Final Project - Group 2

Sam, Eva, Dries, Caden, Jonas

2024-12-12

```r
suppressMessages({
  library(ggplot2)
  library(reshape2)
  library(dplyr)
  library(faraway)
  library(lubridate)
})
```

```
## Warning in check_dep_version(): ABI version mismatch:
## lme4 was built with Matrix ABI version 1
## Current Matrix ABI version is 0
## Please re-install lme4 from source or restore original 'Matrix' package
```

```
## Warning: package 'lubridate' was built under R version 4.3.3
```

1 - Exploratory Data Analysis

1.11 - Testing WHOOP's Activity Strain Metric - LINEAR REGRESSION

```r
workouts <- read.csv("data/workouts.csv")

workouts_1 <- workouts %>%
         select(-`GPS.enabled`,
         -`Distance..meters.`,
         -`Altitude.gain..meters.`,
         -`Altitude.change..meters.`,
         - `Cycle.start.time`,
         - `Cycle.end.time`,
         - `Cycle.timezone`,
         - `Workout.start.time`,
         - `Workout.end.time`,
         - `Activity.name`)

activity_strain <- lm(Activity.Strain ~ ., data = workouts_1)
sumary(activity_strain)
```

```
##                        Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)         -1.95728417  0.75131850 -2.6051  0.009298
## Duration..min.      -0.00303960  0.00127355 -2.3867  0.017154
## Energy.burned..cal.  0.01237841  0.00023213 53.3260 < 2.2e-16
## Max.HR..bpm.         0.01801828  0.00226009  7.9724 3.603e-15
## Average.HR..bpm.     0.04057875  0.00787245  5.1545 2.972e-07
## HR.Zone.1..          -0.00065590  0.00307877 -0.2130  0.831332
## HR.Zone.2..          -0.01182269  0.00398742 -2.9650  0.003087
## HR.Zone.3..          -0.00244303  0.00560155 -0.4361  0.662817
## HR.Zone.4..           0.01077301  0.00720889  1.4944  0.135333
## HR.Zone.5..           0.02787134  0.01163991  2.3945  0.016798
##
## n = 1209, p = 10, Residual SE = 0.69567, R-Squared = 0.95
```

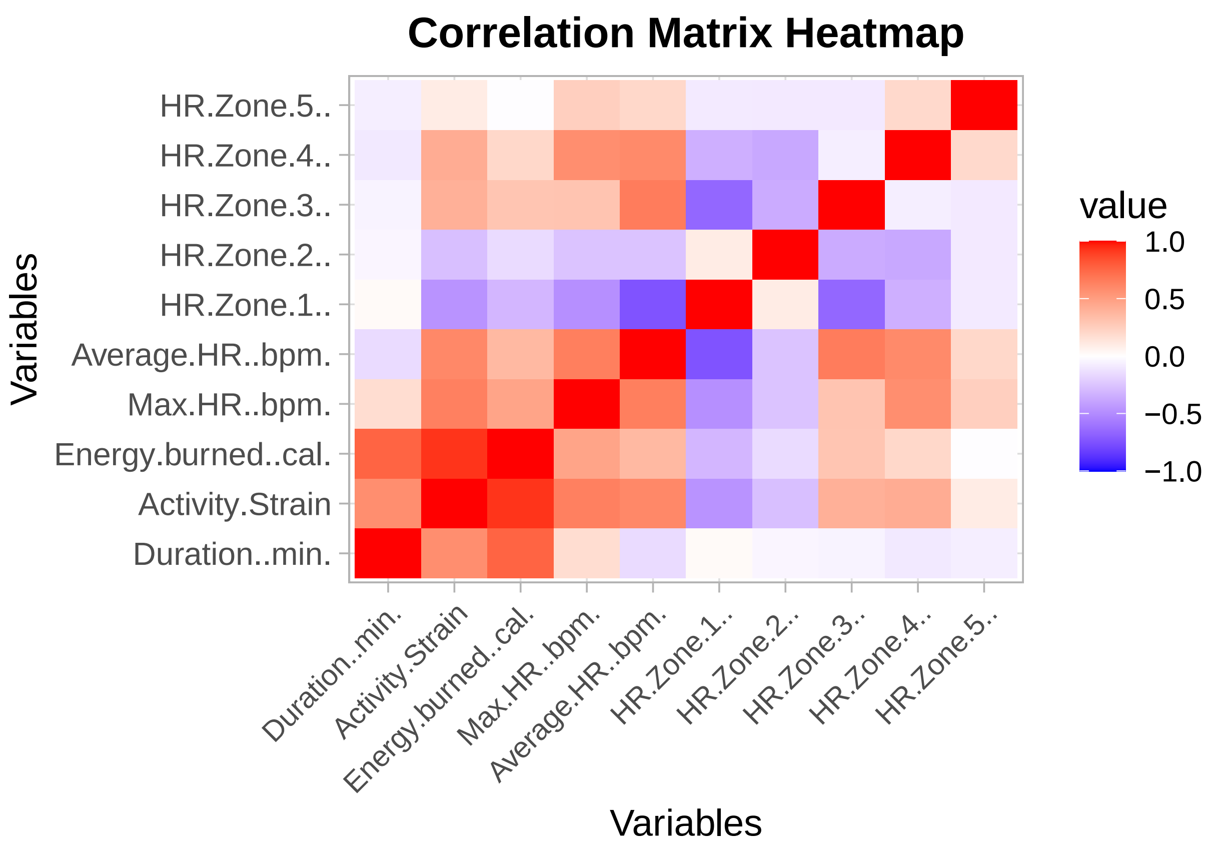
```
summary(activity_strain)$adj.r.squared
```

```
## [1] 0.9541699
```

Conclusion: 95% of the variability in `Activity Strain` is explained by our reduced dataset and a very simple model. This may gives us reason to drop this variable as well.

