# Experiment 02 – Crash Site-Based Vector Analysis

## Details:

Name: Sparsh Karna

Registration Number: 23BDS1172

Slot: L21 + L22

## 1. Objective:

To perform crash site-based survival analysis using only one-dimensional vectors and user-defined functions in R.

## 2. Methodology:

- Vectors created manually for each attribute: age, weight, survival, gender, class, health\_score, and crash\_site.  
- Custom functions implemented to compute required statistics using only base R functions.  
- All data manipulation is done using conditional indexing without using data.frame or lists.  
- cat() and print() are used to format output for summary and analysis.

## 3. R Code with Comments:

# --------------------------------------------------

# Name: Sparsh Karna

# Registration No: 23BDS1172

# Experiment No: 02

# Title: Crash Site-Based Vector Analysis

# Date: 24/Jul/2025

# --------------------------------------------------

# Vector Initialization Section

age <- sample(18:70, 30, replace=TRUE) # Generate 30 ages between 18 and 70

weight <- round(runif(30, 50, 100), 1) # Generate 30 weights between 50 and 100 kg

survival <- sample(c(0, 1), 30, replace=TRUE) # Generate binary survival status (0=deceased, 1=survived)

gender <- sample(c("M", "F"), 30, replace=TRUE) # Generate gender as M or F

class <- sample(c("E", "B", "F"), 30, replace=TRUE) # Generate class as Economy (E), Business (B), or First (F)

health\_score <- sample(0:100, 30, replace=TRUE) # Generate health scores between 0 and 100

crash\_site <- sample(c("Forest", "Sea", "Mountain"), 30, replace=TRUE) # Generate crash site locations

# User-Defined Functions Section

#' @title: Calculate Total Survival Rate

#' @description: Computes and prints the total number of survivors and survival percentage

#' @param survival: Vector of survival status (0=deceased, 1=survived)

#' @returns None: Prints total survivors and survival rate

#' @example

#' # Calculate survival statistics

#' calculate\_survival\_rate(survival)

calculate\_survival\_rate <- function(survival) {

total\_survivors <- sum(survival) # Sum the survival vector to count survivors

percent\_survived <- (total\_survivors / length(survival)) \* 100 # Calculate percentage

cat("Total Survivors:", total\_survivors, "\n") # Print number of survivors

cat("Survival Rate:", round(percent\_survived, 2), "%\n") # Print percentage rounded to 2 decimals

}

#' @title: Calculate Average Age of Survivors

#' @description: Computes and prints the average age of passengers who survived

#' @param age: Vector of passenger ages

#' @param survival: Vector of survival status

#' @returns None: Prints average age of survivors

#' @example

#' # Calculate average age of survivors

#' average\_age\_survivors(age, survival)

average\_age\_survivors <- function(age, survival) {

survivor\_ages <- age[survival == 1] # Filter ages where survival is 1

avg\_age <- mean(survivor\_ages) # Calculate mean age of survivors

cat("Average Age of Survivors:", round(avg\_age, 2), "\n") # Print rounded average age

}

#' @title: Calculate Average Health Score of Deceased

#' @description: Computes and prints the average health score of deceased passengers

#' @param health\_score: Vector of health scores

#' @param survival: Vector of survival status

#' @returns None: Prints average health score of deceased

#' @example

#' # Calculate average health score of deceased

#' average\_health\_score\_deceased(health\_score, survival)

average\_health\_score\_deceased <- function(health\_score, survival) {

deceased\_scores <- health\_score[survival == 0] # Filter health scores where survival is 0

avg\_score <- mean(deceased\_scores) # Calculate mean health score

cat("Average Health Score of Deceased:", round(avg\_score, 2), "\n") # Print rounded average

}

#' @title: Find Heaviest Survivor

#' @description: Identifies and prints the weight of the heaviest surviving passenger

