Consulting Project Report

Xiaohan Shi, Zihao Zhang, Suheng Yao

2024-11-16

```
# Read in the data
df_score <- read.csv("diversity_score.csv")
df_sex <- read.csv("diversity_score_withsex.csv")
df_final <- read.csv("final_data.csv")

df_final_subset <- df_final[, c("Name", "Follow.Number", "Minutes.awake")]
df_final_subset <- df_final_subset %>%
  rename(
    FollowNumber = Follow.Number
  )
df_final_subset$Name <- tolower(df_final_subset$Name)
df_sex <- merge(df_sex, df_final_subset, by = c("Name", "FollowNumber"), all.x = TRUE)
```

Introduction

Our group's project is on the positional behaviors of orangutans. Orangutans are primates that live in the rain forests of Southeast Asia and are known for their red fur and high intelligence. They are also the largest arboreal mammals living in the world. They are found mainly in Borneo and Sumatra and are listed as an endangered species due to habitat loss and threats from illegal hunting. The main objective of this consulting project is to help the client analyze the positional behaviors of orangutans over developmental stages to understand how these behaviors evolve with age. The client used a measurement called Shannor Weaver Index to calculate the diversity score based on the positional behavior and try to find its relationship with age. How Shannor Weaver Index is calculated will be covered more in the method part of the report.

The original data that the client gave us contains 237 individual orangutans that had been followed for 30 years at different periods. Each orangutan was observed for a period of time at different times, and the timing of the behavior at each location was recorded. In total, there are 77 unique positional behaviors recorded for each orangutan. These behaviors are created by the combination of three behavior groups: activity type, body position, and tree position.

In this report, we will mainly talk about the basic EDA to analyze the data, the methods used to assess the relationship, including the models used, and finally the conclusion based on the results.

Method

How does Shannon Weaver Index calculate?

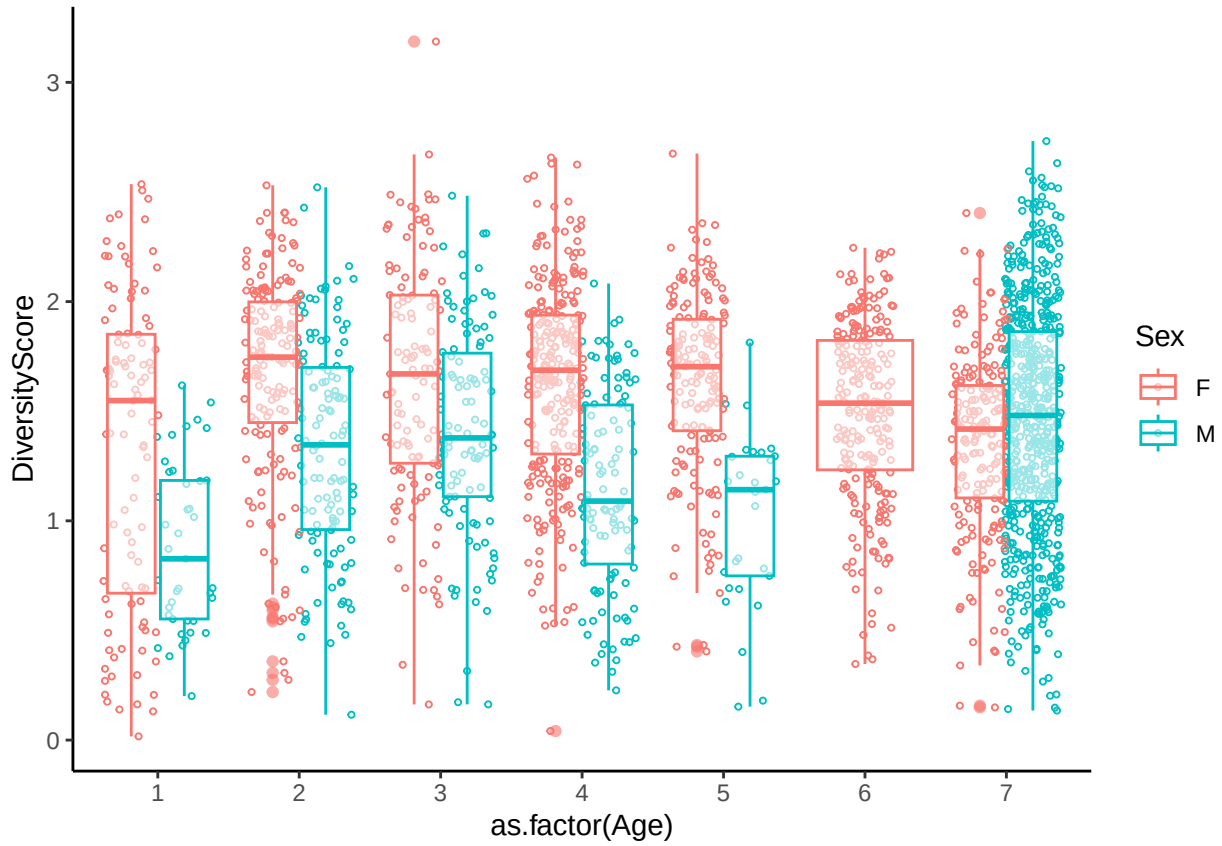
The formula is shown below: $H' = -\sum_{i=1}^{S} p_i \ln(p_i)$ In this formula, S represents the number of unique categories in the data, p_i is the proportion, which in the client's data refers to the proportion of recorded time of each distinct behaviors of a specific orangutan in relation to this orangutan's total minutes of awake.

Data Cleaning on Diversity Score with Sex Dataset

```
df_sex$Sex <- gsub("M?", "M", df_sex$Sex, fixed = TRUE)
df_sex <- df_sex[!df_sex$Sex %in% c("?", "Q"), ]
unique(df_sex$Sex)
```

```
## [1] "M" "F"
```

```
library(ggplot2)
ggplot() +
  geom_point(data=df_sex, aes(x=as.factor(Age), y=DiversityScore, fill=Sex, color=Sex), size = 0.75, sha
  geom_boxplot(data=df_sex, aes(x=as.factor(Age), y=DiversityScore, color=Sex), alpha = 0.6) +
  theme_classic()
```



Client Models

Following client suggestions, we would like to use linear mixed effect models and build upon client's models and see if we improve the client's models. Initial client models are listed below:

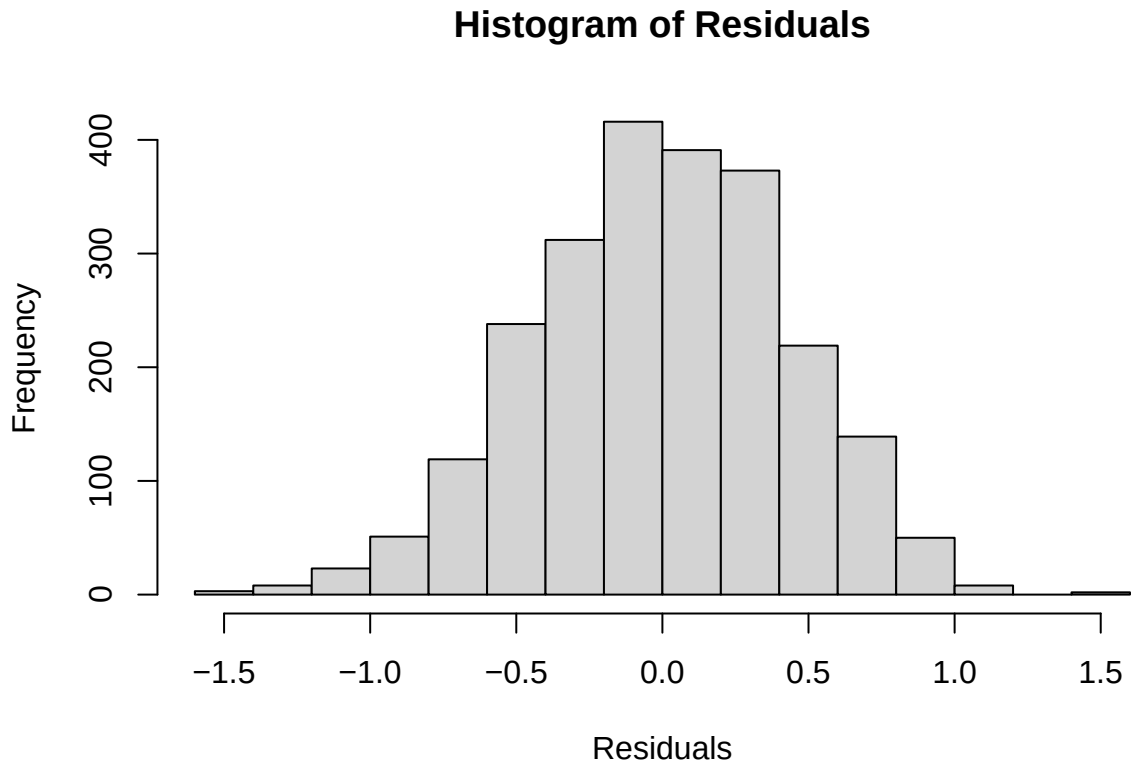
```
model.1 <- glmmTMB(data=df_score,
                   DiversityScore ~  (1|Name) + (1|FollowNumber),
                   family=gaussian(link="log"))
```

```r
model.2 <- glmmTMB(data=df_score,
                   DiversityScore ~ Age + (1|Name)+ (1|FollowNumber),
                   family=gaussian(link="log"))

model.3 <- glmmTMB(data=df_score,
                   DiversityScore ~ Age + I(Age^2) +
                     (1|Name) + (1|FollowNumber),
                   family=gaussian(link="log"))
```

