mlproject

Nuo Shu, Rui Tang, Zhewei Liang

2024-04-14

```r
# Load necessary libraries
library(tidyverse)  # For data manipulation and visualization
library(zoo)         # For time series manipulation
library(faraway)     # For statistical modeling functions
library(MASS)        # For statistical methods and data manipulation
library(leaps)       # For subset selection methods
library(glmnet)      # For Lasso and Ridge regression models
library(caret)       # For model training and evaluation workflows
library(boot)        # For bootstrap resampling and prediction intervals
library(randomForest) # For Random Forest models
library(knitr)       # For generating tables in the output
library(ggplot2)     # For data visualization
```

1. Load the data into R.

```r
rm(list = ls()) # Clear the environment
data <- read.csv("Sleep_Efficiency.csv")
```

2. Perform data cleaning.

```r
data <- data[ , -1]

# Convert Bedtime to numeric
data$Bedtime <- as.numeric(as.POSIXct(data$Bedtime)) - 1600000000
data$Wakeup.time <- as.numeric(as.POSIXct(data$Wakeup.time)) - 1600000000
```

```r
# Remove columns with >30% missing values
data <- data[, colMeans(is.na(data)) <= 0.3]
colSums(is.na(data))
```

```
##                    Age                 Gender                Bedtime
##                      0                      0                      0
##            Wakeup.time         Sleep.duration       Sleep.efficiency
##                      0                      0                      0
##    REM.sleep.percentage  Deep.sleep.percentage Light.sleep.percentage
##                      0                      0                      0
##             Awakenings   Caffeine.consumption    Alcohol.consumption
##                     20                     25                     16
##         Smoking.status     Exercise.frequency
##                      0                      6
```

```r
# Impute missing values for specific features
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Compute means for imputation
mean_Awakenings <- mean(data$Awakenings, na.rm = TRUE)
mean_Caffeine.consumption <- mean(data$Caffeine.consumption, na.rm = TRUE)
mean_Alcohol.consumption <- mean(data$Alcohol.consumption, na.rm = TRUE)
mean_Exercise.frequency <- mean(data$Exercise.frequency, na.rm = TRUE)

# Impute missing values
data$Awakenings <- replace(data$Awakenings, is.na(data$Awakenings), mean_Awakenings)
data$Caffeine.consumption <- replace(data$Caffeine.consumption, is.na(data$Caffeine.consumption), mean_C
data$Alcohol.consumption <- replace(data$Alcohol.consumption, is.na(data$Alcohol.consumption), mean_Alc
data$Exercise.frequency <- replace(data$Exercise.frequency, is.na(data$Exercise.frequency), mean_Exerci

# Encode categorical variables as factors
data$Gender <- factor(data$Gender)
data$Smoking.status <- factor(data$Smoking.status)
```

3. Randomly split the data into training (80%) and test (20%) sets.

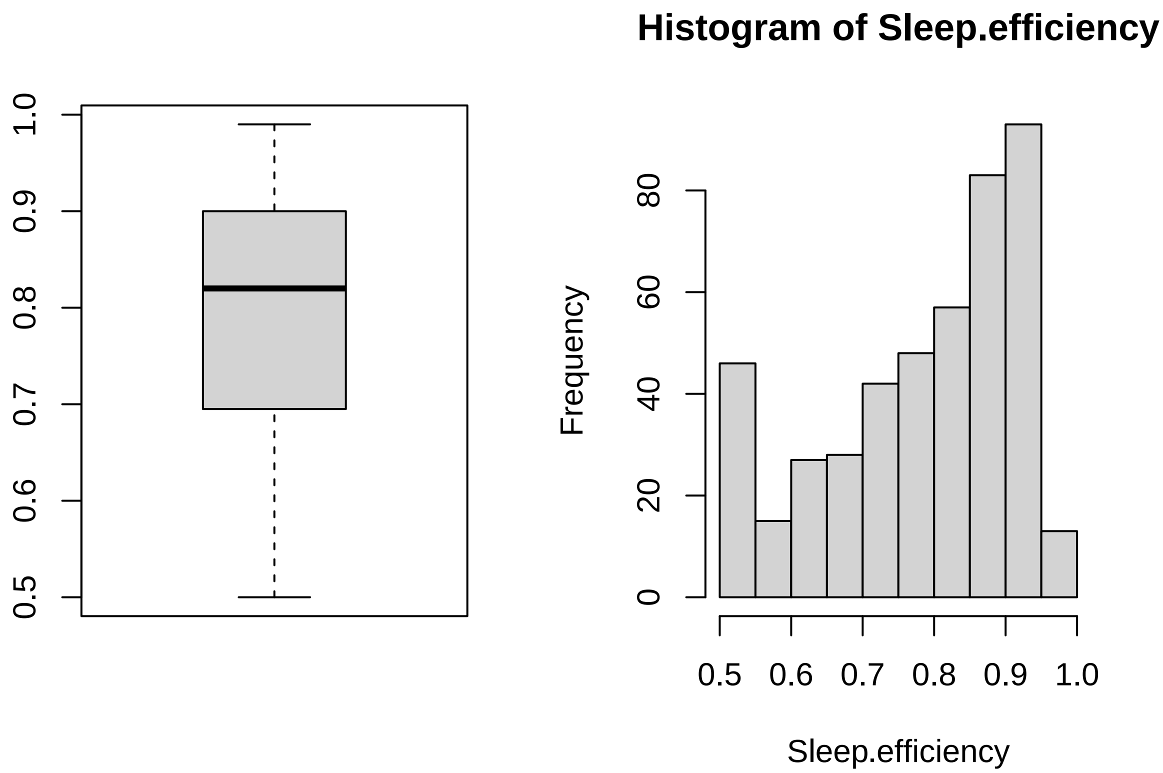
```r
set.seed(0) #Set the random seed for reproducibility
n <- nrow(data)
n_tr <- round(n * 0.8) #Number of observations in the training set (80%)
ind_tr <- sample(n, n_tr) #Randomly choose n_tr numbers from numbers 1 to n
data_tr <- data[ind_tr, ]
data_te <- data[-ind_tr, ]
```

4.Identify the response variable, which must be continuous, and the predictors, which cannot all be categorical.