#' @param weight: Vector of passenger weights

#' @param survival: Vector of survival status

#' @returns None: Prints weight of heaviest survivor

#' @example

#' # Find heaviest survivor

#' heaviest\_survivor(weight, survival)

heaviest\_survivor <- function(weight, survival) {

survivor\_weights <- weight[survival == 1] # Filter weights where survival is 1

max\_weight <- max(survivor\_weights) # Find maximum weight

cat("Heaviest Survivor Weight:", max\_weight, "kg\n") # Print heaviest weight

}

#' @title: Find Youngest First-Class Survivor

#' @description: Identifies and prints the age of the youngest first-class survivor

#' @param age: Vector of passenger ages

#' @param survival: Vector of survival status

#' @param class: Vector of passenger classes

#' @returns None: Prints age of youngest first-class survivor

#' @example

#' # Find youngest first-class survivor

#' youngest\_first\_class\_survivor(age, survival, class)

youngest\_first\_class\_survivor <- function(age, survival, class) {

first\_class\_survivors <- age[survival == 1 & class == "F"] # Filter ages where survival is 1 and class is F

min\_age <- min(first\_class\_survivors) # Find minimum age

cat("Youngest First Class Survivor Age:", min\_age, "\n") # Print youngest age

}

#' @title: Calculate Gender-Wise Survival Ratio

#' @description: Computes and prints survival percentages for male and female passengers

#' @param survival: Vector of survival status

#' @param gender: Vector of passenger genders

#' @returns None: Prints male and female survival rates

#' @example

#' # Calculate gender-wise survival ratios

#' gender\_wise\_survival\_ratio(survival, gender)

gender\_wise\_survival\_ratio <- function(survival, gender) {

male\_survivors <- sum(survival == 1 & gender == "M") # Count male survivors

female\_survivors <- sum(survival == 1 & gender == "F") # Count female survivors

total\_males <- sum(gender == "M") # Count total males

total\_females <- sum(gender == "F") # Count total females

male\_rate <- (male\_survivors / total\_males) \* 100 # Calculate male survival percentage

female\_rate <- (female\_survivors / total\_females) \* 100 # Calculate female survival percentage

cat("Male Survival Rate:", round(male\_rate, 2), "%\n") # Print male rate

cat("Female Survival Rate:", round(female\_rate, 2), "%\n") # Print female rate

}

#' @title: Categorize Health Scores

#' @description: Categorizes health scores into Low, Medium, High and prints counts

#' @param health\_score: Vector of health scores

#' @returns categories: Vector of health categories

#' @example

#' # Categorize health scores

#' health\_cats <- health\_category(health\_score)

health\_category <- function(health\_score) {

categories <- ifelse(health\_score < 35, "Low", # Assign Low if score < 35

ifelse(health\_score < 70, "Medium", "High")) # Assign Medium if < 70, else High

print(table(categories)) # Print frequency table of categories

return(categories) # Return category vector

}

#' @title: Survival by Health Category

#' @description: Counts and prints survivors in each health category

#' @param health\_score: Vector of health scores

#' @param survival: Vector of survival status

#' @returns None: Prints survivor counts per health category

#' @example

#' # Calculate survivors by health category

#' survival\_by\_health\_category(health\_score, survival)

survival\_by\_health\_category <- function(health\_score, survival) {

categories <- health\_category(health\_score) # Get health categories

survivor\_categories <- categories[survival == 1] # Filter categories for survivors

cat("Survivors by Health Category:\n") # Print header

print(table(survivor\_categories)) # Print frequency table of survivor categories

}

#' @title: Most Common Class Among Survivors

#' @description: Identifies and prints the most common passenger class among survivors

#' @param class: Vector of passenger classes

#' @param survival: Vector of survival status

#' @returns None: Prints most common class

#' @example

#' # Find most common class among survivors

#' most\_common\_class\_survived(class, survival)

most\_common\_class\_survived <- function(class, survival) {

survivor\_classes <- class[survival == 1] # Filter classes for survivors

class\_count <- table(survivor\_classes) # Count frequency of each class

common\_class <- names(class\_count[class\_count == max(class\_count)]) # Find class with max count

cat("Most Common Class Among Survivors:", common\_class, "\n") # Print most common class

