean_tidy)
```

```
   ID   Name      Age Salary Score
1   1  Alice 25.00000  50000    80
2   2    Bob 33.71429  60000    90
3   3   <NA> 30.00000  55000   Inf
4   4  David 29.00000  65875    85
5   5   Emma 33.71429  70000    88     22
6   6  Frank 35.00000  75000    92
```

```
7    7   <NA> 40.00000  80000   Inf
8    8 Hannah 33.71429  65000    77
9    9    Ian 50.00000  65875    95
10  10   Jack 27.00000  72000   Inf
```

Drop columns with excessive missing data (df <- df[, colSums(is.na(df)) < nrow(df) * 0.5])

[12]:
```
df_mean_filtered <- df_mean[, colSums(is.na(df_mean)) < (nrow(df_mean) *
 ↪0.5)]
print(df_mean_filtered)
```

```
   ID   Name      Age Salary Score
1   1  Alice 25.00000  50000    80
2   2    Bob 33.71429  60000    90
3   3   <NA> 30.00000  55000   Inf
4   4  David 29.00000  65875    85
5   5   Emma 33.71429  70000    88
6   6  Frank 35.00000  75000    92
7   7   <NA> 40.00000  80000   Inf
8   8 Hannah 33.71429  65000    77
9   9    Ian 50.00000  65875    95
10 10   Jack 27.00000  72000   Inf
```

Fill missing categorical values with the mode.

[13]:
```
# viii. Fill missing categorical values with mode
fill_mode <- function(x) {
  if (is.character(x)) {
    mode_value <- names(sort(table(x), decreasing = TRUE))[1]
    x[is.na(x)] <- mode_value
  }
  return(x)
}
df_mean_mode <- df_mean
df_mean_mode$Name <- fill_mode(df_mean_mode$Name)
print(df_mean_mode)
```

```
   ID   Name      Age Salary Score
1   1  Alice 25.00000  50000    80
2   2    Bob 33.71429  60000    90
3   3  Alice 30.00000  55000   Inf
4   4  David 29.00000  65875    85
5   5   Emma 33.71429  70000    88
6   6  Frank 35.00000  75000    92
7   7  Alice 40.00000  80000   Inf
8   8 Hannah 33.71429  65000    77
9   9    Ian 50.00000  65875    95
```