1.12 - Testing WHOOP's Activity Strain Metric - CORRELATION PLOT

```
cor_matrix <- cor(workouts_1, use = "complete.obs")
melted_cor_matrix <- melt(cor_matrix)

ggplot(melted_cor_matrix, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradientn(colors = c("blue", "white", "red"), values = c(0, 0.5, 1), limit = c(-1, 1)) +
  theme_light(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_text(size = 12)) +
  labs(title = "Correlation Matrix Heatmap", x = "Variables", y = "Variables") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))
```



```r
threshold <- 0.85
high_corr_pairs <- which(abs(cor_matrix) > threshold, arr.ind = TRUE)
high_corr_df <- data.frame(
  Column1 = rownames(cor_matrix)[high_corr_pairs[, 1]],
  Column2 = colnames(cor_matrix)[high_corr_pairs[, 2]],
  Correlation = cor_matrix[high_corr_pairs]
)
high_corr_df <- high_corr_df[high_corr_df$Column1 != high_corr_df$Column2, ]
print(high_corr_df)
```

```
##               Column1            Column2 Correlation
## 3 Energy.burned..cal.    Activity.Strain   0.9256782
## 4     Activity.Strain Energy.burned..cal.   0.9256782
```

Conclusion: to avoid collinearity between `Energy Burned` and `Activity Strain`, we will drop the `Activity Strain` column since `Energy Burned` provides an empirical datapoint.

Conclusion: to avoid collinearity between `Energy Burned` and `Activity Strain`, we will drop the `Activity Strain` column since `Energy Burned` provides an empirical datapoint.

---

2 - Data Prep, Preparing the Explanatory Variables (Workouts)

2.1 - daily__workouts

```r
##note: premise of time series = One Day One Observation

# Step 1: Drop the columns with no data
daily_workouts <- workouts %>%
  select(-c("GPS.enabled", "Distance..meters.",
            "Altitude.gain..meters.", "Altitude.change..meters."))

# Step 2: Remove the useless or redundant data
daily_workouts <- daily_workouts %>%
  select(-c("Cycle.start.time", "Cycle.end.time", "Cycle.timezone"))

# Step 3: Convert Activity.name to a factor
# Note: There was a level in the factor called "Other" with 1 observation,
# which was removed for clarity, since "Other" is no interpretable.
# Note: We also removed the activity "Activity" also for lack of interpretability.
# This activity had 65 observations.
daily_workouts$Activity.name <- as.factor(daily_workouts$Activity.name)
daily_workouts <- daily_workouts %>%
  mutate(
    Assault_Bike = Activity.name == "Assault Bike",
    Cycling = Activity.name == "Cycling",
    Elliptical = Activity.name == "Elliptical",
    Functional_Fitness = Activity.name == "Functional Fitness",
    Hiking_Rucking = Activity.name == "Hiking/Rucking",
    Powerlifting = Activity.name == "Powerlifting",
    Rowing = Activity.name == "Rowing",
    Running = Activity.name == "Running",
    Spin = Activity.name == "Spin",
    Stairmaster = Activity.name == "Stairmaster",
    Walking = Activity.name == "Walking",
    Weightlifting = Activity.name == "Weightlifting",
    Yoga = Activity.name == "Yoga"
  ) %>%
  select(-Activity.name)  # Remove the original Activity.name column

# Step 4: Convert 'Workout.start.time' to Date and extract the date portion
daily_workouts <- daily_workouts %>%
  mutate(Workout.start.date = as.Date(Workout.start.time, format = "%Y-%m-%d %H:%M:%S"))

# Step 5: Group by date and summarize
daily_workouts <- daily_workouts %>%
  group_by(Workout.start.date) %>%
  summarise(
    # Sum of 'Duration..min.' and 'Energy.burned..cal.'
    Total.Duration.min = sum(`Duration..min.`, na.rm = TRUE),
    Total.Energy.burned.cal = sum(`Energy.burned..cal.`, na.rm = TRUE),

    # Weighted average time spent at average heart rate
    Weighted.Avg.HR = sum(`Average.HR..bpm.` * `Duration..min.`, na.rm = TRUE) /
      sum(`Duration..min.`, na.rm = TRUE),

    # Weighted average time spent for heart rate zones
    Weighted.Avg.HR.Zone.1 = sum(`HR.Zone.1..` * `Duration..min.`, na.rm = TRUE) /
      sum(`Duration..min.`, na.rm = TRUE),
    Weighted.Avg.HR.Zone.2 = sum(`HR.Zone.2..` * `Duration..min.`, na.rm = TRUE) /
      sum(`Duration..min.`, na.rm = TRUE),
    Weighted.Avg.HR.Zone.3 = sum(`HR.Zone.3..` * `Duration..min.`, na.rm = TRUE) /
      sum(`Duration..min.`, na.rm = TRUE),
    Weighted.Avg.HR.Zone.4 = sum(`HR.Zone.4..` * `Duration..min.`, na.rm = TRUE) /
      sum(`Duration..min.`, na.rm = TRUE),
    Weighted.Avg.HR.Zone.5 = sum(`HR.Zone.5..` * `Duration..min.`, na.rm = TRUE) /
      sum(`Duration..min.`, na.rm = TRUE),

    # Keep the Max HR for the day
    Max.HR.bpm = max(`Max.HR..bpm.`, na.rm = TRUE),

    # Sum of boolean values for each activity (to count the occurrences per day)
    Total.Assault_Bike = sum(Assault_Bike, na.rm = TRUE),
    Total.Cycling = sum(Cycling, na.rm = TRUE),
    Total.Elliptical = sum(Elliptical, na.rm = TRUE),
    Total.Functional_Fitness = sum(Functional_Fitness, na.rm = TRUE),
    Total.Hiking_Rucking = sum(Hiking_Rucking, na.rm = TRUE),
    Total.Powerlifting = sum(Powerlifting, na.rm = TRUE),
    Total.Rowing = sum(Rowing, na.rm = TRUE),
    Total.Running = sum(Running, na.rm = TRUE),
    Total.Spin = sum(Spin, na.rm = TRUE),
    Total.Stairmaster = sum(Stairmaster, na.rm = TRUE),
    Total.Walking = sum(Walking, na.rm = TRUE),
    Total.Weightlifting = sum(Weightlifting, na.rm = TRUE),
    Total.Yoga = sum(Yoga, na.rm = TRUE)) %>%

  #Create a new column for heart zone 0
  mutate(Weighted.Avg.HR.Zone.0 = round((100 - Weighted.Avg.HR.Zone.1 - Weighted.Avg.HR.Zone.2 -
           Weighted.Avg.HR.Zone.3 - Weighted.Avg.HR.Zone.4 - Weighted.Avg.HR.Zone.5), 2))
```