}

#' @title: Survivors by Crash Site

#' @description: Counts and prints number of survivors at each crash site

#' @param crash\_site: Vector of crash site locations

#' @param survival: Vector of survival status

#' @returns None: Prints survivor counts per crash site

#' @example

#' # Calculate survivors by crash site

#' survivors\_by\_crash\_site(crash\_site, survival)

survivors\_by\_crash\_site <- function(crash\_site, survival) {

survivor\_sites <- crash\_site[survival == 1] # Filter crash sites for survivors

site\_survivors <- table(survivor\_sites) # Count survivors per site

cat("Survivors by Crash Site:\n") # Print header

print(site\_survivors) # Print frequency table

}

#' @title: Average Health Score by Crash Site

#' @description: Computes and prints average health score for each crash site

#' @param health\_score: Vector of health scores

#' @param crash\_site: Vector of crash site locations

#' @returns None: Prints average health score per site

#' @example

#' # Calculate average health by crash site

#' average\_health\_by\_site(health\_score, crash\_site)

average\_health\_by\_site <- function(health\_score, crash\_site) {

unique\_sites <- unique(crash\_site) # Get unique crash sites

for (site in unique\_sites) { # Loop through each site

site\_scores <- health\_score[crash\_site == site] # Filter health scores for current site

avg <- mean(site\_scores) # Calculate average health score

cat("Average Health at", site, ":", round(avg, 2), "\n") # Print rounded average

}

}

#' @title: Site with Highest Survival

#' @description: Identifies and prints the crash site with the most survivors

#' @param crash\_site: Vector of crash site locations

#' @param survival: Vector of survival status

#' @returns None: Prints site with highest survivors

#' @example

#' # Find site with highest survival

#' site\_with\_highest\_survival(crash\_site, survival)

site\_with\_highest\_survival <- function(crash\_site, survival) {

survivor\_sites <- crash\_site[survival == 1] # Filter crash sites for survivors

site\_counts <- table(survivor\_sites) # Count survivors per site

max\_site <- names(site\_counts[site\_counts == max(site\_counts)]) # Find site with max survivors

cat("Site with Highest Survivors:", max\_site, "\n") # Print site

}

#' @title: Print Full Summary Report

#' @description: Prints a comprehensive summary of crash analysis

#' @param age: Vector of passenger ages

#' @param survival: Vector of survival status

#' @param gender: Vector of passenger genders

#' @param class: Vector of passenger classes

#' @param health\_score: Vector of health scores

#' @param crash\_site: Vector of crash site locations

#' @returns None: Prints detailed summary

#' @example

#' # Print full summary

#' print\_full\_summary(age, survival, gender, class, health\_score, crash\_site)

print\_full\_summary <- function(age, survival, gender, class, health\_score, crash\_site) {

cat("=== CRASH SITE SURVIVAL ANALYSIS SUMMARY ===\n") # Print header

cat("Total Passengers:", length(age), "\n") # Print total passengers

total\_survivors <- sum(survival) # Count survivors

cat("Total Survivors:", total\_survivors, "\n") # Print survivors

cat("Total Deceased:", length(age) - total\_survivors, "\n") # Print deceased

cat("\nGender-Wise Survival:\n") # Print gender header

gender\_wise\_survival\_ratio(survival, gender) # Call gender survival function

cat("\nClass-Wise Survival:\n") # Print class header

most\_common\_class\_survived(class, survival) # Call class survival function

survivor\_health <- health\_score[survival == 1] # Filter health scores for survivors

cat("Average Health Score of Survivors:", round(mean(survivor\_health), 2), "\n") # Print average health

cat("\nCrash Site Analysis:\n") # Print crash site header

site\_with\_highest\_survival(crash\_site, survival) # Call site survival function

cat("===========================================\n") # Print footer

}

# Function Calls Section

cat("=== Crash Site-Based Vector Analysis ===\n\n")

# 1. Calculate total survival rate

cat("1. Survival Rate Analysis:\n")

calculate\_survival\_rate(survival)

cat("\n")