```
10 10    Jack 27.00000  72000    Inf
```

Experiment 3

Question 3

0.1 Outlier Detection & Handling

0.1.1 Preprocessing

[9]:
```
library(tidyverse)
```

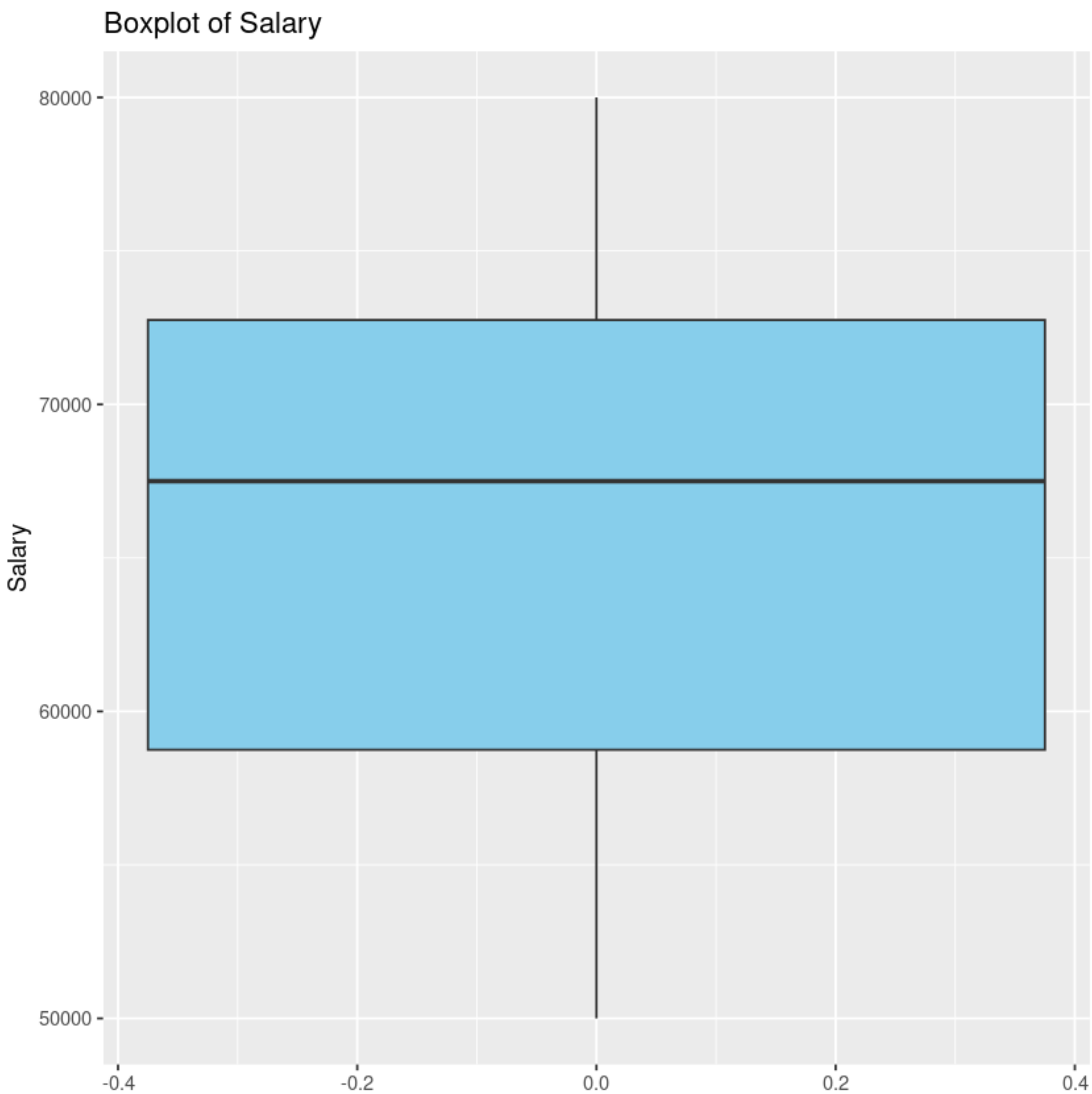
[10]:
```
setwd("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/Experiment␣
↪3")
```

[11]:
```
df_mean <- data.frame(
  ID = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Name = c("Alice", "Bob", NA, "David", "Emma", "Frank", NA, "Hannah",␣
↪"Ian", "Jack"),
  Age = c(25, NA, 30, 29, NA, 35, 40, NA, 50, 27),
  Salary = c(50000, 60000, 55000, NA, 70000, 75000, 80000, 65000, NA,␣
↪72000),
  Score = c(80, 90, NA, 85, 88, 92, NA, 77, 95, Inf)
)
```

Boxplot Visualization to visualize salary data

[12]:
```
# i. Boxplot Visualization to visualize Salary data
ggplot(df_mean, aes(y = Salary)) +
  geom_boxplot(fill = "skyblue", outlier.color = "red", outlier.shape =␣
↪16) +
  labs(title = "Boxplot of Salary", y = "Salary")
```

```
Warning message:
“Removed 2 rows containing non-finite outside the scale range
(`stat_boxplot()`).”
```



Z-Score Method (values outside ±3 standard deviations).

[13]:
```
# ii. Z-Score Method (Values outside ±3 standard deviations)
df_mean_z <- df_mean %>%
  mutate(Salary_Z = as.numeric(scale(Salary))) %>%  # Convert scale␣
↪output to numeric
  filter(abs(Salary_Z) <= 3) %>%  # Remove outliers
  select(-Salary_Z)  # Remove Z-score column
print(df_mean_z)
```

```
  ID   Name Age Salary Score
1  1  Alice  25  50000    80
2  2    Bob  NA  60000    90
3  3   <NA>  30  55000    NA
```

26