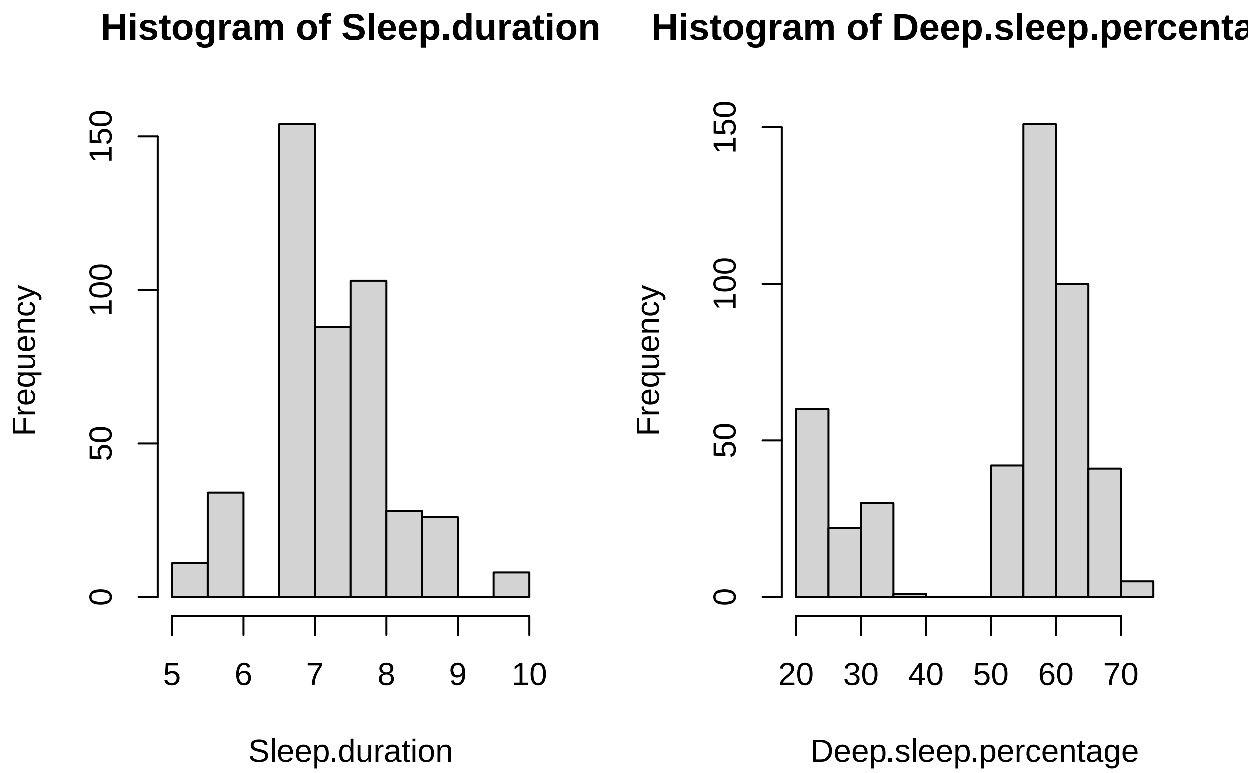
Sleep Efficiency is a continuous response variable. Predictors include numeric, categorical, and time-related features.

5. Conduct preliminary summary statistics and create graphs

```r
# Boxplot and histogram for Sleep Efficiency
attach(data)
par(mfrow = c(1,2))
boxplot(Sleep.efficiency)
hist(Sleep.efficiency)
```




```r
# Boxplot and histogram for Sleep Duration
attach(data)
par(mfrow = c(1,2))
hist(Sleep.duration)
hist(Deep.sleep.percentage)
```




```r
# T-test for Gender and Smoking status
t.test(Sleep.efficiency ~ Gender, data, var.equal=TRUE)
```

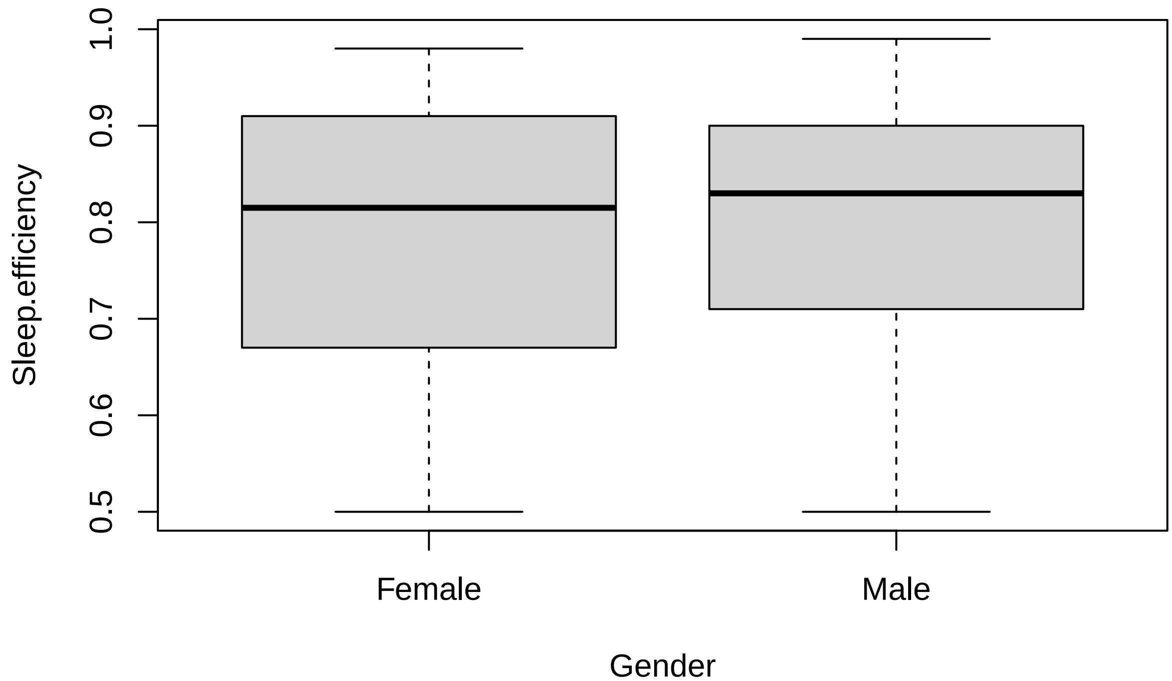
```
##
##  Two Sample t-test
##
## data:  Sleep.efficiency by Gender
## t = -0.21345, df = 450, p-value = 0.8311
## alternative hypothesis: true difference in means between group Female and group Male is not equal to
## 95 percent confidence interval:
##  -0.02774797  0.02231094
## sample estimates:
## mean in group Female   mean in group Male
##            0.7875446            0.7902632
```

```r
t.test(Sleep.efficiency ~ Smoking.status, data, var.equal=TRUE)
```