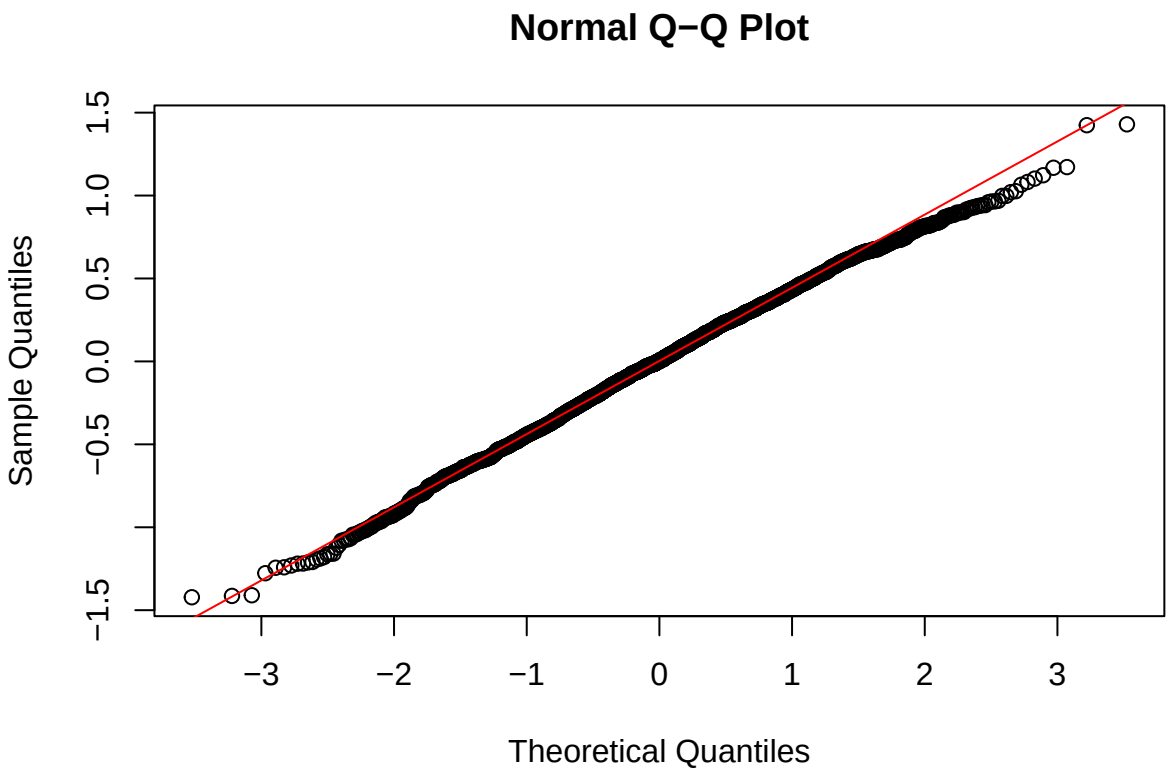
Check Client's Model Assumptions

We first need to check all the linear assumptions are met:

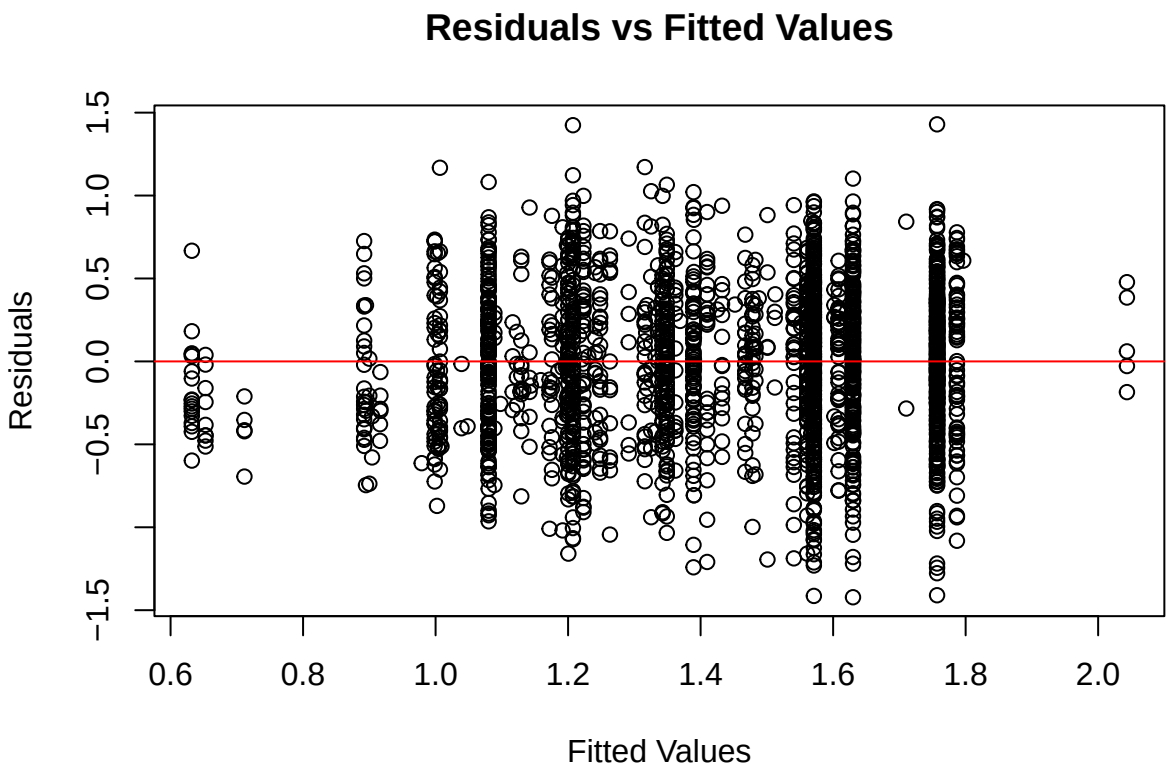
```r
residuals_model <- residuals(model.1, type = "pearson")

# Plot histogram of residuals
hist(residuals_model, main = "Histogram of Residuals", xlab = "Residuals")
```



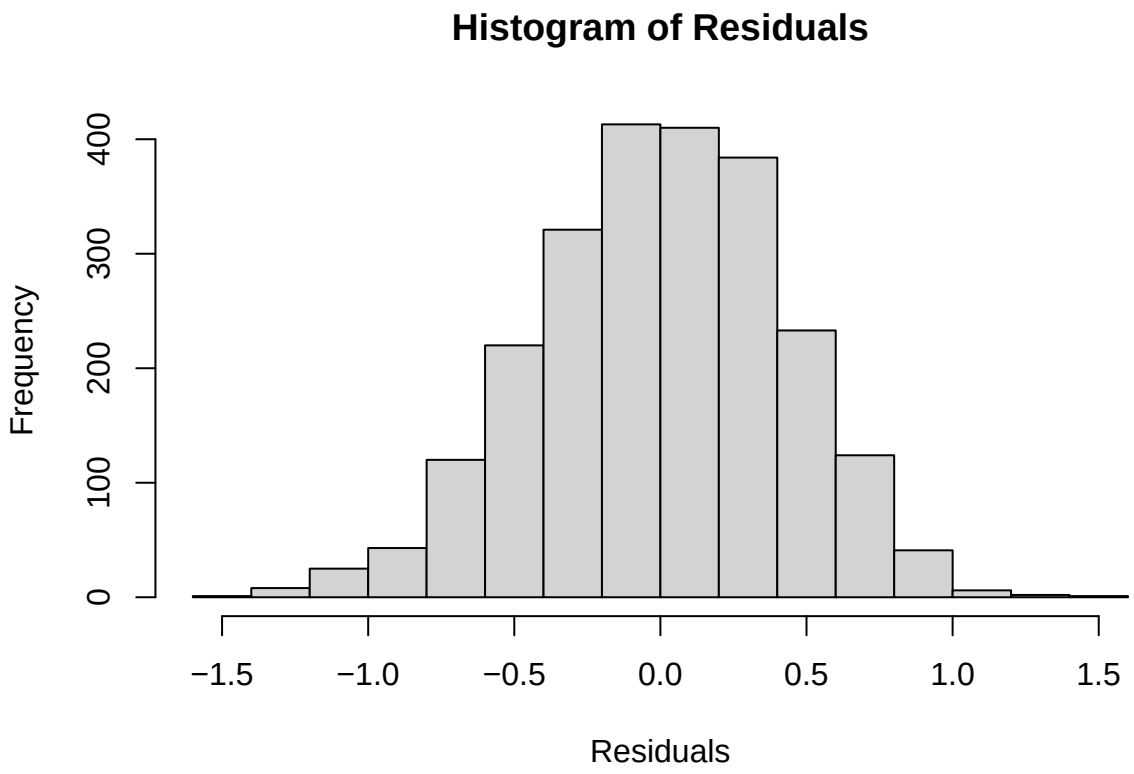

```r
# Q-Q plot to check normality
qqnorm(residuals_model)
qqline(residuals_model, col = "red")
```