```
4  5   Emma  NA  70000    88
5  6  Frank  35  75000    92
6  7   <NA>  40  80000    NA
7  8 Hannah  NA  65000    77
8 10   Jack  27  72000   Inf
```

iii. IQR Method: Remove values outside Q1 - 1.5*IQR and Q3 + 1.5*IQR.

[15]:
```r
# iii. IQR Method: Remove values outside Q1 - 1.5*IQR and Q3 + 1.5*IQR
Q1 <- quantile(df_mean$Salary, 0.25, na.rm=TRUE)
Q3 <- quantile(df_mean$Salary, 0.75, na.rm=TRUE)
IQR_value <- Q3 - Q1
lower_bound <- Q1 - 1.5 * IQR_value
upper_bound <- Q3 + 1.5 * IQR_value
```

[16]:
```r
df_mean_iqr <- df_mean %>%
  filter(Salary >= lower_bound & Salary <= upper_bound)
```

iv. Winsorization: Replace extreme values with percentiles (Winsorize()).

[8]:
```r
# iv. Winsorization: Replace extreme values with 5th and 95th percentiles
library(DescTools)
df_mean_winsorized <- df_mean %>%
  mutate(Salary = Winsorize(Salary, probs = c(0.05, 0.95)))
```

```
Error in `mutate()`:
 In argument: `Salary = Winsorize(Salary, probs = c(0.05, 0.95))`.
Caused by error in `Winsorize()`:
! unused argument (probs = c(0.05, 0.95))
Traceback:

1. mutate(., Salary = Winsorize(Salary, probs = c(0.05, 0.95)))
2. mutate.data.frame(., Salary = Winsorize(Salary, probs = c(0.05,
 .     0.95)))
3. mutate_cols(.data, dplyr_quosures(...), by)
4. withCallingHandlers(for (i in seq_along(dots)) {
 .     poke_error_context(dots, i, mask = mask)
 .     context_poke("column", old_current_column)
 .     new_columns <- mutate_col(dots[[i]], data, mask, new_columns)
 . }, error = dplyr_error_handler(dots = dots, mask = mask, bullets =
↪mutate_bullets,
 .     error_call = error_call, error_class = "dplyr:::mutate_error"),
 .     warning = dplyr_warning_handler(state = warnings_state, mask = mask
 .         error_call = error_call))
5. mutate_col(dots[[i]], data, mask, new_columns)
6. mask$eval_all_mutate(quo)
```

```
7. eval()
8. .handleSimpleError(function (cnd)
 . {
 .     local_error_context(dots, i = frame[[i_sym]], mask = mask)
 .     if (inherits(cnd, "dplyr:::internal_error")) {
 .         parent <- error_cnd(message = bullets(cnd))
 .     }
 .     else {
 .         parent <- cnd
 .     }
 .     message <- c(cnd_bullet_header(action), i = if␣
 ↪(has_active_group_context(mask)) cnd_bullet_cur_group_label())
 .     abort(message, class = error_class, parent = parent, call =␣
 ↪error_call)
 . }, "unused argument (probs = c(0.05, 0.95))", base::
 ↪quote(Winsorize(Salary,
 .     probs = c(0.05, 0.95))))
9. h(simpleError(msg, call))
10. abort(message, class = error_class, parent = parent, call = error_call
11. signal_abort(cnd, .file)
12. signalCondition(cnd)
```

v. Detect & Remove Outliers Using tidyverse (filter())

[17]:
```
# v. Detect & Remove Outliers Using tidyverse (filter method)
df_mean_tidy_outliers <- df_mean %>%
  filter(between(Salary, lower_bound, upper_bound))
```

vi. Detect Outliers in Multiple Columns (apply()).

[19]:
```
# vi. Detect Outliers in Multiple Columns using apply() (Z-score method)
detect_outliers <- function(x) {
  if (is.numeric(x)) {
    z_scores <- scale(x)
    return(abs(z_scores) > 3)
  } else {
    return(rep(FALSE, length(x)))
  }
}