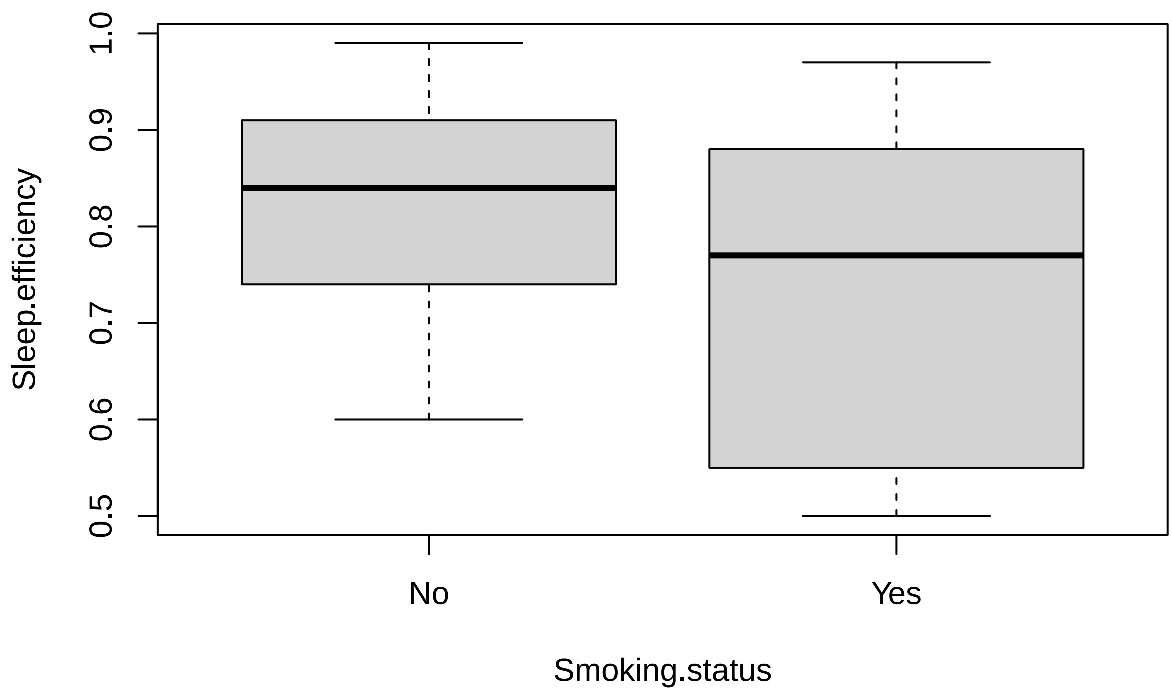
```
##
##  Two Sample t-test
##
## data:  Sleep.efficiency by Smoking.status
## t = 7.4558, df = 450, p-value = 4.629e-13
## alternative hypothesis: true difference in means between group No and group Yes is not equal to 0
## 95 percent confidence interval:
##  0.06888258 0.11819347
## sample estimates:
##  mean in group No mean in group Yes
```

```
##           0.8222337           0.7286957
```

```r
plot(Sleep.efficiency ~ Gender, data = data)
```




```r
plot(Sleep.efficiency ~ Smoking.status, data = data)
```



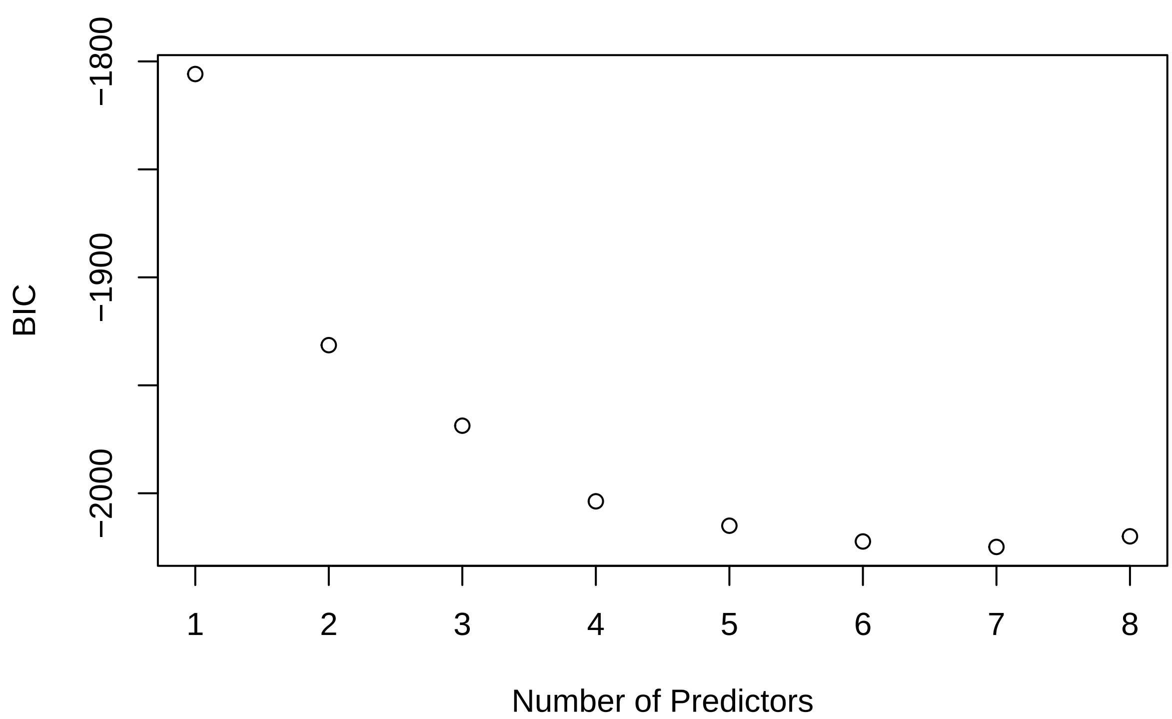
6. Feature selection

```
# Subset selection
b <- regsubsets(Sleep.efficiency ~ Age + Gender + Bedtime + Sleep.duration +
              REM.sleep.percentage + Deep.sleep.percentage + Awakenings +
              Caffeine.consumption + Alcohol.consumption + Smoking.status +
              Exercise.frequency, data = data_tr)
rs <- summary(b)
rs$which
```

```
##   (Intercept)   Age GenderMale Bedtime Sleep.duration REM.sleep.percentage
## 1        TRUE FALSE      FALSE   FALSE          FALSE                FALSE
## 2        TRUE FALSE      FALSE   FALSE          FALSE                FALSE
## 3        TRUE FALSE      FALSE   FALSE          FALSE                FALSE
## 4        TRUE FALSE      FALSE   FALSE          FALSE                 TRUE
## 5        TRUE FALSE      FALSE   FALSE          FALSE                 TRUE
## 6        TRUE  TRUE      FALSE   FALSE          FALSE                 TRUE
## 7        TRUE  TRUE      FALSE   FALSE          FALSE                 TRUE
## 8        TRUE  TRUE      FALSE   FALSE          FALSE                 TRUE
##   Deep.sleep.percentage Awakenings Caffeine.consumption Alcohol.consumption
## 1                  TRUE      FALSE                FALSE               FALSE
## 2                  TRUE       TRUE                FALSE               FALSE
## 3                  TRUE       TRUE                FALSE               FALSE
## 4                  TRUE       TRUE                FALSE               FALSE
## 5                  TRUE       TRUE                FALSE                TRUE
## 6                  TRUE       TRUE                FALSE                TRUE
```

```
## 7                    TRUE        TRUE                  FALSE                 TRUE
## 8                    TRUE        TRUE                   TRUE                 TRUE
##   Smoking.statusYes Exercise.frequency
## 1             FALSE              FALSE
## 2             FALSE              FALSE
## 3              TRUE              FALSE
## 4              TRUE              FALSE
## 5              TRUE              FALSE
## 6              TRUE              FALSE
## 7              TRUE               TRUE
## 8              TRUE               TRUE
```

```r
# Calculate BIC
BIC <- n_tr*log(rs$rss/n_tr) + (2:9)*log(n_tr)
plot(BIC ~ I(1:8), ylab="BIC", xlab="Number of Predictors")
```




```r
which.min(BIC)
```

```
## [1] 7
```