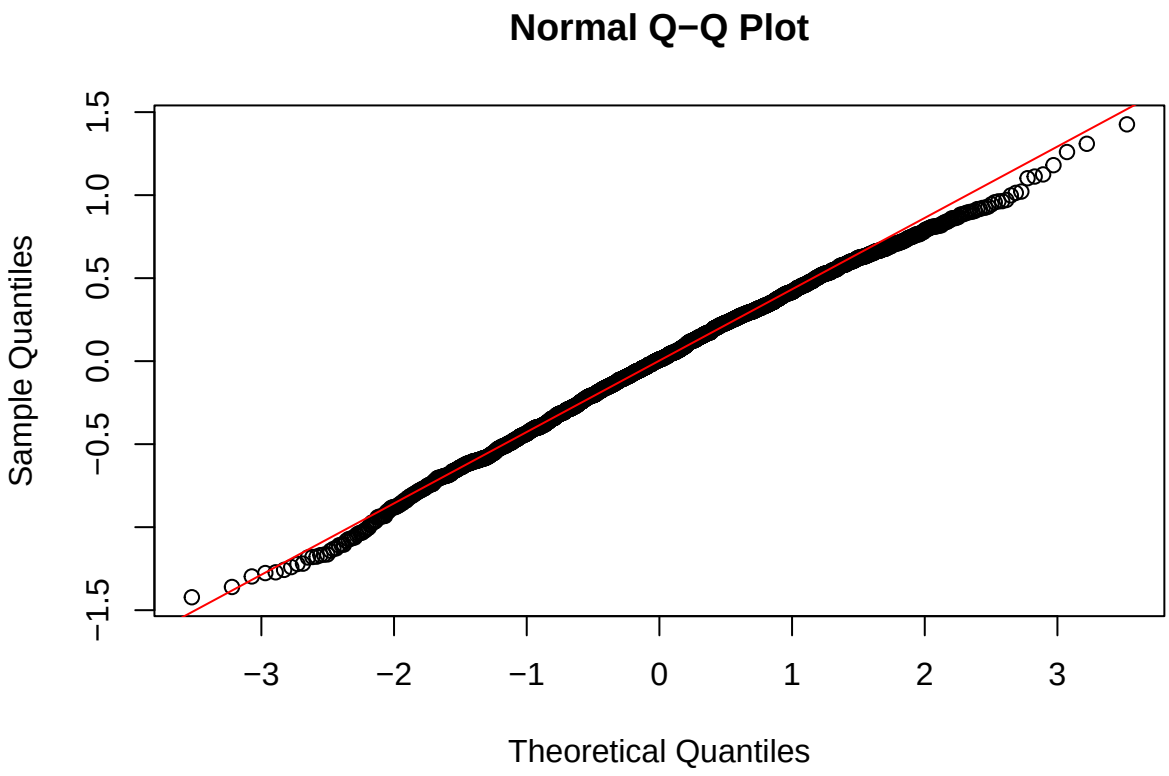

```r
fitted_values <- predict(model.1, type = "response")

# Plot residuals vs fitted values
plot(fitted_values, residuals_model, main = "Residuals vs Fitted Values",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



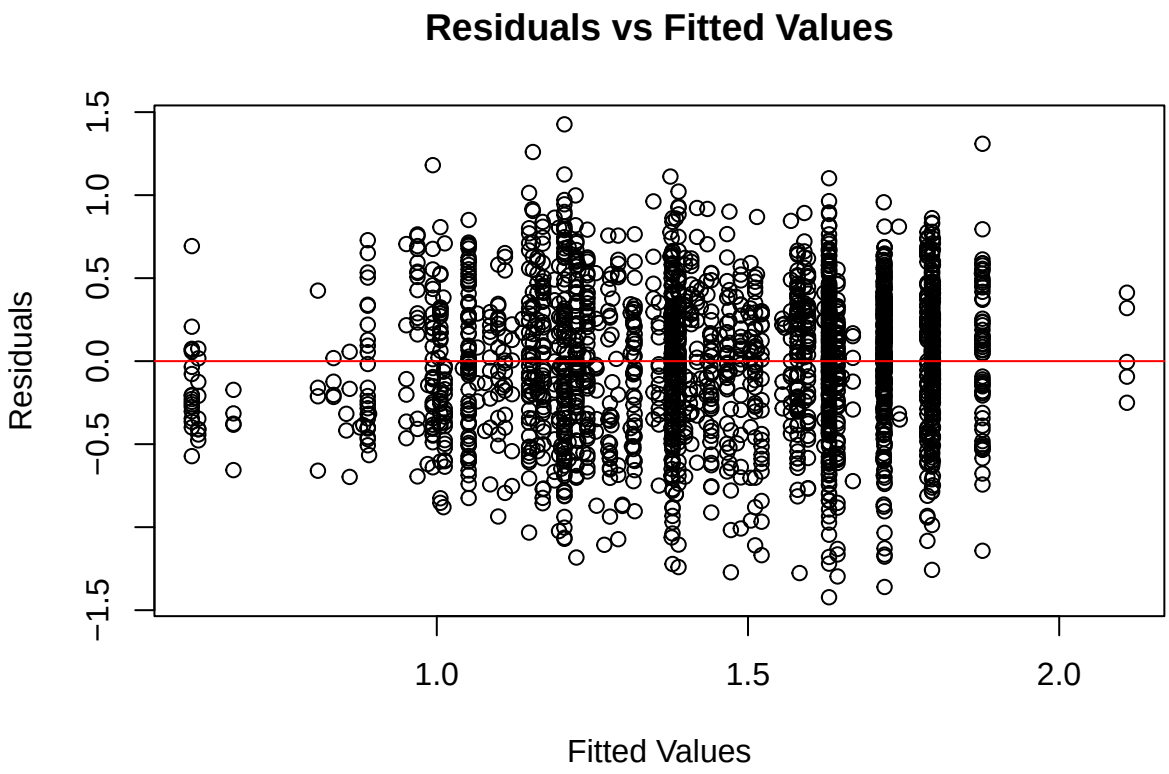

```r
residuals_model <- residuals(model.2, type = "pearson")

# Plot histogram of residuals
hist(residuals_model, main = "Histogram of Residuals", xlab = "Residuals")
```



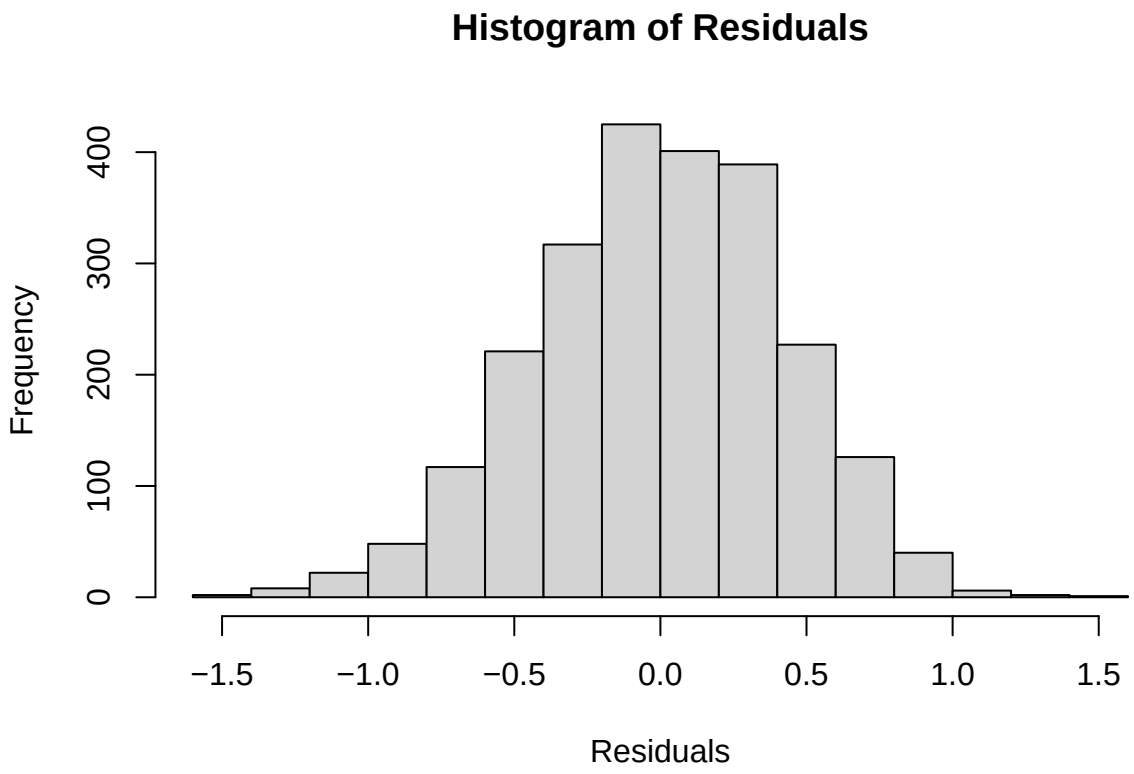