outlier_matrix <- apply(df_mean, 2, detect_outliers)
df_mean_clean <- df_mean[!rowSums(outlier_matrix), ]  # Remove rows with␣
 ↪outliers
```


vii. Create a Clean Dataset After Removing Outliers

[21]:
```
# vii. Create a Clean Dataset After Removing Outliers
df_mean_final <- df_mean_iqr  # Using IQR method for final clean dataset
write.csv(df_mean_final, "Clean_Dataset.csv", row.names = FALSE)
```

Experiment 3

Question 4

0.1 Data Imputation

[1]:
```
### Preprocessing
```

[2]:
```
library(tidyverse)
```

```
Attaching core tidyverse packages
tidyverse 2.0.0
dplyr     1.1.4      readr     2.1.5
forcats   1.0.0      stringr   1.5.1
ggplot2   3.5.1      tibble    3.2.1
lubridate 1.9.4      tidyr     1.3.1
purrr     1.0.4
Conflicts

tidyverse_conflicts()
dplyr::filter() masks stats::filter()
dplyr::lag()    masks stats::lag()
Use the conflicted package
(<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

[3]:
```
setwd("/home/asus/content/Notes/Semester 4/FDN Lab/Experiments/Experiment␣
↪3")
```

[4]:
```
df <- data.frame(
  ID = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10),
  Name = c("Alice", "Bob", NA, "David", "Emma", "Frank", NA, "Hannah",␣
↪"Ian", "Jack"),
  Age = c(25, NA, 30, 29, NA, 35, 40, NA, 50, 27),
  Salary = c(50000, 60000, 55000, NA, 70000, 75000, 80000, 65000, NA,␣
↪72000),
  Score = c(80, 90, NA, 85, 88, 92, NA, 77, 95, Inf)
)
```

Convert NaN and Inf values to NA before applying imputation.

[5]:
```
df <- df %>%
  mutate_all(~ ifelse(. == Inf | . == -Inf, NA, .)) %>%
  mutate_all(~ ifelse(is.nan(.), NA, .))
```


Remove rows with missing values using na.omit(df).

[6]:
```
df_no_na <- na.omit(df)  # Remove rows with any NA
```

Drop columns where more than 50% of data is missing.

[7]:
```
df <- df[, colSums(is.na(df)) < (0.5 * nrow(df))]
```

Replace all NA values with 0 for numerical columns.

[8]:
```
df[sapply(df, is.numeric)] <- lapply(df[sapply(df, is.numeric)],␣
↪function(x) { replace(x, is.na(x), 0) })
```

Replace missing values in Age with the mean.

[9]:
```
df$Age[is.na(df$Age)] <- mean(df$Age, na.rm = TRUE)
```

Replace missing values in Salary with the median.

[10]:
```
df$Salary[is.na(df$Salary)] <- median(df$Salary, na.rm = TRUE)
```

Replace missing Name values with the most frequent name (Mode)

[11]:
```
fill_mode <- function(x) {
  mode_value <- names(sort(table(x), decreasing = TRUE))[1]
  x[is.na(x)] <- mode_value
  return(x)
}

df$Name <- fill_mode(df$Name)  # Apply mode function to Name column
```

Summary

[12]:
```
summary(df)  # Check if missing values are handled
```

```
       ID            Name                Age            Salary
 Min.   : 1.00   Length:10          Min.   : 0.00   Min.   :    0
 1st Qu.: 3.25   Class :character   1st Qu.: 6.25   1st Qu.:51250
 Median : 5.50   Mode  :character   Median :28.00   Median :62500
 Mean   : 5.50                      Mean   :23.60   Mean   :52700
 3rd Qu.: 7.75                      3rd Qu.:33.75   3rd Qu.:71500
 Max.   :10.00                      Max.   :50.00   Max.   :80000
     Score
 Min.   : 0.00
 1st Qu.:19.25
 Median :82.50
 Mean   :60.70
 3rd Qu.:89.50
 Max.   :95.00
```