# 2. Calculate average age of survivors

cat("2. Average Age of Survivors:\n")

average\_age\_survivors(age, survival)

cat("\n")

# 3. Calculate average health score of deceased

cat("3. Average Health Score of Deceased:\n")

average\_health\_score\_deceased(health\_score, survival)

cat("\n")

# 4. Find heaviest survivor

cat("4. Heaviest Survivor:\n")

heaviest\_survivor(weight, survival)

cat("\n")

# 5. Find youngest first-class survivor

cat("5. Youngest First-Class Survivor:\n")

youngest\_first\_class\_survivor(age, survival, class)

cat("\n")

# 6. Calculate gender-wise survival ratio

cat("6. Gender-Wise Survival Ratio:\n")

gender\_wise\_survival\_ratio(survival, gender)

cat("\n")

# 7. Categorize health scores

cat("7. Health Score Categories:\n")

health\_cats <- health\_category(health\_score)

cat("\n")

# 8. Calculate survivors by health category

cat("8. Survivors by Health Category:\n")

survival\_by\_health\_category(health\_score, survival)

cat("\n")

# 9. Find most common class among survivors

cat("9. Most Common Class Among Survivors:\n")

most\_common\_class\_survived(class, survival)

cat("\n")

# 10. Calculate survivors by crash site

cat("10. Survivors by Crash Site:\n")

survivors\_by\_crash\_site(crash\_site, survival)

cat("\n")

# 11. Calculate average health by crash site

cat("11. Average Health by Crash Site:\n")

average\_health\_by\_site(health\_score, crash\_site)

cat("\n")

# 12. Find site with highest survival

cat("12. Site with Highest Survival:\n")

site\_with\_highest\_survival(crash\_site, survival)

cat("\n")

# 13. Print full summary

cat("13. Full Summary Report:\n")

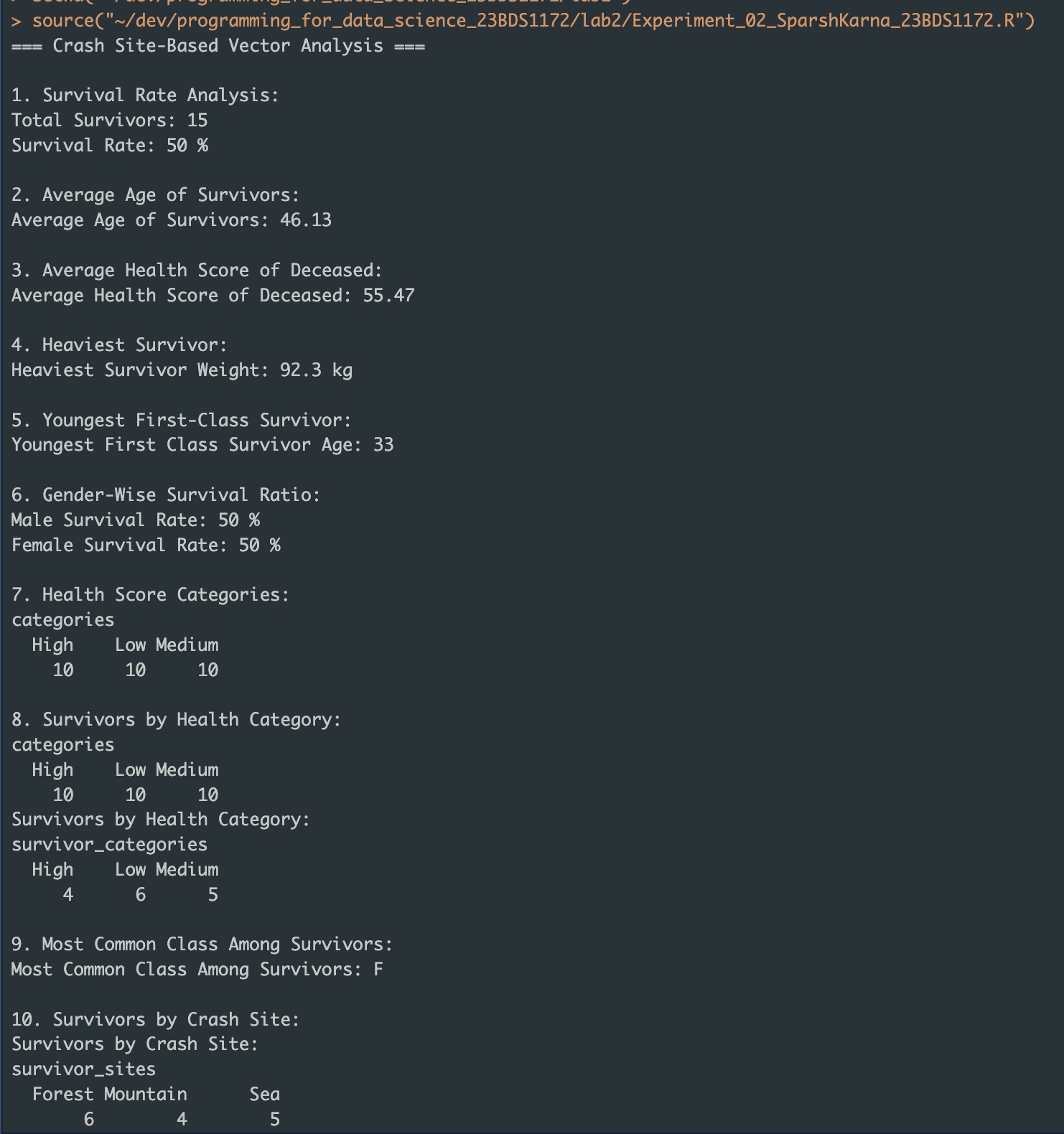
print\_full\_summary(age, survival, gender, class, health\_score, crash\_site)