2.2 - missing_dates_df

```
# Step 1: Identify the date range (min and max dates)
date_range <- seq.Date(min(daily_workouts$Workout.start.date),
                       max(daily_workouts$Workout.start.date), by = "day")

# Step 2: Create a data frame for all dates in the range
missing_dates_df <- data.frame(
  Workout.start.date = date_range,
  Total.Duration.min = 0,
  Total.Energy.burned.cal = 0,
  Weighted.Avg.HR = 57,  # Set resting heart rate
  Weighted.Avg.HR.Zone.1 = 0,
  Weighted.Avg.HR.Zone.2 = 0,
  Weighted.Avg.HR.Zone.3 = 0,
  Weighted.Avg.HR.Zone.4 = 0,
  Weighted.Avg.HR.Zone.5 = 0,
  Max.HR.bpm = 57,  # Set resting heart rate for Max HR
  Total.Assault_Bike = 0,
  Total.Cycling = 0,
  Total.Elliptical = 0,
  Total.Functional_Fitness = 0,
  Total.Hiking_Rucking = 0,
  Total.Powerlifting = 0,
  Total.Rowing = 0,
  Total.Running = 0,
  Total.Spin = 0,
  Total.Stairmaster = 0,
  Total.Walking = 0,
  Total.Weightlifting = 0,
  Total.Yoga = 0,
  Weighted.Avg.HR.Zone.0 = 100  # Set HR Zone 0 to 100
)
```

2.3 - Combining daily_workouts and missing_dates_df into One Continuous Exploratory Dataset

```
# Step 3: Combine with the original data
daily_workouts <- bind_rows(
  daily_workouts,
  missing_dates_df
)

# Step 4: Remove any duplicates and sort by date
daily_workouts <- daily_workouts %>%
  distinct(Workout.start.date, .keep_all = TRUE) %>%
  arrange(Workout.start.date)
```

Columns in wrangled workouts dataset:

- Workout.start.date
- Total.Duration.min
- Total.Energy.burned.cal
- Weighted.Avg.HR
- Weighted.Avg.HR.Zone.0
- Weighted.Avg.HR.Zone.1
- Weighted.Avg.HR.Zone.2
- Weighted.Avg.HR.Zone.3
- Weighted.Avg.HR.Zone.4
- Weighted.Avg.HR.Zone.5
- Max.HR.bpm
- Total.Assault_Bike
- Total.Cycling
- Total.Elliptical
- Total.Functional_Fitness
- Total.Hiking_Rucking
- Total.Powerlifting
- Total.Rowing
- Total.Running
- Total.Spin
- Total.Stairmaster
- Total.Walking
- Total.Weightlifting
- Total.Yoga

---

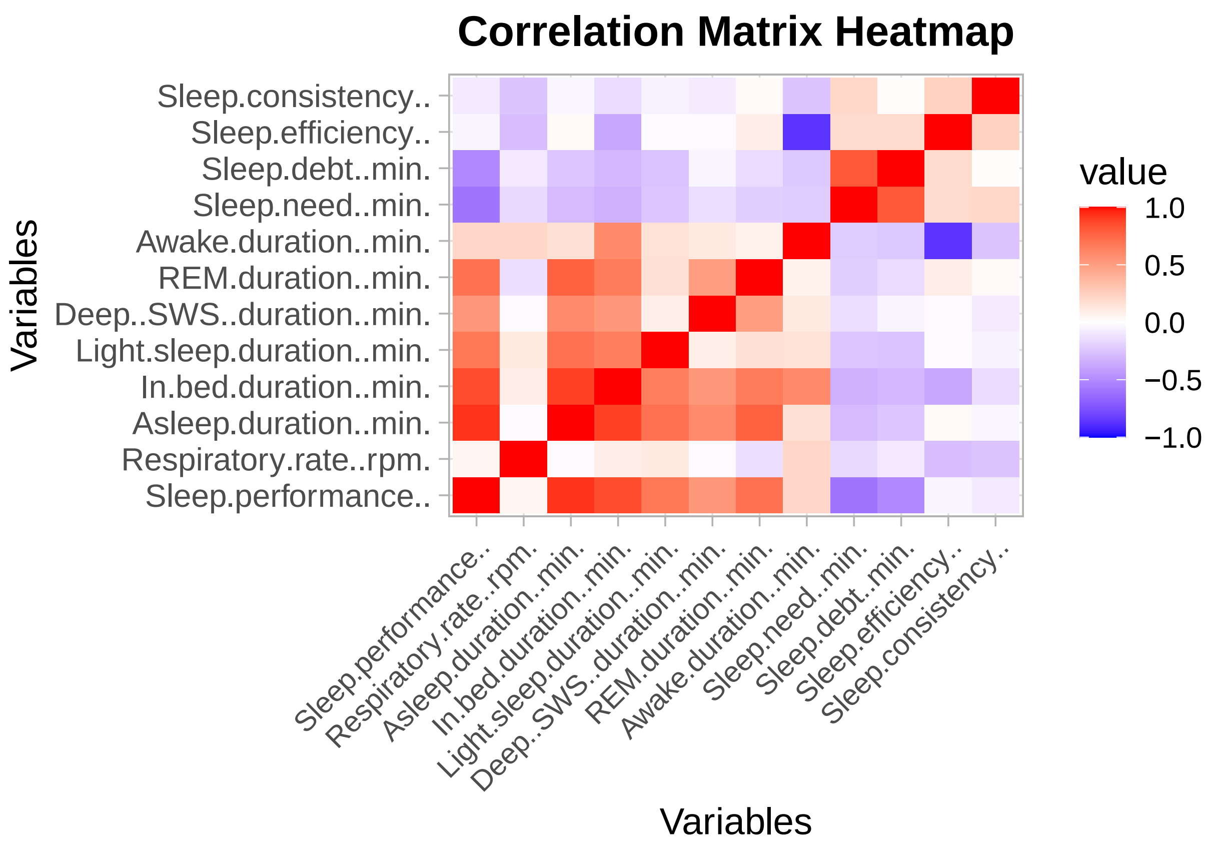
3 - Data Prep, Preparing the Response Variable: Sleep