7. Fit a linear model.

```r
# Set seed for reproducibility
set.seed(0)

# 10-fold CV control
cv_control <- trainControl(method = "cv", number=10, savePredictions = "final")
```

```r
linear_model <- train(Sleep.efficiency ~ Age + REM.sleep.percentage +
    Deep.sleep.percentage + Awakenings + Alcohol.consumption +
    Smoking.status + Exercise.frequency, data = data_tr,
              method = 'lm', trControl = cv_control)
summary(linear_model)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.167602 -0.035455  0.004717  0.039881  0.140618
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.4326746  0.0289056  14.969  < 2e-16 ***
## Age                     0.0007761  0.0002337   3.320 0.000993 ***
## REM.sleep.percentage    0.0048927  0.0007821   6.256 1.14e-09 ***
## Deep.sleep.percentage   0.0053561  0.0002355  22.746  < 2e-16 ***
## Awakenings             -0.0327792  0.0024388 -13.441  < 2e-16 ***
## Alcohol.consumption    -0.0090583  0.0021087  -4.296 2.25e-05 ***
## Smoking.statusYes      -0.0475382  0.0066104  -7.191 3.84e-12 ***
## Exercise.frequency      0.0064532  0.0022352   2.887 0.004127 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0578 on 354 degrees of freedom
## Multiple R-squared:  0.8167, Adjusted R-squared:  0.8131
## F-statistic: 225.4 on 7 and 354 DF,  p-value: < 2.2e-16
```

8. Run ridge and lasso model.

```r
set.seed(0)

ridge_model <- train(Sleep.efficiency ~ Age + REM.sleep.percentage +
    Deep.sleep.percentage + Awakenings + Alcohol.consumption +
    Smoking.status + Exercise.frequency, data = data_tr,
    method = 'glmnet',
    trControl = cv_control,
    tuneGrid = expand.grid(alpha = 0,lambda = 10^seq(3,-2,length = 100)),
    preProcess = "scale",
    family = "gaussian")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```r
lasso_model <- train(Sleep.efficiency ~ Age + REM.sleep.percentage +
    Deep.sleep.percentage + Awakenings + Alcohol.consumption +
    Smoking.status + Exercise.frequency, data = data_tr,
    method = 'glmnet',
    trControl = cv_control,
    tuneGrid = expand.grid(alpha = 1,lambda = 10^seq(3,-2,length = 100)),
    preProcess = "scale",
```

```r
    family = "gaussian")
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```r
summary(ridge_model)
```

```
##             Length Class      Mode
## a0          100    -none-     numeric
## beta        700    dgCMatrix  S4
## df          100    -none-     numeric
## dim           2    -none-     numeric
## lambda      100    -none-     numeric
## dev.ratio   100    -none-     numeric
## nulldev       1    -none-     numeric
## npasses       1    -none-     numeric
## jerr          1    -none-     numeric
## offset        1    -none-     logical
## call          5    -none-     call
## nobs          1    -none-     numeric
## lambdaOpt     1    -none-     numeric
## xNames        7    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         1    -none-     list
```

```r
summary(lasso_model)
```

```
##             Length Class      Mode
## a0           62    -none-     numeric
## beta        434    dgCMatrix  S4
## df           62    -none-     numeric
## dim           2    -none-     numeric
## lambda       62    -none-     numeric
## dev.ratio    62    -none-     numeric
## nulldev       1    -none-     numeric
## npasses       1    -none-     numeric
## jerr          1    -none-     numeric
## offset        1    -none-     logical
## call          5    -none-     call
## nobs          1    -none-     numeric
## lambdaOpt     1    -none-     numeric
## xNames        7    -none-     character
## problemType   1    -none-     character
## tuneValue     2    data.frame list
## obsLevels     1    -none-     logical
## param         1    -none-     list
```

```r
# Best lambdas for Ridge and Lasso model
ridge_model$bestTune
```

```
##   alpha lambda
## 1     0   0.01
```

```r
lasso_model$bestTune
```

```
##   alpha lambda
## 1     1   0.01
```

```r
# Linear Model Coefficients
coef_linear <- summary(linear_model)$coefficients[, "Estimate"]
df_linear <- data.frame(Variable = rownames(summary(linear_model)$coefficients),
                        Estimate_linear = coef_linear)


# Ridge Model Coefficients
coef_ridge <- coef(ridge_model$finalModel, s = ridge_model$bestTune$lambda)
df_ridge <- data.frame(
  Variable = rownames(coef_ridge),
  Estimate_ridge = as.vector(coef_ridge),
  row.names = NULL
)

# Lasso Model Coefficients
coef_lasso <- coef(lasso_model$finalModel, s = lasso_model$bestTune$lambda)
df_lasso <- data.frame(
  Variable = rownames(coef_lasso),
  Estimate_lasso = as.vector(coef_lasso),
  row.names = NULL
)

# Combine into a single data frame for comparison
df_combined <- data.frame(
  Variable = df_linear$Variable,
  Linear = df_linear$Estimate_linear,
  Ridge = df_ridge$Estimate_ridge,
  Lasso = df_lasso$Estimate_lasso
)

kable(df_combined, caption = "Variables and Estimates of Models")
```