```r
# Q-Q plot to check normality
qqnorm(residuals_model)
qqline(residuals_model, col = "red")
```



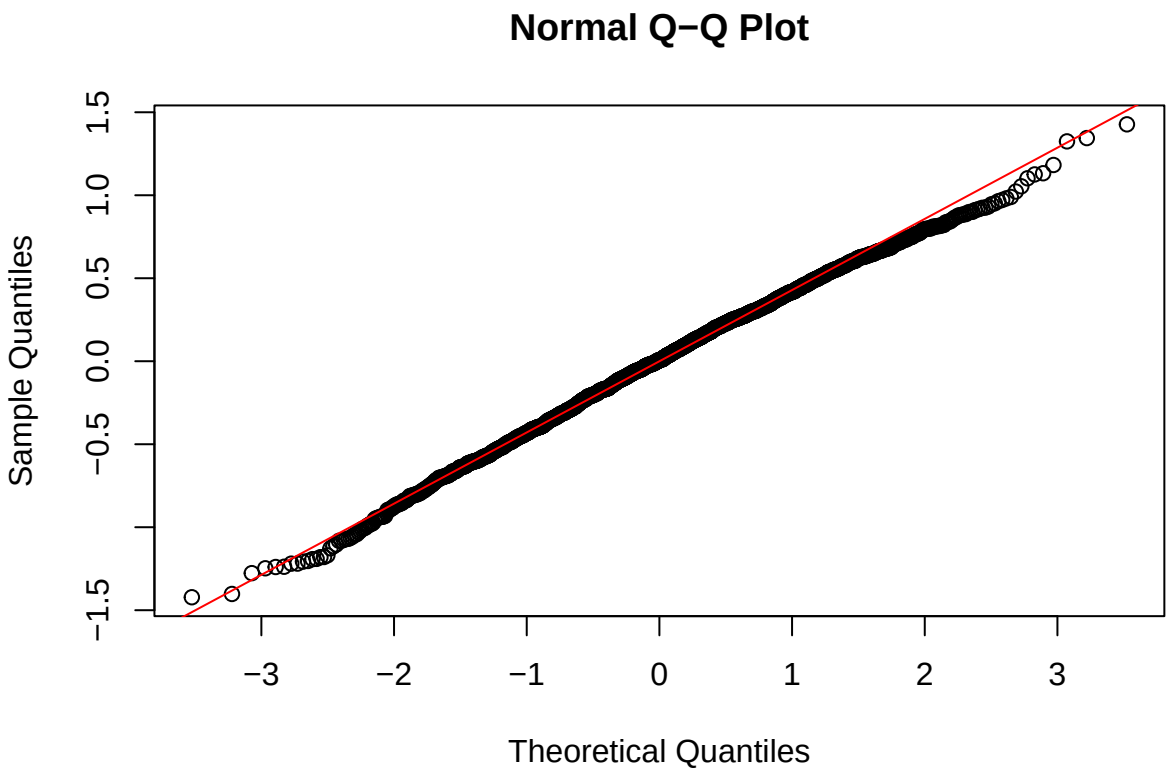

```r
fitted_values <- predict(model.2, type = "response")

# Plot residuals vs fitted values
plot(fitted_values, residuals_model, main = "Residuals vs Fitted Values",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



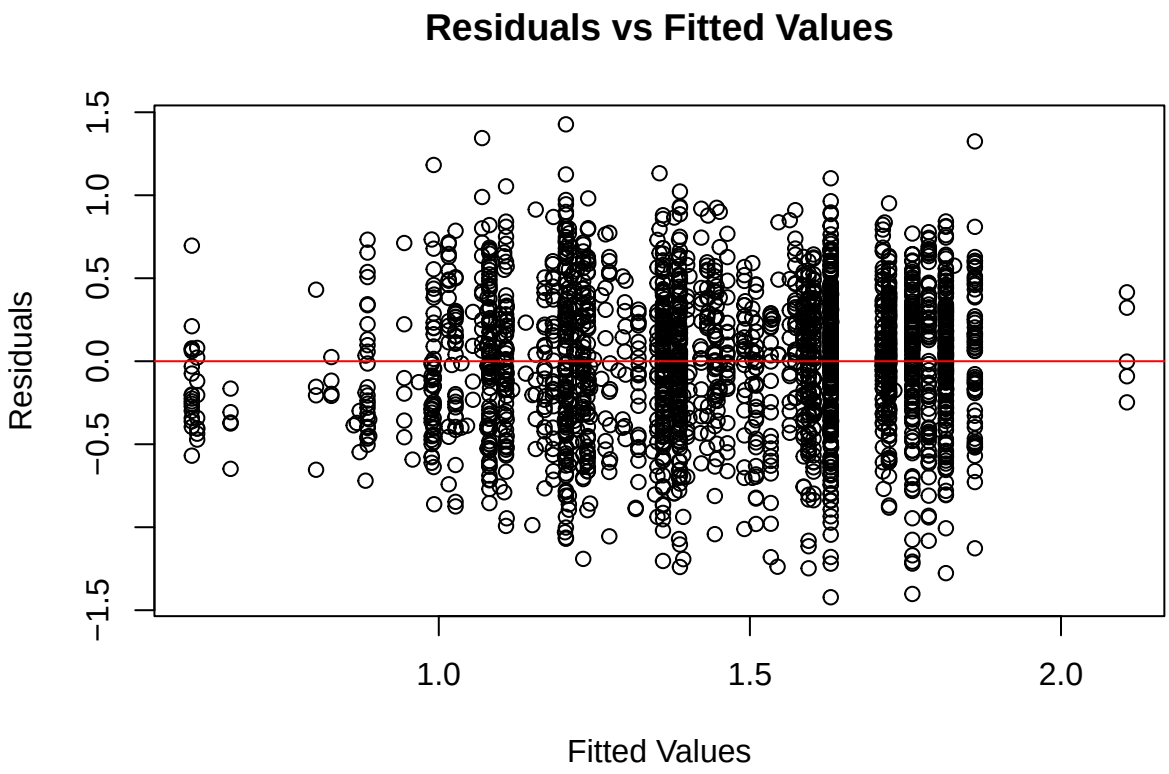

```r
residuals_model <- residuals(model.3, type = "pearson")

# Plot histogram of residuals
hist(residuals_model, main = "Histogram of Residuals", xlab = "Residuals")
```




```r
# Q-Q plot to check normality
qqnorm(residuals_model)
qqline(residuals_model, col = "red")
```




```r
fitted_values <- predict(model.3, type = "response")

# Plot residuals vs fitted values
plot(fitted_values, residuals_model, main = "Residuals vs Fitted Values",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



Based on the assumptions check above, all three models have normal distributed residuals, no-heavy tails in qq plot and random distribution in residual vs fitted plot, indicating that all the models satisfy linear assumptions.

Build Upon Client's Model

Here we propose three new models building upon the client models:

```
df_sex$Age <- as.factor(df_sex$Age)
df_sex <- df_sex %>%
  filter(!is.na(Minutes.awake))

newmodel.1 <- glmmTMB(data=df_sex, DiversityScore ~ Age + (1|Name),
                  family = gaussian("log"))
summary(newmodel.1)
```

```
##  Family: gaussian  ( log )
## Formula:          DiversityScore ~ Age + (1 | Name)
## Data: df_sex
##
##      AIC      BIC   logLik deviance df.resid
##   2660.4   2711.8  -1321.2   2642.4     2225
##
## Random effects:
##
```

```
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  Name     (Intercept) 0.08727  0.2954
##  Residual             0.17876  0.4228
## Number of obs: 2234, groups:  Name, 71
##
## Dispersion estimate for gaussian family (sigma^2): 0.179
##
## Conditional model:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.258472   0.051024   5.066 4.07e-07 ***
## Age2         0.077519   0.034781   2.229 0.025830 *
## Age3         0.110632   0.044576   2.482 0.013069 *
## Age4        -0.001466   0.044610  -0.033 0.973779
## Age5        -0.043123   0.046467  -0.928 0.353384
## Age6        -0.148249   0.043541  -3.405 0.000662 ***
## Age7        -0.197449   0.039128  -5.046 4.51e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
newmodel.2 <- glmmTMB(data=df_sex,
                 DiversityScore ~ Age + Sex + (1|Name),
                  family = gaussian("log"))
summary(newmodel.2)
```

```
##  Family: gaussian  ( log )
## Formula:          DiversityScore ~ Age + Sex + (1 | Name)
## Data: df_sex
##
##      AIC      BIC   logLik deviance df.resid
##   2661.0   2718.1  -1320.5   2641.0     2224
##
## Random effects:
##
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  Name     (Intercept) 0.08594  0.2931
##  Residual             0.17872  0.4228
## Number of obs: 2234, groups:  Name, 71
##
## Dispersion estimate for gaussian family (sigma^2): 0.179
##
## Conditional model:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.206606   0.066868   3.090 0.002003 **
## Age2         0.076450   0.034743   2.200 0.027774 *
## Age3         0.114238   0.044661   2.558 0.010532 *
## Age4         0.002154   0.044707   0.048 0.961567
## Age5        -0.039269   0.046578  -0.843 0.399182
## Age6        -0.145574   0.043584  -3.340 0.000838 ***
## Age7        -0.200867   0.039160  -5.129 2.91e-07 ***
## SexM         0.098890   0.082280   1.202 0.229414
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
newmodel.3 <- glmmTMB(data = df_sex,
                  DiversityScore ~ Age + Sex + Minutes.awake + (1 | Name),
                  family = gaussian("log"))
```

```
## Warning in (function (start, objective, gradient = NULL, hessian = NULL, :
## NA/NaN function evaluation
## Warning in (function (start, objective, gradient = NULL, hessian = NULL, :
## NA/NaN function evaluation
```

```r
summary(newmodel.3)
```

```
##  Family: gaussian  ( log )
## Formula:          DiversityScore ~ Age + Sex + Minutes.awake + (1 | Name)
## Data: df_sex
##
##      AIC      BIC   logLik deviance df.resid
##   2651.6   2714.4  -1314.8   2629.6     2223
##
## Random effects:
##
## Conditional model:
##  Groups   Name        Variance Std.Dev.
##  Name     (Intercept) 0.08584  0.2930
##  Residual             0.17780  0.4217
## Number of obs: 2234, groups:  Name, 71
##
## Dispersion estimate for gaussian family (sigma^2): 0.178
##
## Conditional model:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    5.508e-02  8.107e-02   0.679 0.496850
## Age2           8.349e-02  3.464e-02   2.410 0.015942 *
## Age3           1.168e-01  4.454e-02   2.623 0.008725 **
## Age4           2.910e-03  4.456e-02   0.065 0.947920
## Age5          -4.386e-02  4.645e-02  -0.944 0.345042
## Age6          -1.504e-01  4.345e-02  -3.462 0.000537 ***
## Age7          -2.030e-01  3.904e-02  -5.200 1.99e-07 ***
## SexM           1.002e-01  8.220e-02   1.219 0.222848
## Minutes.awake  2.371e-04  7.152e-05   3.316 0.000913 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(newmodel.1, newmodel.2)
```

