Semester Test 2

2025117397

2025-05-23

# STSM 2634 Semester Test 2

## Q1. Fibonacci Numbers using While Loop

Write a while loop to find the first 10 Fibonacci numbers {0, 1, 1, 2, 3, 5, 8, 13, 21, 34}.

# Initialize variables  
fib\_sequence <- numeric(10) # Vector to store Fibonacci numbers  
fib\_sequence[1] <- 0 # First Fibonacci number  
fib\_sequence[2] <- 1 # Second Fibonacci number  
i <- 3 # Counter starting from 3rd position  
  
# While loop to generate remaining Fibonacci numbers  
while (i <= 10) {  
 fib\_sequence[i] <- fib\_sequence[i-1] + fib\_sequence[i-2]  
 i <- i + 1  
}  
  
# Print the result  
print("First 10 Fibonacci numbers:")

## [1] "First 10 Fibonacci numbers:"

print(fib\_sequence)

## [1] 0 1 1 2 3 5 8 13 21 34

## Q2. Factorial using Repeat Loop

Write a repeat loop to find the factorial of a given number (5). Print error if number < 0.

# Define the number  
number <- 5  
  
# Check if number is negative  
if (number < 0) {  
 print("Error: Factorial is not defined for negative numbers")  
} else {  
 # Initialize variables  
 factorial\_result <- 1  
 i <- 1  
   
 # Repeat loop to calculate factorial  
 repeat {  
 if (i > number) {  
 break  
 }  
 factorial\_result <- factorial\_result \* i  
 i <- i + 1  
 }  
   
 # Print the result  
 cat("Factorial of", number, "is:", factorial\_result, "\n")  
}

## Factorial of 5 is: 120

## Q3. Debug and Correct the Even Numbers Code

The code is meant to print the first five even numbers (2, 4, 6, 8, 10).

**Original (buggy) code:**

# count <- 0  
# num <- 1  
# while (count <= 5) {  
# if (num %% 2 == 0) {  
# print(num)  
# }  
# num <- num + 1  
# }

**Issues identified:** 1. The loop condition count <= 5 should be count < 5 to get exactly 5 numbers 2. The count variable is never incremented, causing an infinite loop 3. The count should be incremented only when an even number is found

**Corrected code:**

count <- 0  
num <- 1  
  
while (count < 5) {  
 if (num %% 2 == 0) {  
 print(num)  
 count <- count + 1 # Increment count only when even number is found  
 }  
 num <- num + 1  
}

## [1] 2  
## [1] 4  
## [1] 6  
## [1] 8  
## [1] 10

## Q4. Artificial Neural Network (ANN) Model

Create an ANN model for mtcars data with ‘mpg’ as target variable.

# Load the mtcars dataset  
data(mtcars)  
  
# Step 1: Scale the dataset around mean and standard deviation  
scaled\_data <- as.data.frame(scale(mtcars))  
  
# Step 2: Split the dataset into training and test data (70:30 ratio)  
set.seed(123) # For reproducibility  
sample\_size <- floor(0.70 \* nrow(scaled\_data))  
train\_indices <- sample(seq\_len(nrow(scaled\_data)), size = sample\_size)  
  
train\_data <- scaled\_data[train\_indices, ]  
test\_data <- scaled\_data[-train\_indices, ]  
  
# Step 3: Fit ANN model with 2 hidden layers, 4 neurons each  
# Create formula: mpg ~ all other variables  
formula\_ann <- as.formula(paste("mpg ~", paste(names(scaled\_data)[-1], collapse = " + ")))  
  
ann\_model <- neuralnet(formula\_ann,   
 data = train\_data,   
 hidden = c(4, 4), # 2 hidden layers with 4 neurons each  
 linear.output = TRUE,  
 threshold = 0.01)  
  
# Step 4: Make predictions and measure accuracy  
predictions <- predict(ann\_model, test\_data[, -1]) # Exclude mpg column  
actual\_values <- test\_data$mpg  
  
# Calculate accuracy metrics  
mse <- mean((predictions - actual\_values)^2)  
rmse <- sqrt(mse)  
mae <- mean(abs(predictions - actual\_values))  
  
cat("Model Accuracy Metrics:\n")