Columns in sleep dataset:

- Cycle.start.time
- Cycle.end.time
- Cycle.timezone
- Sleep.onset
- Wake.onset
- Sleep.performance..
- Respiratory.rate..rpm.
- Asleep.duration..min.
- In.bed.duration..min.
- Light.sleep.duration..min.
- Deep..SWS..duration..min.
- REM.duration..min.
- Awake.duration..min.
- Sleep.need..min.
- Sleep.debt..min.
- Sleep.efficiency..
- Sleep.consistency..
- Nap

3.1 - Sleep CORRELATION PLOT

```r
# Load necessary libraries
library(ggplot2)
library(reshape2)

# Read the CSV file
sleeps <- read.csv("data/sleeps.csv")

# Select only numeric columns
sleeps_numeric <- sleeps[sapply(sleeps, is.numeric)]

# Compute the correlation matrix using complete observations
cor_matrix2 <- cor(sleeps_numeric, use = "complete.obs")

# Melt the correlation matrix for visualization
melted_cor_matrix2 <- melt(cor_matrix2)

# Create the heatmap using ggplot2
ggplot(melted_cor_matrix2, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradientn(colors = c("blue", "white", "red"), values = c(0, 0.5, 1), limit = c(-1, 1)) +
  theme_light(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_text(size = 12)) +
  labs(title = "Correlation Matrix Heatmap", x = "Variables", y = "Variables") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))
```



```r
# Define the correlation threshold
threshold <- 0.85

# Find pairs of variables with correlations exceeding the threshold
high_corr_pairs2 <- which(abs(cor_matrix2) > threshold, arr.ind = TRUE)

# Create a data frame of the high-correlation pairs
high_corr_df2 <- data.frame(
  Column1 = rownames(cor_matrix2)[high_corr_pairs2[, 1]],
  Column2 = colnames(cor_matrix2)[high_corr_pairs2[, 2]],
  Correlation = cor_matrix2[high_corr_pairs2]
)

# Remove self-correlations (diagonal elements)
high_corr_df2 <- high_corr_df2[high_corr_df2$Column1 != high_corr_df2$Column2, ]

# Print the high-correlation pairs
print(high_corr_df2)
```

```
##                  Column1                Column2 Correlation
## 2  Asleep.duration..min.    Sleep.performance..   0.9266651
## 3  In.bed.duration..min.    Sleep.performance..   0.8528991
## 5    Sleep.performance.. Asleep.duration..min.   0.9266651
## 7  In.bed.duration..min. Asleep.duration..min.   0.8900266
## 8    Sleep.performance.. In.bed.duration..min.   0.8528991
## 9  Asleep.duration..min. In.bed.duration..min.   0.8900266
## 15     Sleep.efficiency..  Awake.duration..min.  -0.8693928
## 18  Awake.duration..min.     Sleep.efficiency..  -0.8693928
```

```r
#Step 1: remove unnecessary or redundant variables
daily_sleep <- sleeps %>%
  select(- `Cycle.start.time`,
         - `Cycle.end.time`,
         - `Cycle.timezone`,
         - `Sleep.performance..`,
         - `In.bed.duration..min.`,
         - `Sleep.debt..min.`,
         - `Sleep.efficiency..`,
         - `Nap`)

# Step 2: Group by Wake.date and calculate the number of wake-ups and total sleep time
nap_data <- daily_sleep %>%
  mutate(Wake.date = as.Date(Wake.onset)) %>%
  group_by(Wake.date) %>%
  summarise(
    Nap.count = n_distinct(Wake.onset) - 1,
    Nap.duration = sum(Asleep.duration..min., na.rm = TRUE) - max(Asleep.duration..min.)
  ) %>%
  ungroup() %>%
  mutate(Nap.duration = ifelse(Nap.count == 0, 0, Nap.duration))

# Step 3: Group by Wake.onset date and select the row with the largest Asleep.duration..min.
sleep_data <- daily_sleep %>%
  # Group by Wake.onset
  group_by(as.Date(Wake.onset)) %>%
  # Filter to keep the row with the max Asleep.duration..min.
  filter(Asleep.duration..min. == max(Asleep.duration..min.)) %>%
  slice(1) %>% ungroup()

#Step 4: We will think of your sleep as the day that you worked out
sleep_data <- sleep_data %>%
  mutate(Sleep.date = as.Date(Wake.onset) - days(1))

# Step 5: Merge sleep_data and nap_data by their respective date columns
sleep_nap_data <- sleep_data %>%
  left_join(nap_data, by = c("Sleep.date" = "Wake.date")) %>%
  filter(!is.na(Nap.count))
```