Table 1: Variables and Estimates of Models

Variable	Linear	Ridge	Lasso
(Intercept)	0.4326746	0.4732153	0.5344961
Age	0.0007761	0.0091345	0.0008185
REM.sleep.percentage	0.0048927	0.0170214	0.0087537
Deep.sleep.percentage	0.0053561	0.0768812	0.0782523
Awakenings	-0.0327792	-0.0426182	-0.0386606
Alcohol.consumption	-0.0090583	-0.0164229	-0.0094342
Smoking.statusYes	-0.0475382	-0.0222049	-0.0143153
Exercise.frequency	0.0064532	0.0099479	0.0029628

Interpretation:

Intercept: Indicates the baseline level of sleep efficiency when all predictors are zero.

Age: Shows a positive correlation with sleep efficiency in all models, indicating that older individuals may have higher sleep efficiency.

REM Sleep Percentage: Positive in all models, suggesting that higher REM sleep correlates with increased sleep efficiency.

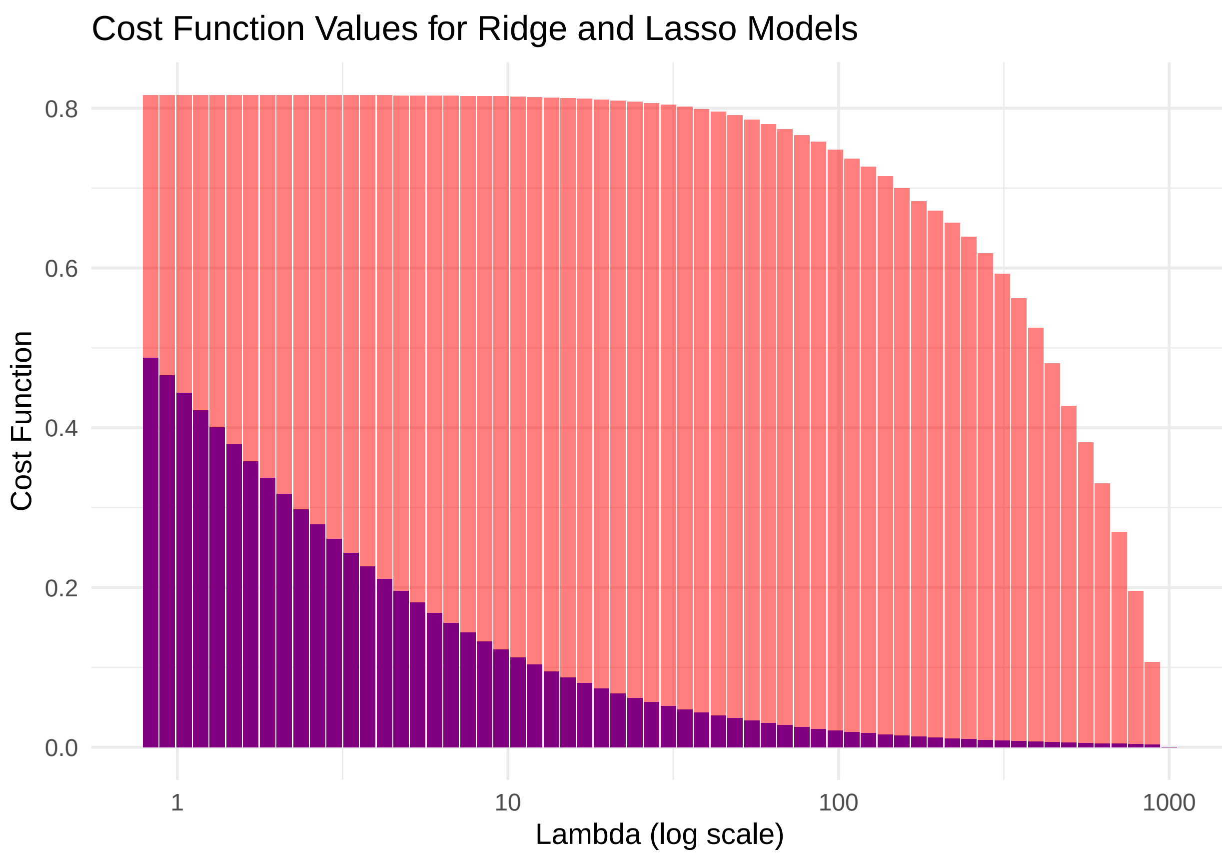
Deep Sleep Percentage: Positive coefficients across models, indicating that higher deep sleep percentages correspond to higher sleep efficiency.

Awakenings: Negative coefficients, suggesting that frequent awakenings reduce sleep efficiency.

Alcohol Consumption: Negative coefficients, implying that higher alcohol intake correlates with lower sleep efficiency.