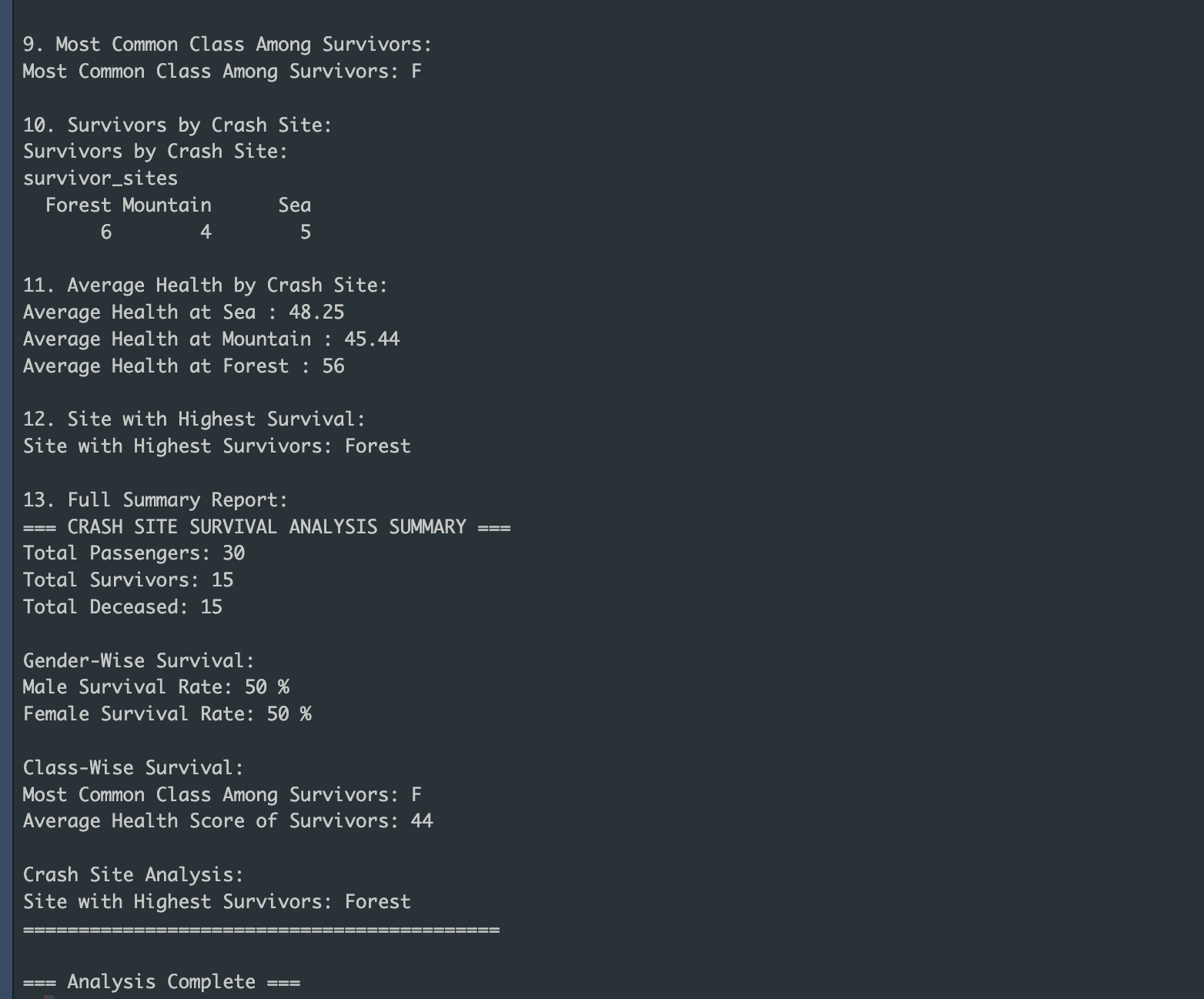
cat("\n")