4 - Final Dataset, Merge Sleep and Workout Data

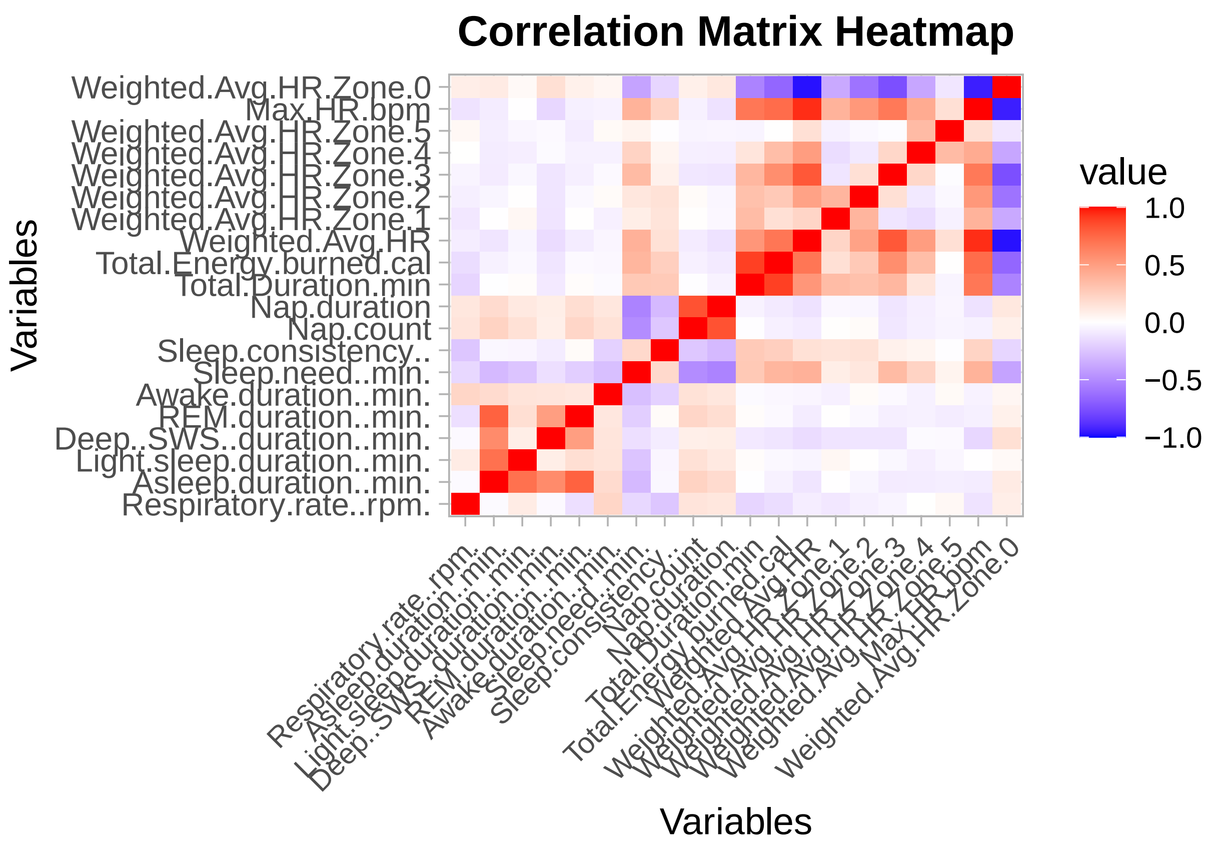
```r
merged_data <- sleep_nap_data %>%
  inner_join(daily_workouts, by = c("Sleep.date" = "Workout.start.date")) %>%
  select(-`Sleep.onset`,
         - `Wake.onset`,
         - `as.Date(Wake.onset)`)

str(merged_data)
```

```
## tibble [673 x 34] (S3: tbl_df/tbl/data.frame)
##  $ Respiratory.rate..rpm.    : num [1:673] 13.8 14.3 14.1 13.6 14.3 13.9 14.8 14.2 14.4 13.9 ...
##  $ Asleep.duration..min.     : int [1:673] 444 422 286 321 369 378 502 428 412 536 ...
##  $ Light.sleep.duration..min.: int [1:673] 172 175 75 81 78 121 153 125 188 234 ...
##  $ Deep..SWS..duration..min. : int [1:673] 93 52 81 102 83 91 101 88 93 90 ...
##  $ REM.duration..min.        : int [1:673] 179 195 130 138 208 166 248 215 131 212 ...
##  $ Awake.duration..min.      : int [1:673] 44 78 55 44 93 150 126 31 52 34 ...
##  $ Sleep.need..min.          : int [1:673] 546 545 512 606 611 609 542 488 450 474 ...
##  $ Sleep.consistency..       : int [1:673] NA NA 67 78 66 74 58 60 61 46 ...
##  $ Sleep.date                : Date[1:673], format: "2022-12-11" "2022-12-12" ...
##  $ Nap.count                 : num [1:673] 0 0 1 0 0 0 1 0 1 1 ...
##  $ Nap.duration              : num [1:673] 0 0 37 0 0 0 58 0 84 38 ...
##  $ Total.Duration.min        : num [1:673] 78 113 41 130 82 243 131 0 206 281 ...
##  $ Total.Energy.burned.cal   : num [1:673] 416 483 376 722 621 ...
##  $ Weighted.Avg.HR           : num [1:673] 129 119 155 129 145 ...
##  $ Weighted.Avg.HR.Zone.1    : num [1:673] 26.13 18.48 0 22.86 0.22 ...
##  $ Weighted.Avg.HR.Zone.2    : num [1:673] 28.4 53.3 2 34.8 26.8 ...
##  $ Weighted.Avg.HR.Zone.3    : num [1:673] 37.8 18.3 29 35.1 53.6 ...
##  $ Weighted.Avg.HR.Zone.4    : num [1:673] 0.769 0 66 4.377 19.463 ...
##  $ Weighted.Avg.HR.Zone.5    : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Max.HR.bpm                : num [1:673] 166 162 171 181 183 177 176 57 158 163 ...
##  $ Total.Assault_Bike        : num [1:673] 1 0 0 0 0 0 0 0 0 1 ...
##  $ Total.Cycling             : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Elliptical          : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Functional_Fitness  : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Hiking_Rucking      : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Powerlifting        : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Rowing              : num [1:673] 0 2 0 3 2 2 2 0 1 3 ...
##  $ Total.Running             : num [1:673] 1 0 1 1 0 0 0 0 0 0 ...
##  $ Total.Spin                : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Stairmaster         : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Walking             : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Total.Weightlifting       : num [1:673] 1 0 0 0 0 0 0 0 1 0 ...
##  $ Total.Yoga                : num [1:673] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Weighted.Avg.HR.Zone.0    : num [1:673] 6.9 9.93 3 2.84 0 ...
```

```r
# Load necessary libraries
library(dplyr)

# Select the specified variables from merged_data_1
merged_data_1 <- merged_data %>%
  select(
    -Sleep.date,
    -Total.Assault_Bike,
    -Total.Cycling,
    -Total.Elliptical,
    -Total.Functional_Fitness,
    -Total.Hiking_Rucking,
    -Total.Powerlifting,
    -Total.Rowing,
    -Total.Running,
    -Total.Spin,
    -Total.Stairmaster,
    -Total.Walking,
    -Total.Weightlifting,
    -Total.Yoga
  )
```