Smoking Status: Negative coefficients for Lasso and Linear, indicating that smokers may have lower sleep efficiency.

```r
# Extracting cost function values
ridge_costs <- ridge_model$finalModel$dev.ratio
lasso_costs <- lasso_model$finalModel$dev.ratio

# Creating a data frame for visualization
lambda_values <- 10^seq(3, -2, length = 100)

# Ensuring consistent lengths
min_len <- min(length(ridge_costs), length(lasso_costs), length(lambda_values))
ridge_costs <- ridge_costs[1:min_len]
lasso_costs <- lasso_costs[1:min_len]
lambda_values <- lambda_values[1:min_len]

df <- data.frame(Lambda = lambda_values, Ridge_Cost = ridge_costs, Lasso_Cost = lasso_costs)

# Plotting the bar chart
ggplot(df, aes(x = Lambda)) +
  geom_bar(aes(y = Ridge_Cost), stat = "identity", fill = "blue") +
  geom_bar(aes(y = Lasso_Cost), stat = "identity", fill = "red", alpha = 0.5) +
  scale_x_log10() +
  labs(title = "Cost Function Values for Ridge and Lasso Models",
       x = "Lambda (log scale)", y = "Cost Function") +
  theme_minimal()
```



9. Run random forest model.

```
set.seed(0)

rf_model <- train(Sleep.efficiency ~ Age + REM.sleep.percentage +
    Deep.sleep.percentage + Awakenings + Alcohol.consumption +
    Smoking.status + Exercise.frequency, data = data_tr,
    method = 'rf',
    trControl = cv_control,
    ntree = 20)
```

10. Provide interpretation, inference, and make predictions based on the models.

```
# Predicting on training data
red1 <- predict(linear_model, newdata = data_tr)
red2 <- predict(ridge_model, newdata = data_tr)
red3 <- predict(lasso_model, newdata = data_tr)
red4 <- predict(rf_model, newdata = data_tr)

# Residual sum of squares (SSR) for training data
ssr_tr1 <- sum((red1 - data_tr$Sleep.efficiency)^2)
ssr_tr2 <- sum((red2 - data_tr$Sleep.efficiency)^2)
ssr_tr3 <- sum((red3 - data_tr$Sleep.efficiency)^2)
ssr_tr4 <- sum((red4 - data_tr$Sleep.efficiency)^2)
```

```r
# Total sum of squares (TSS)
tss_tr <- sum((data_tr$Sleep.efficiency - mean(data_tr$Sleep.efficiency))^2)

# Compute R-squared values for training set
r_squared_tr1 <- 1 - (ssr_tr1 / tss_tr)
r_squared_tr2 <- 1 - (ssr_tr2 / tss_tr)
r_squared_tr3 <- 1 - (ssr_tr3 / tss_tr)
r_squared_tr4 <- 1 - (ssr_tr4 / tss_tr)
```

```r
# Number of observations and predictors
n <- nrow(data_tr)
p1 <- length(coef(linear_model$finalModel)) - 1
p2 <- length(coef(ridge_model$finalModel, s = ridge_model$bestTune$lambda)) - 1
p3 <- length(coef(lasso_model$finalModel, s = lasso_model$bestTune$lambda)) - 1
p4 <- length(rf_model$finalModel$importance) # Adjust this as needed for Random Forest

# Compute Adjusted R-squared
adj_r_squared_tr1 <- 1 - (1 - r_squared_tr1) * (n - 1) / (n - p1 - 1)
adj_r_squared_tr2 <- 1 - (1 - r_squared_tr2) * (n - 1) / (n - p2 - 1)
adj_r_squared_tr3 <- 1 - (1 - r_squared_tr3) * (n - 1) / (n - p3 - 1)
adj_r_squared_tr4 <- 1 - (1 - r_squared_tr4) * (n - 1) / (n - p4 - 1)

# Create a data frame for display
df <- data.frame(models = c("Linear Model", "Ridge Model", "Lasso Model", "Random Forest"),
        adj_r_squared = c(adj_r_squared_tr1, adj_r_squared_tr2, adj_r_squared_tr3, adj_r_squared_tr4))

kable(df, caption = "Adjusted R-squared of Models in Training Set")
```

Table 2: Adjusted R-squared of Models in Training Set

models	adj_r_squared
Linear Model	0.8130980
Ridge Model	0.8102320
Lasso Model	0.7838983
Random Forest	0.9646202

Interpretation:

Random Forest: 0.9646, indicating that it explains 96% of the variance, making it the most comprehensive model.

Linear Model: 0.8131, showing that it explains 81% of the variance.

Ridge: 0.8102, close to the linear model's performance.

Lasso: 0.7839, slightly lower, indicating it may struggle with over-regularization.

```r
# MSE for training set
mse_tr1 <- sum((red1 - data_tr$Sleep.efficiency)^2) / nrow(data_tr)
mse_tr2 <- sum((red2 - data_tr$Sleep.efficiency)^2) / nrow(data_tr)
mse_tr3 <- sum((red3 - data_tr$Sleep.efficiency)^2) / nrow(data_tr)
mse_tr4 <- sum((red4 - data_tr$Sleep.efficiency)^2) / nrow(data_tr)

df <- data.frame(models = c("Linear Model", "Ridge Model", "Lasso Model", "Random Forest"),
        mse = c(mse_tr1, mse_tr2, mse_tr3, mse_tr4))
```

```r
kable(df, caption = "Mean Square Error of Models in Training Set")
```

Table 3: Mean Square Error of Models in Training Set

models	mse
Linear Model	0.0032675
Ridge Model	0.0033176
Lasso Model	0.0037780
Random Forest	0.0006185

Interpretation:

Random Forest: 0.0006, showing the lowest error, indicating it captures data patterns effectively.

Linear: 0.0033, demonstrating moderate accuracy.

Ridge: 0.0033, similar to the linear model.

Lasso: 0.0038, slightly higher error, suggesting over-regularization.

```r
# Predicting on the test set
prediction1 <- predict(linear_model, newdata = data_te)
prediction2 <- predict(ridge_model, newdata = data_te)
prediction3 <- predict(lasso_model, newdata = data_te)
prediction4 <- predict(rf_model, newdata = data_te)

# MSE for the test set
mse_te1 <- sum((prediction1 - data_te$Sleep.efficiency)^2) / nrow(data_te)
mse_te2 <- sum((prediction2 - data_te$Sleep.efficiency)^2) / nrow(data_te)
mse_te3 <- sum((prediction3 - data_te$Sleep.efficiency)^2) / nrow(data_te)
mse_te4 <- sum((prediction4 - data_te$Sleep.efficiency)^2) / nrow(data_te)

df <- data.frame(models = c("Linear Model", "Ridge Model", "Lasso Model", "Random Forest"),
        mse = c(mse_te1, mse_te2, mse_te3, mse_te4))

kable(df, caption = "Mean Square Error of Models in Test Set")
```

Table 4: Mean Square Error of Models in Test Set

models	mse
Linear Model	0.0040854
Ridge Model	0.0042975
Lasso Model	0.0048101
Random Forest	0.0025413

Interpretation:

Random Forest: 0.0025, indicating it generalizes well to unseen data.