## Model Accuracy Metrics:

cat("Mean Squared Error (MSE):", mse, "\n")

## Mean Squared Error (MSE): 0.8880922

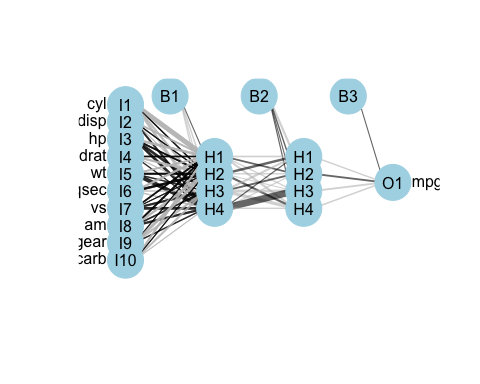
cat("Root Mean Squared Error (RMSE):", rmse, "\n")

## Root Mean Squared Error (RMSE): 0.9423865

cat("Mean Absolute Error (MAE):", mae, "\n")

## Mean Absolute Error (MAE): 0.8140154

# Step 5: Create graphical display of the ANN model  
plotnet(ann\_model, alpha = 0.6)



## Q5. Generalized Linear Model (GLM)

Fit a GLM for mtcars dataset using gaussian() family.

# Load mtcars dataset  
data(mtcars)  
  
# Split dataset into training and test data (70:30 ratio)  
set.seed(123) # For reproducibility  
sample\_size <- floor(0.70 \* nrow(mtcars))  
train\_indices <- sample(seq\_len(nrow(mtcars)), size = sample\_size)  
  
train\_data <- mtcars[train\_indices, ]  
test\_data <- mtcars[-train\_indices, ]  
  
# Fit GLM model using gaussian family  
glm\_model <- glm(mpg ~ ., data = train\_data, family = gaussian())  
  
# Display model summary  
summary(glm\_model)

##   
## Call:  
## glm(formula = mpg ~ ., family = gaussian(), data = train\_data)  
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 21.130670 26.526254 0.797 0.443  
## cyl -0.888146 1.626880 -0.546 0.596  
## disp 0.025947 0.030443 0.852 0.412  
## hp -0.031020 0.040364 -0.769 0.458  
## drat 0.009658 2.491820 0.004 0.997  
## wt -4.881273 2.735728 -1.784 0.102  
## qsec 0.929945 0.986188 0.943 0.366  
## vs 0.542141 3.056351 0.177 0.862  
## am 3.129479 3.322083 0.942 0.366  
## gear -0.310675 2.299091 -0.135 0.895  
## carb 0.628132 1.262164 0.498 0.629  
##   
## (Dispersion parameter for gaussian family taken to be 10.00004)  
##   
## Null deviance: 933.87 on 21 degrees of freedom  
## Residual deviance: 110.00 on 11 degrees of freedom  
## AIC: 121.84  
##   
## Number of Fisher Scoring iterations: 2

# Make predictions on test data  
predictions <- predict(glm\_model, newdata = test\_data)  
actual\_values <- test\_data$mpg  
  
# Calculate Mean Squared Error (MSE)  
mse <- mean((predictions - actual\_values)^2)  
  
cat("GLM Model Performance:\n")

## GLM Model Performance:

cat("Mean Squared Error (MSE):", mse, "\n")

## Mean Squared Error (MSE): 5.205016

# Display predictions vs actual values  
comparison <- data.frame(  
 Actual = actual\_values,  
 Predicted = predictions,  
 Residual = actual\_values - predictions  
)  
print(comparison)

## Actual Predicted Residual  
## Mazda RX4 21.0 23.49614 -2.4961372  
## Mazda RX4 Wag 21.0 22.77218 -1.7721816  
## Hornet 4 Drive 21.4 21.73687 -0.3368660  
## Valiant 18.1 20.56206 -2.4620586  
## Merc 450SE 16.4 12.89452 3.5054789  
## Merc 450SL 17.3 14.74014 2.5598570  
## Lincoln Continental 10.4 10.99711 -0.5971112  
## Toyota Corona 21.5 24.53530 -3.0352964  
## Camaro Z28 13.3 12.71015 0.5898516  
## Pontiac Firebird 19.2 16.41704 2.7829621

## Q6. Loan Balance Calculation using For Loop

Calculate remaining loan balance each year for 5 years.