5 - Baseline Model

```
# Remove rows with missing values
cleaned_data <- na.omit(merged_data_1)

# Select only numeric columns
cleaned_numeric <- cleaned_data[sapply(cleaned_data, is.numeric)]

# Compute the correlation matrix using complete observations
cor_matrix3 <- cor(cleaned_numeric, use = "complete.obs")

# Melt the correlation matrix for visualization
melted_cor_matrix3 <- melt(cor_matrix3)

ggplot(melted_cor_matrix3, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradientn(colors = c("blue", "white", "red"), values = c(0, 0.5, 1), limit = c(-1, 1)) +
  theme_light(base_size = 14) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_text(size = 12)) +
  labs(title = "Correlation Matrix Heatmap", x = "Variables", y = "Variables") +
  theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))
```



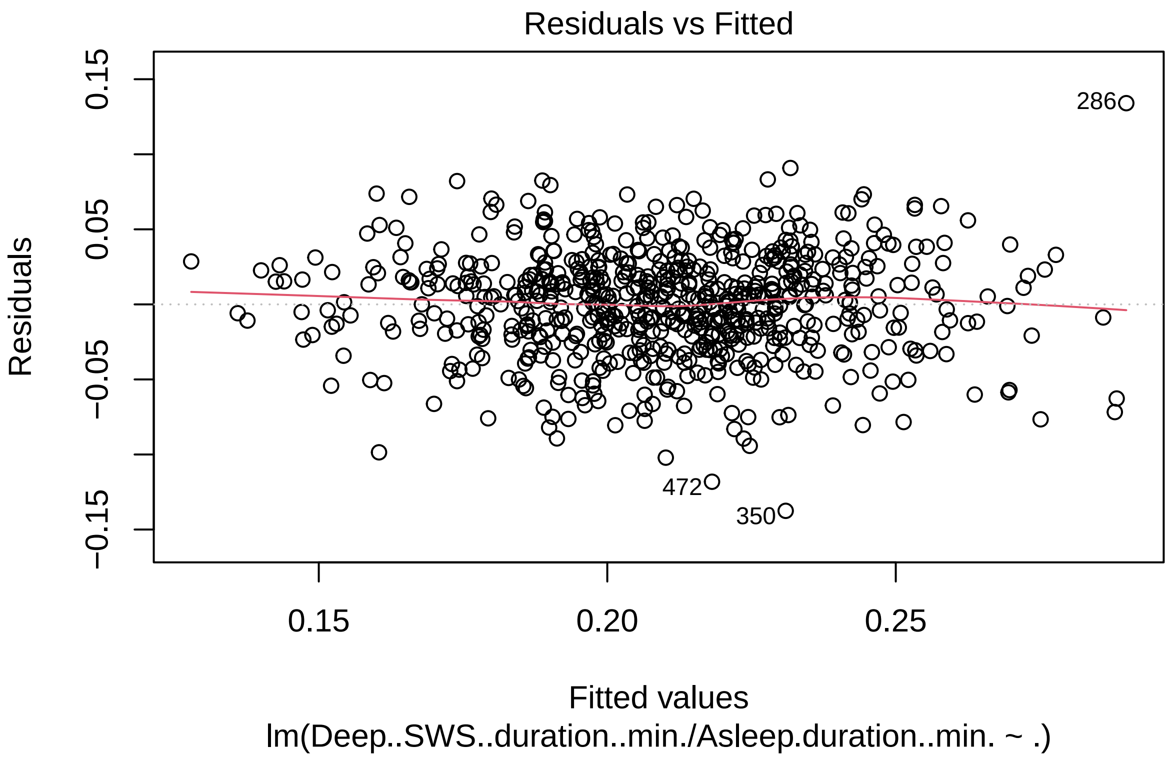
```
# Define the correlation threshold
threshold <- 0.85

high_corr_pairs3 <- which(abs(cor_matrix3) > threshold, arr.ind = TRUE)
high_corr_df3 <- data.frame(
  Column1 = rownames(cor_matrix3)[high_corr_pairs3[, 1]],
  Column2 = colnames(cor_matrix3)[high_corr_pairs3[, 2]],
  Correlation = cor_matrix3[high_corr_pairs3]
)
high_corr_df3 <- high_corr_df3[high_corr_df3$Column1 != high_corr_df3$Column2, ]

print(high_corr_df3)
```

```
##                    Column1                 Column2 Correlation
## 12 Total.Energy.burned.cal      Total.Duration.min   0.8937230
## 13      Total.Duration.min Total.Energy.burned.cal   0.8937230
## 16              Max.HR.bpm         Weighted.Avg.HR   0.9418197
## 17  Weighted.Avg.HR.Zone.0         Weighted.Avg.HR  -0.9730197
## 23         Weighted.Avg.HR              Max.HR.bpm   0.9418197
## 25  Weighted.Avg.HR.Zone.0              Max.HR.bpm  -0.9413541
## 26         Weighted.Avg.HR  Weighted.Avg.HR.Zone.0  -0.9730197
## 27              Max.HR.bpm  Weighted.Avg.HR.Zone.0  -0.9413541
```