Linear: 0.0041, suggesting reasonable generalization.

Ridge: 0.0043, slightly higher error.

Lasso: 0.0048, showing the highest error, indicating potential over-regularization or lack of generalization.

```r
# Prediction rates via bootstrap intervals
bootstrap_prediction_intervals <- function(model, data, n_bootstraps = 1000) {
  boot_preds <- boot(data, function(data, indices) {
    data_boot <- data[indices, ]
    predict(model, newdata = data_boot)
  }, R = n_bootstraps)
  # Calculate 95% confidence intervals
  ci_bounds <- t(apply(boot_preds$t, 2, function(pred) {
    quantile(pred, c(0.025, 0.975))
  }))
  # Check if actual values fall within CI
  within_ci <- (data[["Sleep.efficiency"]] >= ci_bounds[, 1]) & (data[["Sleep.efficiency"]] <= ci_bounds
  prediction_rate <- mean(within_ci)

  return(prediction_rate)
}
```

```r
# Prediction Rates for training set
rate_tr1 <- bootstrap_prediction_intervals(linear_model, data_tr )
rate_tr2 <- bootstrap_prediction_intervals(ridge_model, data_tr )
rate_tr3 <- bootstrap_prediction_intervals(lasso_model, data_tr )
rate_tr4 <- bootstrap_prediction_intervals(rf_model, data_tr)

df <- data.frame(models = c("Linear Model", "Ridge Model", "Lasso Model", "Random Forest"),
        rates = c(rate_tr1, rate_tr2, rate_tr3, rate_tr4))

kable(df, caption = "Prediction Rates of Models in Training Set")
```

Table 5: Prediction Rates of Models in Training Set

models	rates
Linear Model	0.8176796
Ridge Model	0.7790055
Lasso Model	0.6574586
Random Forest	0.8618785

Interpretation:

Random Forest: 86%, indicating robust performance.

Linear: 82%, showing good prediction accuracy.

Ridge: 78%, suggesting some consistency.

Lasso: 66%, indicating potential inconsistencies.

```r
# Prediction Rates for test set
rate_te1 <- bootstrap_prediction_intervals(linear_model, data_te )
rate_te2 <- bootstrap_prediction_intervals(ridge_model, data_te )
rate_te3 <- bootstrap_prediction_intervals(lasso_model, data_te )
rate_te4 <- bootstrap_prediction_intervals(rf_model, data_te)

df <- data.frame(models = c("Linear Model", "Ridge Model", "Lasso Model", "Random Forest"),
        rates = c(rate_te1, rate_te2, rate_te3, rate_te4))
```

```r
kable(df, caption = "Prediction Rates of Models in Test Set")
```

Table 6: Prediction Rates of Models in Test Set

models	rates
Linear Model	0.7555556
Ridge Model	0.7222222
Lasso Model	0.6333333
Random Forest	0.7666667

Interpretation:

Random Forest: 77%, showing it generalizes well.

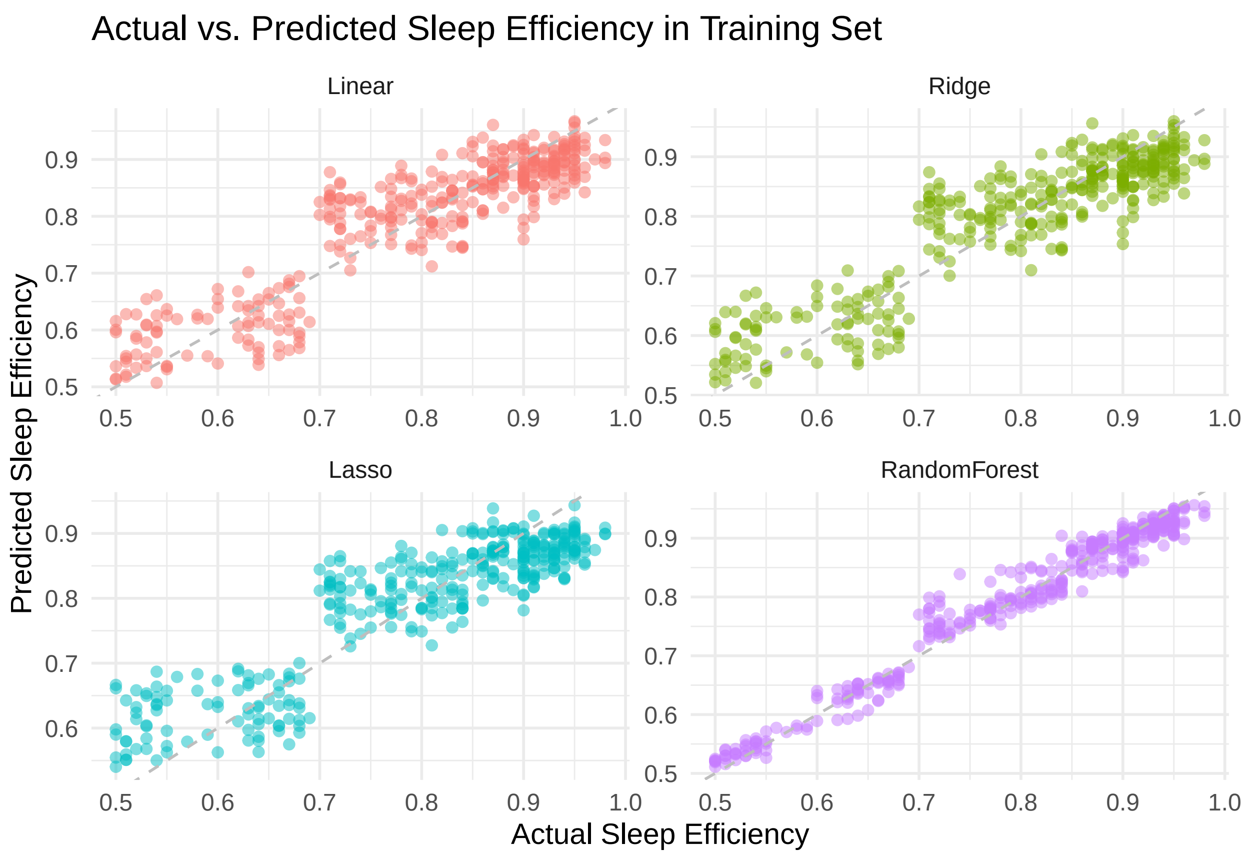
Linear: 76%, demonstrating consistency.