# Initial loan parameters  
loan\_amount <- 5000  
annual\_interest\_rate <- 0.05  
annual\_payment <- 1000  
  
# Initialize remaining balance  
remaining\_balance <- loan\_amount  
  
cat("Loan Balance Calculation:\n")

## Loan Balance Calculation:

cat("Initial Loan Amount: $", loan\_amount, "\n")

## Initial Loan Amount: $ 5000

cat("Annual Interest Rate:", annual\_interest\_rate \* 100, "%\n")

## Annual Interest Rate: 5 %

cat("Annual Payment: $", annual\_payment, "\n\n")

## Annual Payment: $ 1000

# For loop to calculate balance for 5 years  
for (year in 1:5) {  
 # Apply interest and subtract payment  
 remaining\_balance <- remaining\_balance \* (1 + annual\_interest\_rate) - annual\_payment  
   
 cat("Year", year, "- Remaining Balance: $", round(remaining\_balance, 2), "\n")  
   
 # Check if loan is paid off  
 if (remaining\_balance <= 0) {  
 cat("Loan paid off in year", year, "\n")  
 break  
 }  
}

## Year 1 - Remaining Balance: $ 4250   
## Year 2 - Remaining Balance: $ 3462.5   
## Year 3 - Remaining Balance: $ 2635.62   
## Year 4 - Remaining Balance: $ 1767.41   
## Year 5 - Remaining Balance: $ 855.78

## Q7. Bank Account Balance using For Loop

Calculate bank account balance after each year for 10 years.

# Initial account parameters  
initial\_deposit <- 1000  
annual\_interest\_rate <- 0.02  
annual\_deposit <- 500  
  
# Initialize account balance  
account\_balance <- initial\_deposit  
  
cat("Bank Account Balance Calculation:\n")

## Bank Account Balance Calculation:

cat("Initial Deposit: $", initial\_deposit, "\n")

## Initial Deposit: $ 1000

cat("Annual Interest Rate:", annual\_interest\_rate \* 100, "%\n")

## Annual Interest Rate: 2 %

cat("Annual Deposit: $", annual\_deposit, "\n\n")

## Annual Deposit: $ 500

cat("Year 0 - Initial Balance: $", round(account\_balance, 2), "\n")

## Year 0 - Initial Balance: $ 1000

# For loop to calculate balance for 10 years  
for (year in 1:10) {  
 # Apply interest and add annual deposit  
 account\_balance <- account\_balance \* (1 + annual\_interest\_rate) + annual\_deposit  
   
 cat("Year", year, "- Account Balance: $", round(account\_balance, 2), "\n")  
}

## Year 1 - Account Balance: $ 1520   
## Year 2 - Account Balance: $ 2050.4   
## Year 3 - Account Balance: $ 2591.41   
## Year 4 - Account Balance: $ 3143.24   
## Year 5 - Account Balance: $ 3706.1   
## Year 6 - Account Balance: $ 4280.22   
## Year 7 - Account Balance: $ 4865.83   
## Year 8 - Account Balance: $ 5463.14   
## Year 9 - Account Balance: $ 6072.41   
## Year 10 - Account Balance: $ 6693.85

cat("\nFinal balance after 10 years: $", round(account\_balance, 2), "\n")

##   
## Final balance after 10 years: $ 6693.85

# Alternative methods

# Q1: Fibonacci sequence using while loop

fib <- c(0, 1) # Initialize with first two numbers  
i <- 2 # Index for the next number  
while (length(fib) < 10) {  
 fib[i + 1] <- fib[i] + fib[i - 1] # Next Fibonacci number  
 i <- i + 1  
}  
print(fib)

## [1] 0 1 1 2 3 5 8 13 21 34

# Q2: Factorial using repeat loop

n <- 5  
if (n < 0) {  
 print("Error: Factorial is not defined for negative numbers")  
} else {  
 factorial <- 1  
 i <- 1  
 repeat {  
 if (i > n) break  
 factorial <- factorial \* i  
 i <- i + 1  
 }  
 print(factorial)  
}