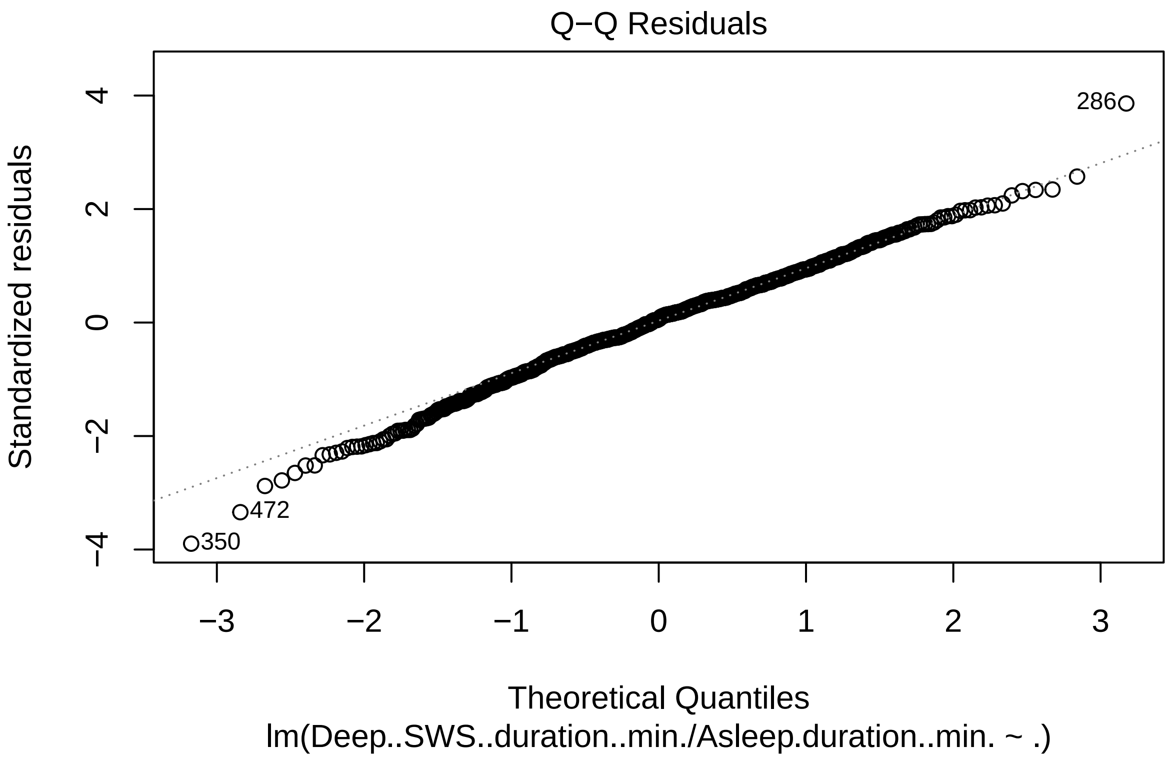
```
# Basic model, no interaction terms
model <- lm(Deep..SWS..duration..min./Asleep.duration..min. ~ ., data = cleaned_data)
summary(model)$adj.r.squared
```

```
## [1] 0.3470618
```

```
plot(model, 1)
```



```
plot(model, 2)
```

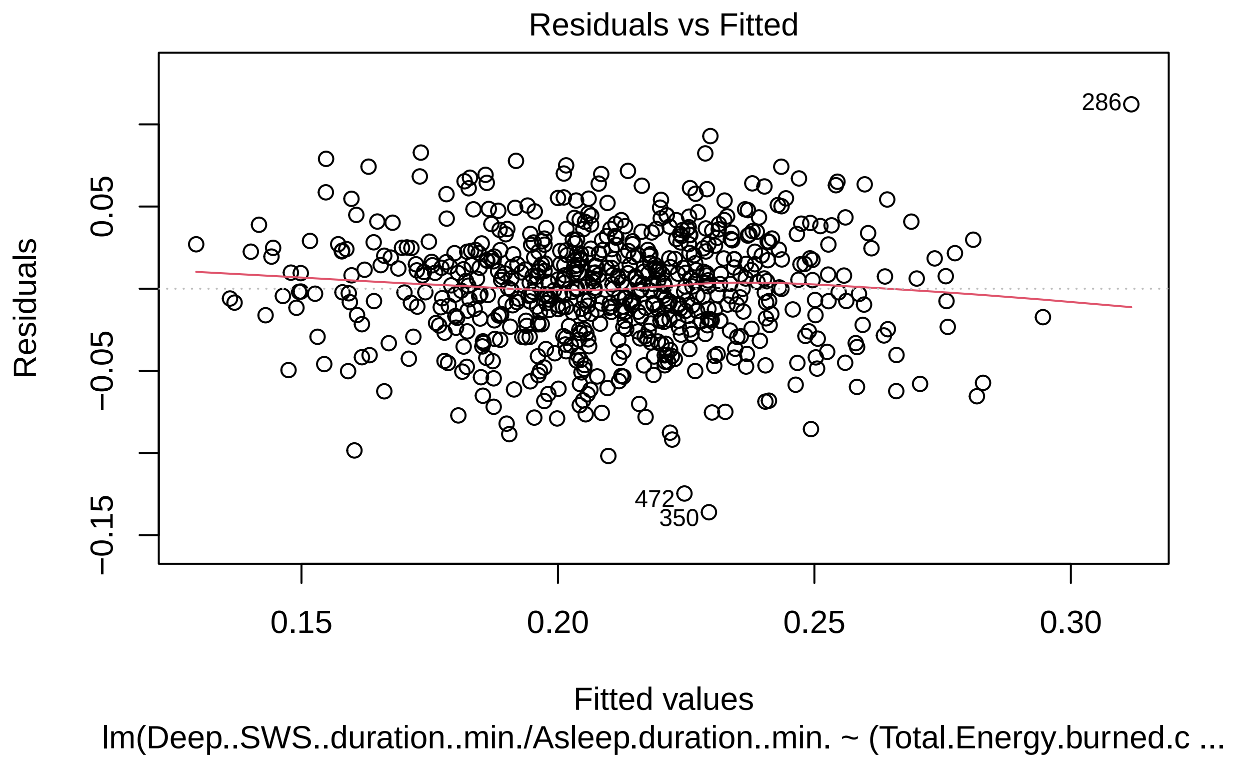


```
# Model with Interactive terms (note this actually has a stronger R^2 than the highly complex model)
model_rr <- lm(Deep..SWS..duration..min./Asleep.duration..min. ~ (Total.Energy.burned.cal)*(.), data =
```