Ridge: 72%, indicating reasonable generalization.

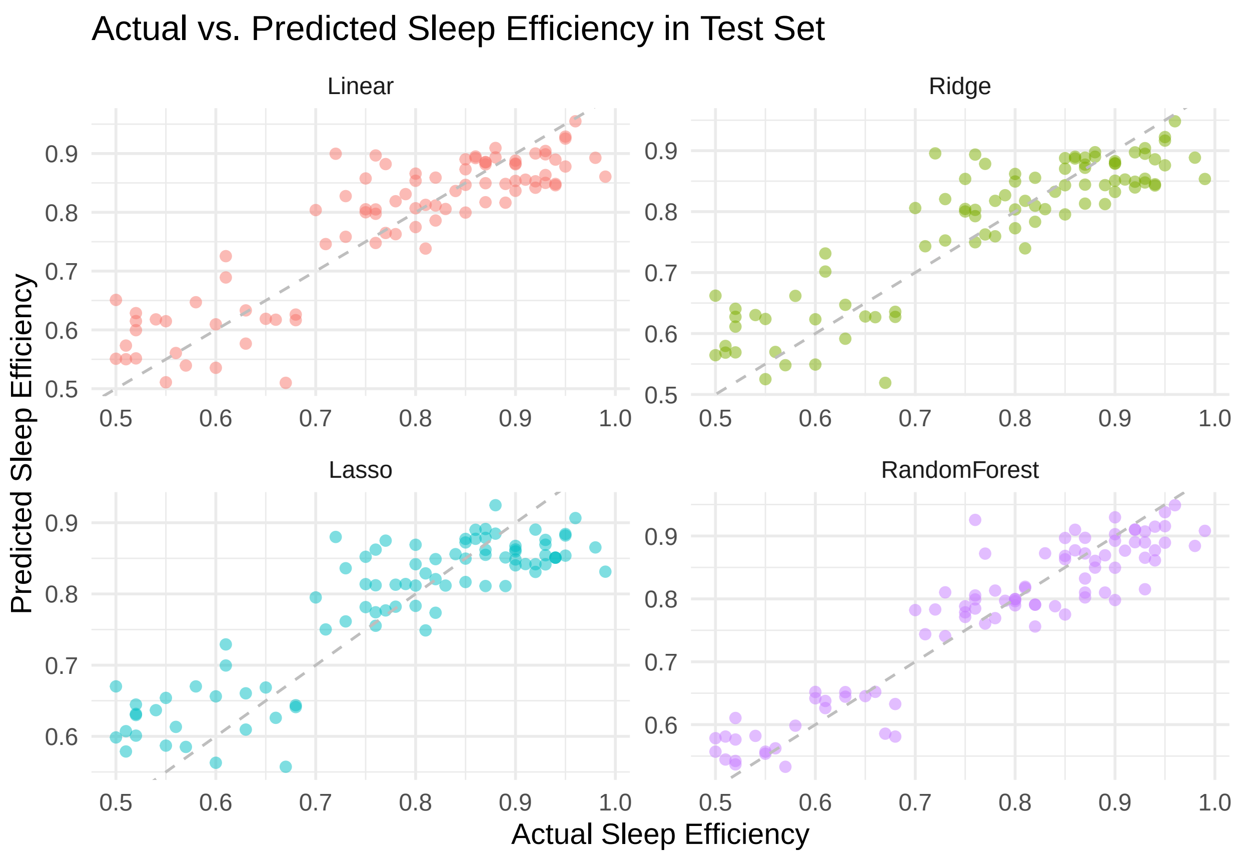
Lasso: 63%, suggesting potential inconsistencies.

11. Visialization of prediction

```r
# Data frame to store actual and predicted values
reds_df <- data.frame(
  Actual = data_tr$Sleep.efficiency,
  Linear = red1,
  Ridge = red2,
  Lasso = red3,
  RandomForest = red4
)

# Melt the data frame for plotting
reds_long <- reshape2::melt(reds_df, id = "Actual", variable.name = "Model", value.name = "Predicted")

# Create scatter plots comparing actual vs. predicted values for each model
ggplot(reds_long, aes(x = Actual, y = Predicted, color = Model)) +
  geom_point(alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
  facet_wrap(~ Model, scales = "free") +
  labs(title = "Actual vs. Predicted Sleep Efficiency in Training Set", x = "Actual Sleep Efficiency", y
  theme_minimal() +
  theme(legend.position = "none")
```




```r
# Data frame to store actual and predicted values
predictions_df <- data.frame(
  Actual = data_te$Sleep.efficiency,
  Linear = prediction1,
  Ridge = prediction2,
  Lasso = prediction3,
  RandomForest = prediction4
)

# Melt the data frame for plotting
predictions_long <- reshape2::melt(predictions_df, id = "Actual", variable.name = "Model", value.name =

# Create scatter plots comparing actual vs. predicted values for each model
ggplot(predictions_long, aes(x = Actual, y = Predicted, color = Model)) +
  geom_point(alpha = 0.5) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gray") +
  facet_wrap(~ Model, scales = "free") +
  labs(title = "Actual vs. Predicted Sleep Efficiency in Test Set", x = "Actual Sleep Efficiency", y =
  theme_minimal() +
  theme(legend.position = "none")
```



Conclusion:

Random Forest: Demonstrates the strongest performance in both training and test sets, with low MSE and high Adjusted R-squared.

Linear Model: Shows good performance, balancing fit and simplicity.

Ridge: Offers a balanced alternative, handling multicollinearity.

Lasso: Struggles with consistency and may over-regularize, leading to higher errors.