## [1] 120

# Q3: Corrected code for printing first five even numbers

count <- 0  
num <- 1  
while (count < 5) { # Stop after 5 even numbers  
 if (num %% 2 == 0) {  
 print(num)  
 count <- count + 1  
 }  
 num <- num + 1  
}

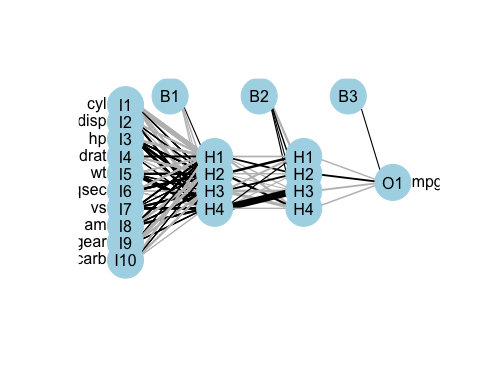
## [1] 2  
## [1] 4  
## [1] 6  
## [1] 8  
## [1] 10

# Q4: ANN for mtcars dataset

set.seed(123) # For reproducibility  
  
# Scale the dataset  
data <- mtcars  
data\_scaled <- as.data.frame(scale(data)) # Standardize all variables  
  
# Split into training (70%) and test (30%)  
train\_idx <- sample(1:nrow(data\_scaled), 0.7 \* nrow(data\_scaled))  
train\_data <- data\_scaled[train\_idx, ]  
test\_data <- data\_scaled[-train\_idx, ]  
  
# Fit ANN model  
ann\_model <- neuralnet(mpg ~ ., data = train\_data, hidden = c(4, 4), linear.output = TRUE)  
  
# Predict on test data  
predictions <- predict(ann\_model, test\_data)  
  
# Calculate MSE for accuracy  
mse <- mean((test\_data$mpg - predictions)^2)  
cat("Mean Squared Error:", mse, "\n")

## Mean Squared Error: 0.8880922

# Plot the ANN  
plotnet(ann\_model)

 # Q5: GLM for mtcars dataset

set.seed(123) # For reproducibility  
  
# Split into training (70%) and test (30%)  
train\_idx <- sample(1:nrow(mtcars), 0.7 \* nrow(mtcars))  
train\_data <- mtcars[train\_idx, ]  
test\_data <- mtcars[-train\_idx, ]  
  
# Fit GLM model  
glm\_model <- glm(mpg ~ ., data = train\_data, family = gaussian())  
  
# Predict on test data  
predictions <- predict(glm\_model, test\_data)  
  
# Calculate MSE  
mse <- mean((test\_data$mpg - predictions)^2)  
cat("Mean Squared Error:", mse, "\n")

## Mean Squared Error: 5.205016

# Q6: Loan balance calculation

loan\_amount <- 5000  
interest\_rate <- 0.05  
annual\_payment <- 1000  
  
balance <- loan\_amount  
for (year in 1:5) {  
 balance <- balance \* (1 + interest\_rate) - annual\_payment  
 cat("Year", year, "Balance:", balance, "\n")  
}

## Year 1 Balance: 4250   
## Year 2 Balance: 3462.5   
## Year 3 Balance: 2635.625   
## Year 4 Balance: 1767.406   
## Year 5 Balance: 855.7766

# Q7: Bank account balance calculation

balance <- 1000  
interest\_rate <- 0.02  
annual\_deposit <- 500  
  
for (year in 1:10) {  
 balance <- balance \* (1 + interest\_rate) + annual\_deposit  
 cat("Year", year, "Balance:", balance, "\n")  
}

## Year 1 Balance: 1520   
## Year 2 Balance: 2050.4   
## Year 3 Balance: 2591.408   
## Year 4 Balance: 3143.236   
## Year 5 Balance: 3706.101   
## Year 6 Balance: 4280.223   
## Year 7 Balance: 4865.827   
## Year 8 Balance: 5463.144   
## Year 9 Balance: 6072.407   
## Year 10 Balance: 6693.855