```
## Data: df_sex
## Models:
## newmodel.1: DiversityScore ~ Age + (1 | Name), zi=~0, disp=~1
## newmodel.2: DiversityScore ~ Age + Sex + (1 | Name), zi=~0, disp=~1
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## newmodel.1  9 2660.4 2711.8 -1321.2   2642.4
## newmodel.2 10 2661.0 2718.1 -1320.5   2641.0 1.4486      1     0.2288
```

```r
anova(newmodel.2, newmodel.3)
```

```
## Data: df_sex
## Models:
## newmodel.2: DiversityScore ~ Age + Sex + (1 | Name), zi=~0, disp=~1
## newmodel.3: DiversityScore ~ Age + Sex + Minutes.awake + (1 | Name), zi=~0, disp=~1
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## newmodel.2 10 2661.0 2718.1 -1320.5   2641.0
## newmodel.3 11 2651.6 2714.4 -1314.8   2629.6 11.365      1  0.0007486 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

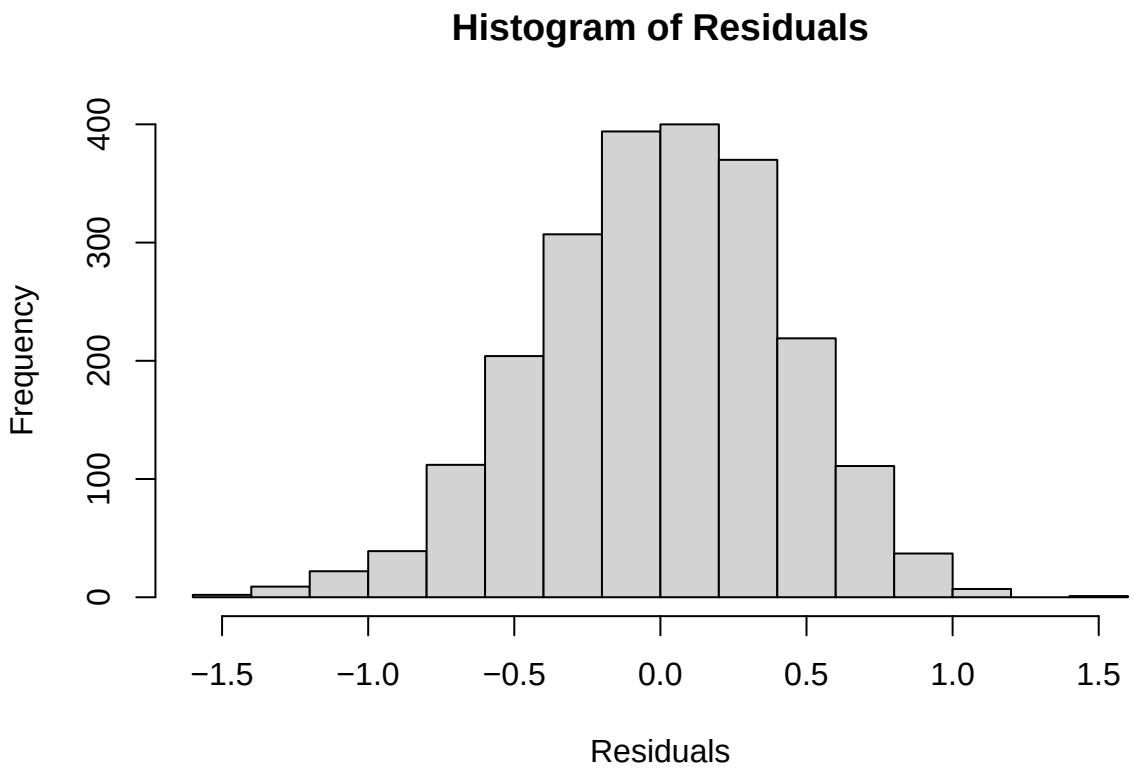
```r
anova(newmodel.1, newmodel.3)
```

```
## Data: df_sex
## Models:
## newmodel.1: DiversityScore ~ Age + (1 | Name), zi=~0, disp=~1
## newmodel.3: DiversityScore ~ Age + Sex + Minutes.awake + (1 | Name), zi=~0, disp=~1
##            Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## newmodel.1  9 2660.4 2711.8 -1321.2   2642.4
## newmodel.3 11 2651.6 2714.4 -1314.8   2629.6 12.813      2   0.001651 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

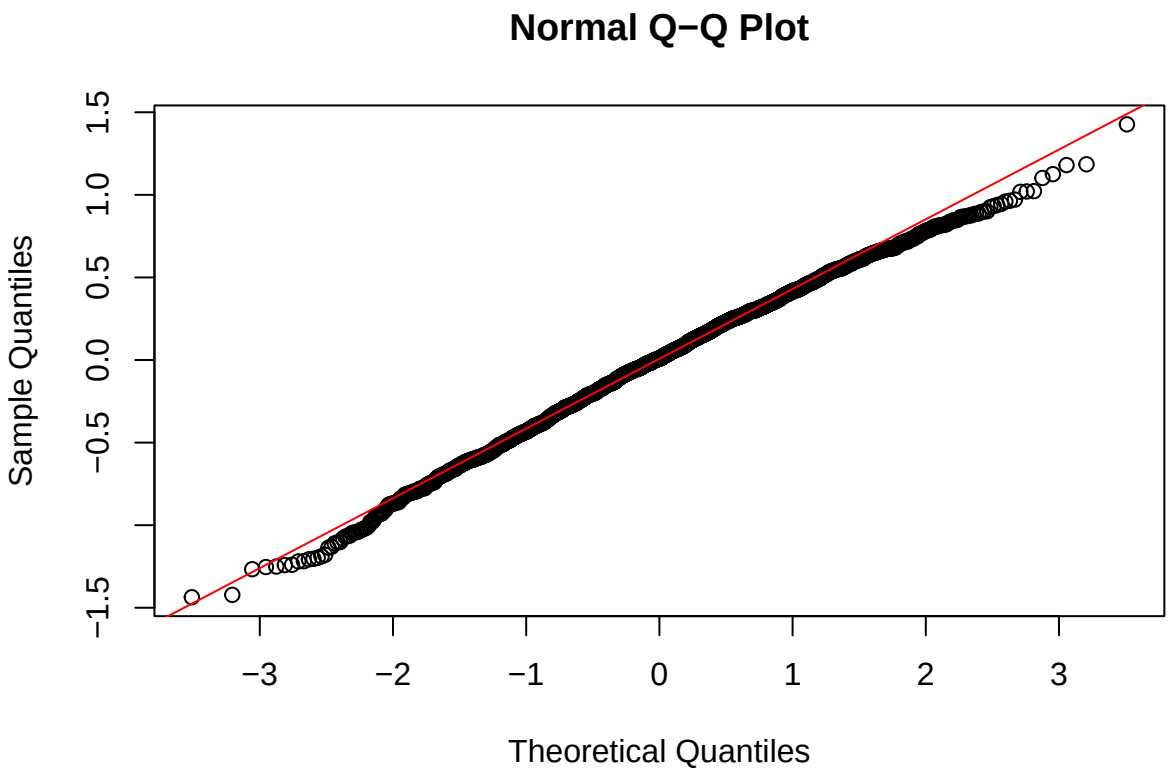
The model will fail to converge for glmer for both model.2 and model.3. Based on the model output result, model.3 is the best model, and minutes awake is a statistically significant variable.

Model Diagnostics for Our Model

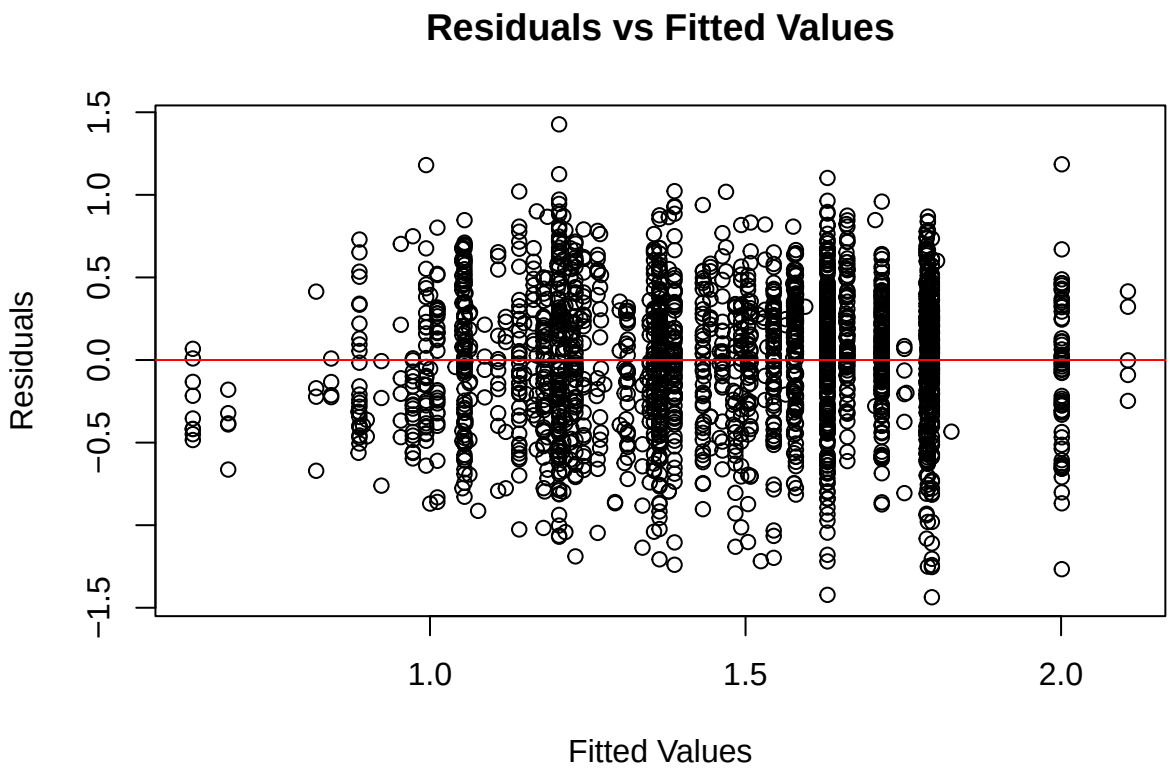
```r
residuals_model <- residuals(newmodel.1, type = "pearson")