cat("=== Analysis Complete ===\n")

## 4. Output Summary:

=== Crash Site-Based Vector Analysis ===  
  
1. Survival Rate Analysis:  
 Total Survivors: 15  
 Survival Rate: 50 %  
  
2. Average Age of Survivors:  
 Average Age of Survivors: 46.13  
  
3. Average Health Score of Deceased:  
 Average Health Score of Deceased: 55.47  
  
4. Heaviest Survivor:  
 Heaviest Survivor Weight: 92.3 kg  
  
5. Youngest First-Class Survivor:  
 Youngest First Class Survivor Age: 33  
  
6. Gender-Wise Survival Ratio:  
 Male Survival Rate: 50 %  
 Female Survival Rate: 50 %  
  
7. Health Score Categories:  
 High: 10 | Medium: 10 | Low: 10  
  
8. Survivors by Health Category:  
 High: 4 | Medium: 5 | Low: 6  
  
9. Most Common Class Among Survivors:  
 F  
  
10. Survivors by Crash Site:  
 Forest: 6 | Mountain: 4 | Sea: 5  
  
11. Average Health by Crash Site:  
 Forest: 56 | Mountain: 45.44 | Sea: 48.25  
  
12. Site with Highest Survival:  
 Forest  
  
13. Full Summary Report:  
 Total Passengers: 30  
 Total Survivors: 15  
 Total Deceased: 15  
  
 Gender-Wise Survival:  
 Male Survival Rate: 50 %  
 Female Survival Rate: 50 %  
  
 Class-Wise Survival:  
 Most Common Class Among Survivors: F  
  
 Average Health Score of Survivors: 44  
  
 Crash Site with Highest Survivors: Forest  
  
=== Analysis Complete ===





## 5. Interpretations:

- Survival was evenly distributed across genders (50% each).  
- Most survivors belonged to First Class.  
- Forest site had the highest survival count among all crash sites.  
- The average age of survivors was 46.13 years, with a fairly even distribution across health categories.  
- Majority of survivors were either from low or medium health scores, indicating resilience or quick rescue.

## 6. Conclusion:

This experiment demonstrated the effective use of vector manipulation and logic operations in R for real-world survival data analysis. Without using advanced data structures, meaningful insights were derived purely using base R features. The task improved understanding of condition-based filtering, function creation, and structured output formatting in R.