# Plot histogram of residuals
hist(residuals_model, main = "Histogram of Residuals", xlab = "Residuals")
```



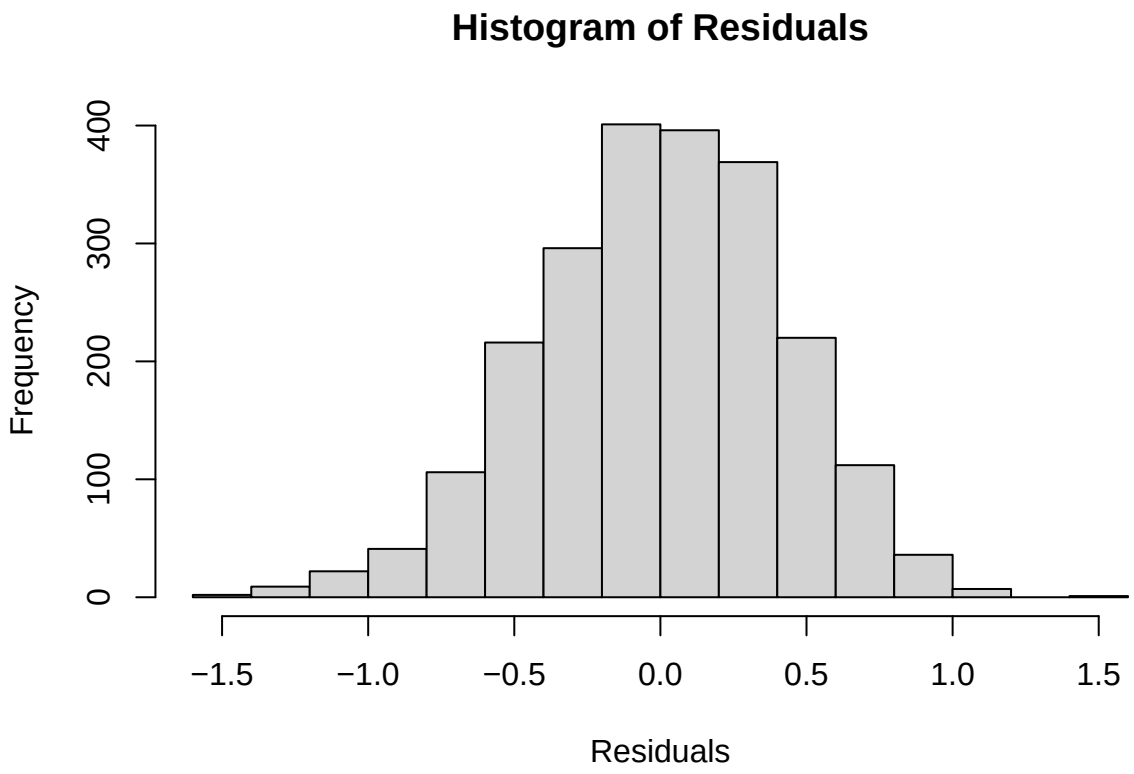

```
# Q-Q plot to check normality
qqnorm(residuals_model)
qqline(residuals_model, col = "red")
```



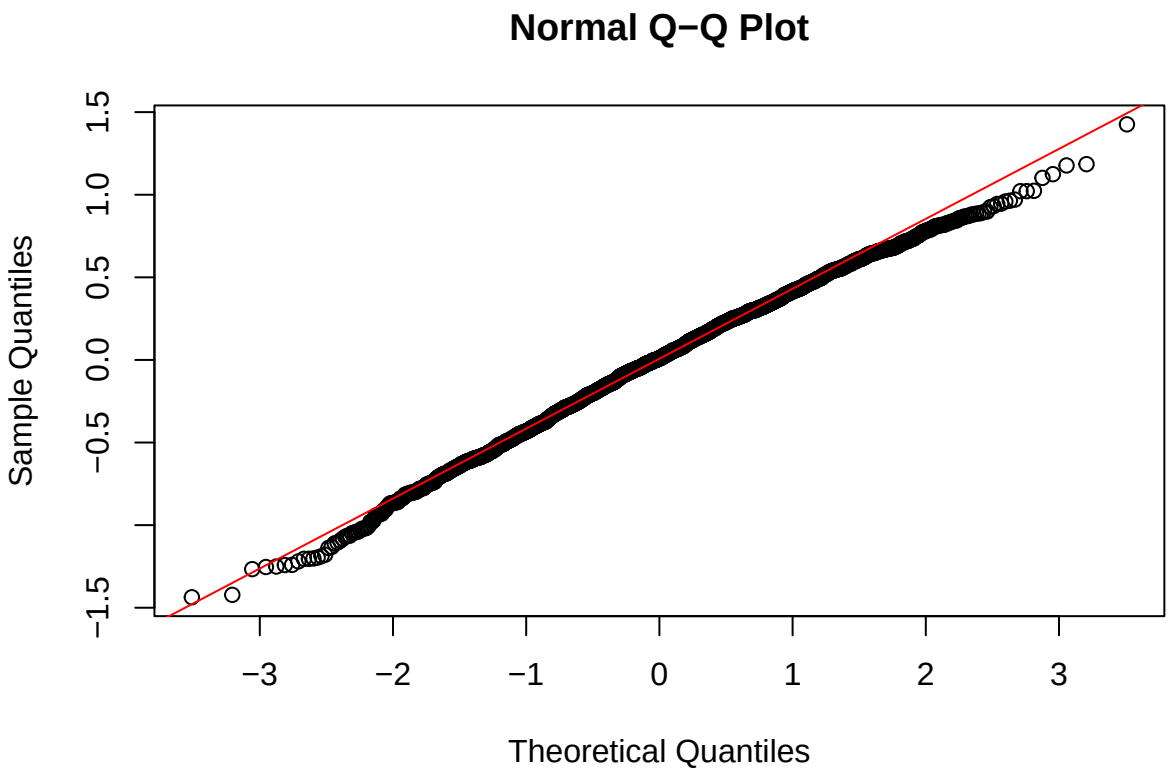

```r
fitted_values <- predict(newmodel.1, type = "response")

# Plot residuals vs fitted values
plot(fitted_values, residuals_model, main = "Residuals vs Fitted Values",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



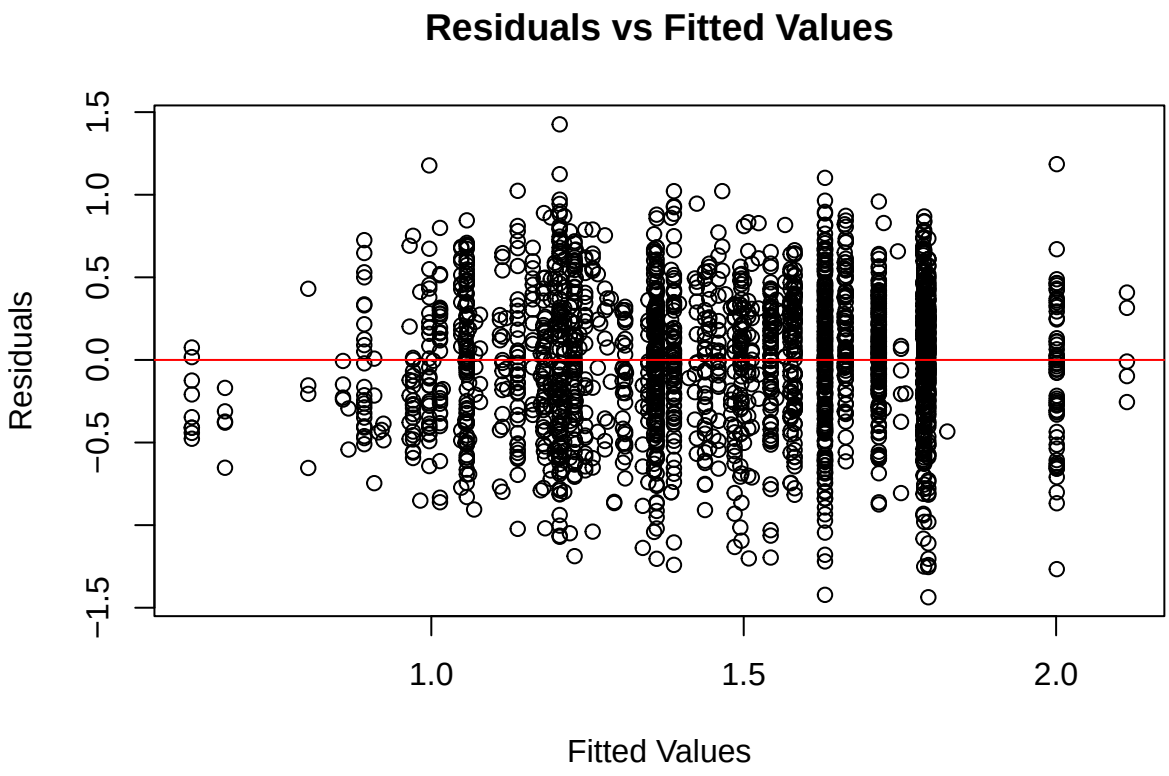

```r
residuals_model <- residuals(newmodel.2, type = "pearson")

# Plot histogram of residuals
hist(residuals_model, main = "Histogram of Residuals", xlab = "Residuals")
```



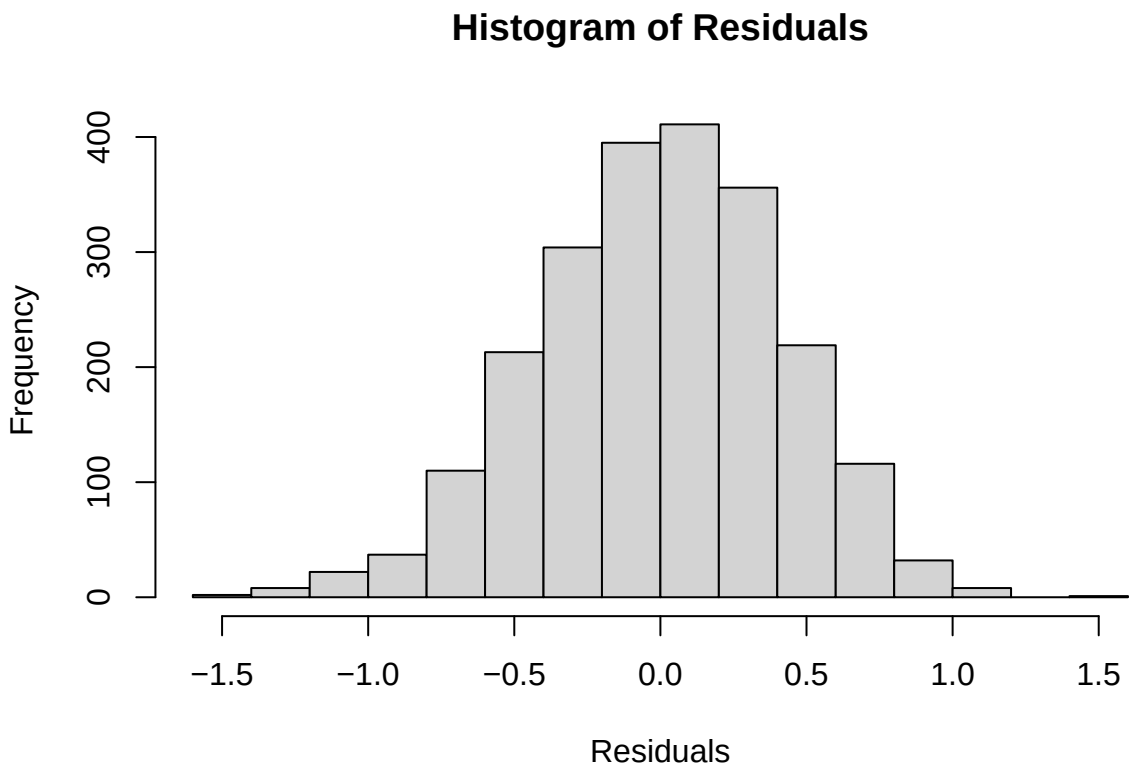

```r
# Q-Q plot to check normality
qqnorm(residuals_model)
qqline(residuals_model, col = "red")
```



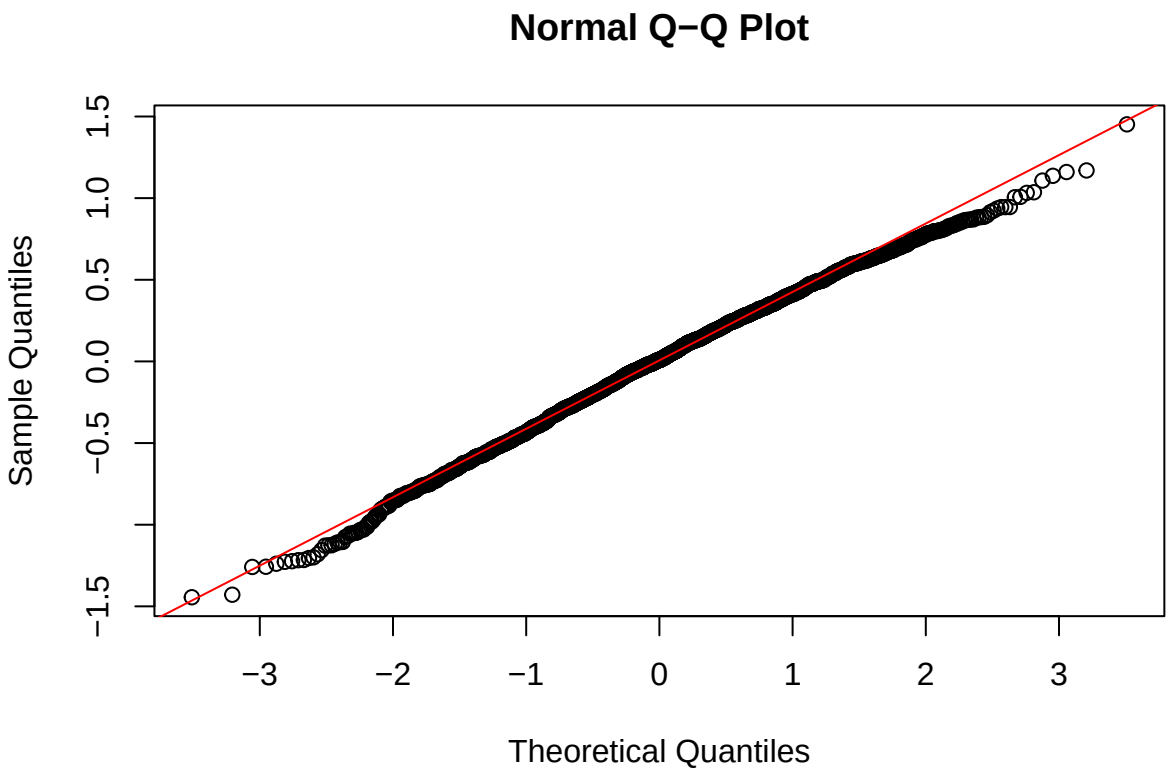

```r
fitted_values <- predict(newmodel.2, type = "response")

# Plot residuals vs fitted values
plot(fitted_values, residuals_model, main = "Residuals vs Fitted Values",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



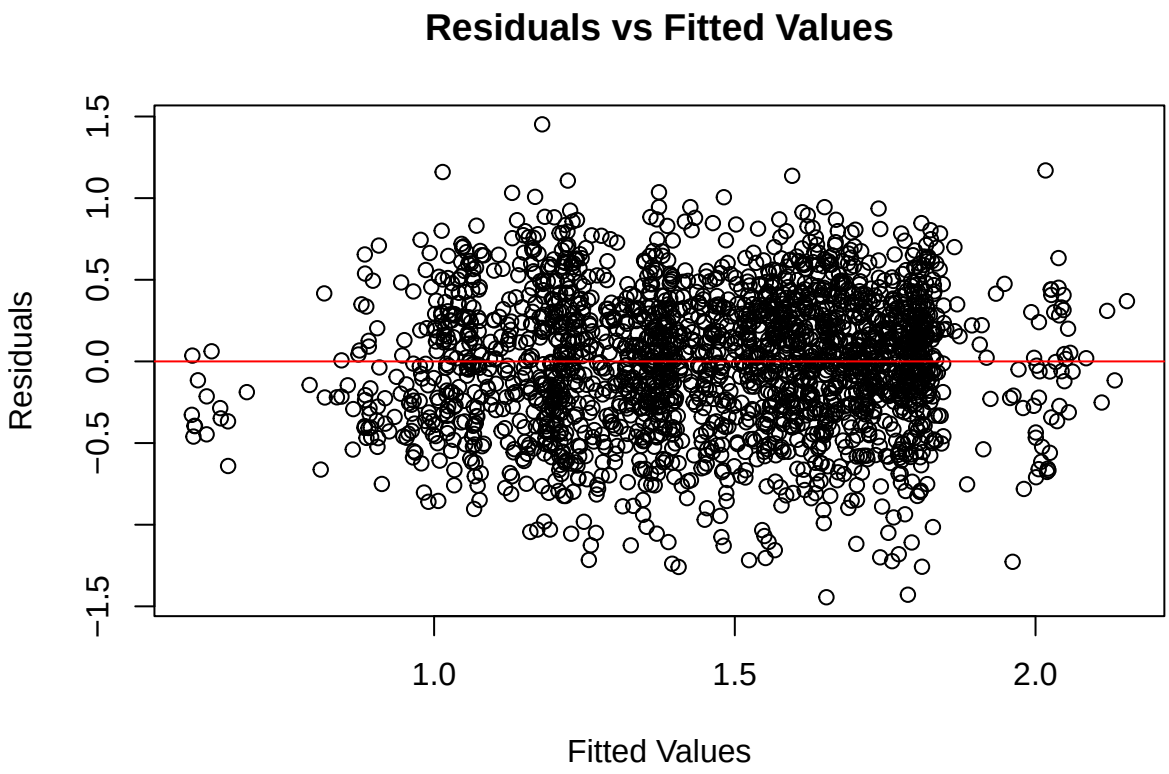

```r
residuals_model <- residuals(newmodel.3, type = "pearson")

# Plot histogram of residuals
hist(residuals_model, main = "Histogram of Residuals", xlab = "Residuals")
```




```r
# Q-Q plot to check normality
qqnorm(residuals_model)
qqline(residuals_model, col = "red")
```




```r
fitted_values <- predict(newmodel.3, type = "response")

# Plot residuals vs fitted values
plot(fitted_values, residuals_model, main = "Residuals vs Fitted Values",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```



By checking the models assumptions, all three of our models satisfy the model assumptions.

Compare Our Model to The Client's Model

```
aic_model1 <- AIC(model.1)
aic_newmodel1 <- AIC(newmodel.1)
aic_model2 <- AIC(model.2)
aic_newmodel2 <- AIC(newmodel.2)
aic_model3 <- AIC(model.3)
aic_newmodel3 <- AIC(newmodel.3)

predictions <- predict(model.1, type = "response")
mse_model1 <- mean((df_sex$DiversityScore - predictions)^2)

predictions <- predict(newmodel.1, type = "response")
mse_newmodel1 <- mean((df_sex$DiversityScore - predictions)^2)

predictions <- predict(model.2, type = "response")
mse_model2 <- mean((df_sex$DiversityScore - predictions)^2)

predictions <- predict(newmodel.2, type = "response")
mse_newmodel2 <- mean((df_sex$DiversityScore - predictions)^2)

predictions <- predict(model.3, type = "response")
mse_model3 <- mean((df_sex$DiversityScore - predictions)^2)
```

```r
predictions <- predict(newmodel.3, type = "response")
mse_newmodel3 <- mean((df_sex$DiversityScore - predictions)^2)


# Create a comparison table
comparison_table <- data.frame(
  Model = c("Model 1", "Our Model 1", "Model 2", "Our Model 2",
            "Model 3", "Our Model 3"),
  AIC = c(aic_model1, aic_newmodel1, aic_model2, aic_newmodel2,
          aic_model3, aic_newmodel3),
  MSE = c(mse_model1, mse_newmodel1, mse_model2, mse_newmodel2,
          mse_model3, mse_newmodel3)
)

# Print the table
print(comparison_table)
```

```
##         Model      AIC       MSE
## 1     Model 1 2953.148 0.3479651
## 2 Our Model 1 2660.426 0.1744798
## 3     Model 2 2868.769 0.3550065
## 4 Our Model 2 2660.977 0.1744503
## 5     Model 3 2858.630 0.3539804
## 6 Our Model 3 2651.612 0.1735441
```

Model 3 Performance Using Cross Validation

To check for the performance of the model, we propose to use k-fold cross validation:

```r
# Custom function to fit models
fit_glmmTMB <- function(train_data, test_data) {
  model <- glmmTMB(
    data = train_data,
    DiversityScore ~ Age + Sex + Minutes.awake + (1 | Name),
    family = gaussian(link = "log")
  )

  predictions <- predict(model, newdata = test_data,
                         allow.new.levels = TRUE, type = "response")
  mse_value <- mean((test_data$DiversityScore - predictions)^2)

  return(mse_value)
}

# Split data into 5 folds
set.seed(724)  # For reproducibility
folds <- createFolds(df_sex$DiversityScore, k = 5, list = TRUE)

cv_results <- sapply(folds, function(test_indices) {
  test_data <- df_sex[test_indices, ]
  train_data <- df_sex[-test_indices, ]
```

```
  fit_glmmTMB(train_data, test_data)
})

# Calculate average RMSE across folds
mean_mse <- mean(cv_results)
cat("Average MSE across folds:", mean_mse, "\n")
```

```
## Average MSE across folds: 0.1866946
```

Try XGBoost Model

```
# Load necessary libraries
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
```

```
## The following object is masked from 'package:dplyr':
##
##     slice
```

```
# Preprocess data
df_sex_clean <- df_sex %>%
  mutate(Sex = ifelse(Sex == "Male", 1, 0))

df_sex_clean <- df_sex_clean %>%
  mutate(
    Age = as.numeric(factor(Age)),  # Convert Age to numeric factor
    Name = as.numeric(factor(Name))  # Convert Name to numeric factor
  )

# Define predictor matrix (X) and target variable (y)
X <- as.matrix(df_sex_clean %>% select(-DiversityScore))
y <- df_sex_clean$DiversityScore

# Split data into training and testing sets
set.seed(724)
trainIndex <- createDataPartition(y, p = 0.8, list = FALSE)
X_train <- X[trainIndex, ]
X_test <- X[-trainIndex, ]
y_train <- y[trainIndex]
y_test <- y[-trainIndex]

# Convert to xgb.DMatrix
dtrain <- xgb.DMatrix(data = X_train, label = y_train)
dtest <- xgb.DMatrix(data = X_test, label = y_test)

# Train XGBoost model
params <- list(
  objective = "reg:squarederror",  # For regression
```

```
  eta = 0.1,  # Learning rate
  max_depth = 6,  # Tree depth
  subsample = 0.8,  # Subsampling ratio
  colsample_bytree = 0.8  # Feature sampling ratio
)
xgb_model <- xgboost(
  data = dtrain,
  params = params,
  nrounds = 100,  # Number of boosting rounds
  verbose = 1
)
```

```
## [1]  train-rmse:0.999081
## [2]  train-rmse:0.915262
## [3]  train-rmse:0.841185
## [4]  train-rmse:0.776013
## [5]  train-rmse:0.718494
## [6]  train-rmse:0.666325
## [7]  train-rmse:0.623359
## [8]  train-rmse:0.584447
## [9]  train-rmse:0.550861
## [10] train-rmse:0.521529
## [11] train-rmse:0.494605
## [12] train-rmse:0.471920
## [13] train-rmse:0.451682
## [14] train-rmse:0.434651
## [15] train-rmse:0.419558
## [16] train-rmse:0.406694
## [17] train-rmse:0.395969
## [18] train-rmse:0.386843
## [19] train-rmse:0.378037
## [20] train-rmse:0.371143
## [21] train-rmse:0.365672
## [22] train-rmse:0.359711
## [23] train-rmse:0.354481
## [24] train-rmse:0.349467
## [25] train-rmse:0.345572
## [26] train-rmse:0.341781
## [27] train-rmse:0.338476
## [28] train-rmse:0.335485
## [29] train-rmse:0.332190
## [30] train-rmse:0.329362
## [31] train-rmse:0.326964
## [32] train-rmse:0.325021
## [33] train-rmse:0.322834
## [34] train-rmse:0.320703
## [35] train-rmse:0.319267
## [36] train-rmse:0.318097
## [37] train-rmse:0.316775
## [38] train-rmse:0.314992
## [39] train-rmse:0.313539
## [40] train-rmse:0.312173
## [41] train-rmse:0.311144
```

```
## [42] train-rmse:0.309729
## [43] train-rmse:0.307538
## [44] train-rmse:0.306973
## [45] train-rmse:0.305115
## [46] train-rmse:0.303946
## [47] train-rmse:0.302123
## [48] train-rmse:0.299651
## [49] train-rmse:0.297747
## [50] train-rmse:0.295209
## [51] train-rmse:0.293348
## [52] train-rmse:0.292339
## [53] train-rmse:0.290950
## [54] train-rmse:0.290231
## [55] train-rmse:0.288539
## [56] train-rmse:0.286567
## [57] train-rmse:0.285195
## [58] train-rmse:0.283385
## [59] train-rmse:0.282691
## [60] train-rmse:0.280947
## [61] train-rmse:0.280023
## [62] train-rmse:0.278611
## [63] train-rmse:0.277362
## [64] train-rmse:0.276238
## [65] train-rmse:0.275350
## [66] train-rmse:0.274320
## [67] train-rmse:0.272994
## [68] train-rmse:0.272738
## [69] train-rmse:0.271926
## [70] train-rmse:0.271020
## [71] train-rmse:0.269777
## [72] train-rmse:0.269325
## [73] train-rmse:0.268247
## [74] train-rmse:0.267922
## [75] train-rmse:0.267336
## [76] train-rmse:0.266244
## [77] train-rmse:0.264863
## [78] train-rmse:0.263179
## [79] train-rmse:0.262018
## [80] train-rmse:0.261327
## [81] train-rmse:0.260844
## [82] train-rmse:0.259315
## [83] train-rmse:0.257789
## [84] train-rmse:0.255886
## [85] train-rmse:0.255164
## [86] train-rmse:0.253895
## [87] train-rmse:0.252906
## [88] train-rmse:0.252328
## [89] train-rmse:0.251659
## [90] train-rmse:0.250209
## [91] train-rmse:0.249625
## [92] train-rmse:0.248997
## [93] train-rmse:0.248583
## [94] train-rmse:0.247658
## [95] train-rmse:0.246951
```

```
## [96] train-rmse:0.245514
## [97] train-rmse:0.244221
## [98] train-rmse:0.243286
## [99] train-rmse:0.241949
## [100]    train-rmse:0.241343
```

```r
# Predict on test set
y_pred <- predict(xgb_model, dtest)

# Evaluate model
mse <- mean((y_test - y_pred)^2)
cat("Mean Squared Error (XGBoost):", mse, "\n")
```

```
## Mean Squared Error (XGBoost): 0.1591424
```