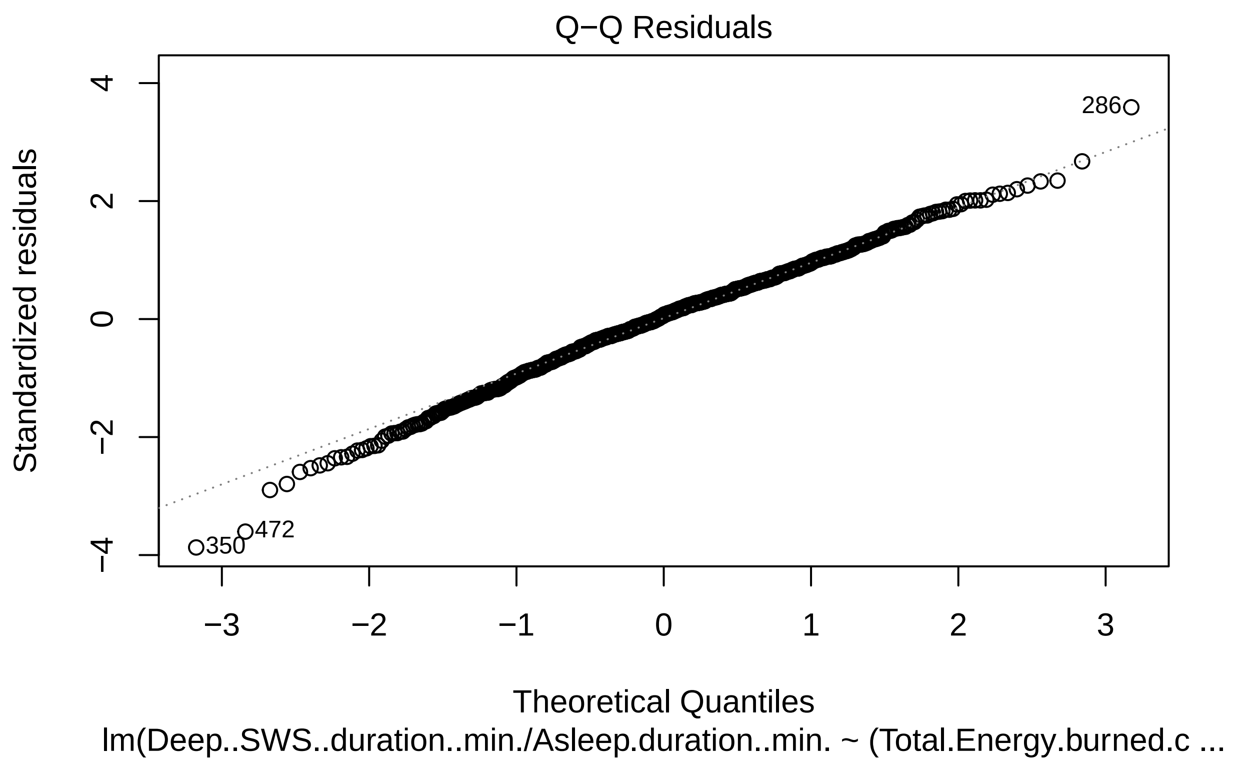
```r
summary(model_rr)$adj.r.squared
```

```
## [1] 0.3509511
```

```r
plot(model_rr, 1)
```



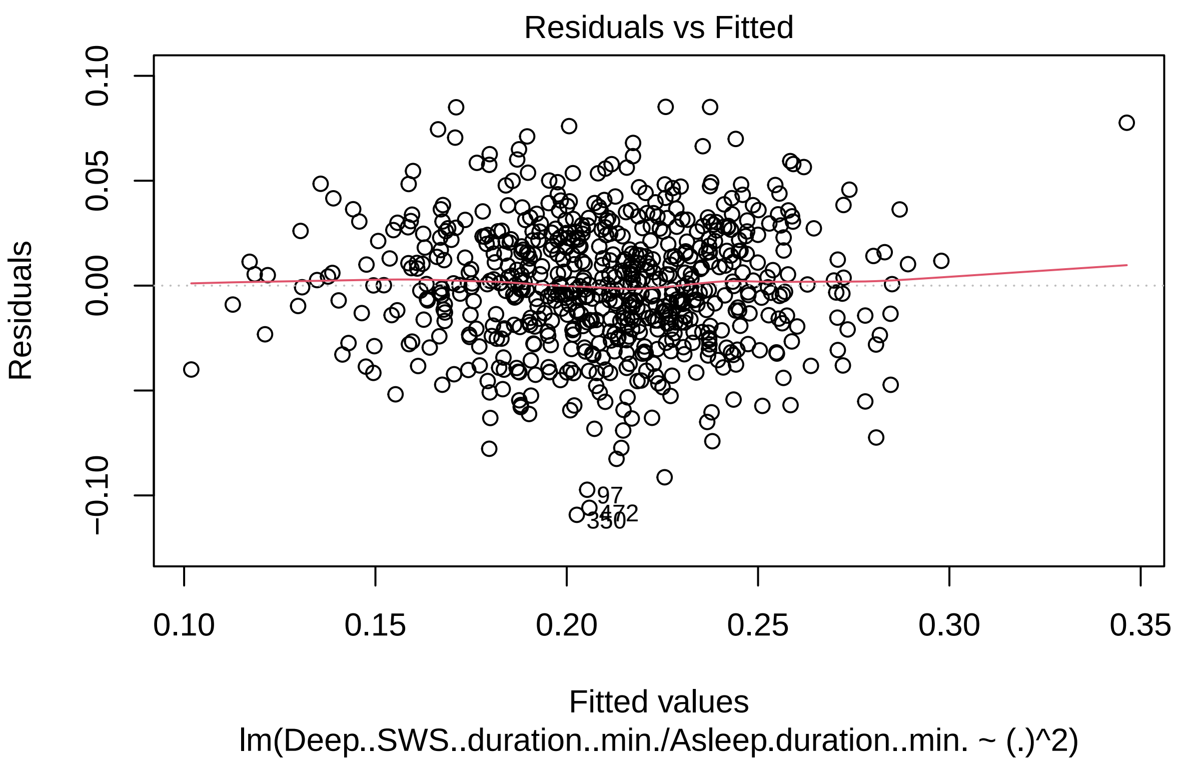
```r
plot(model_rr, 2)
```



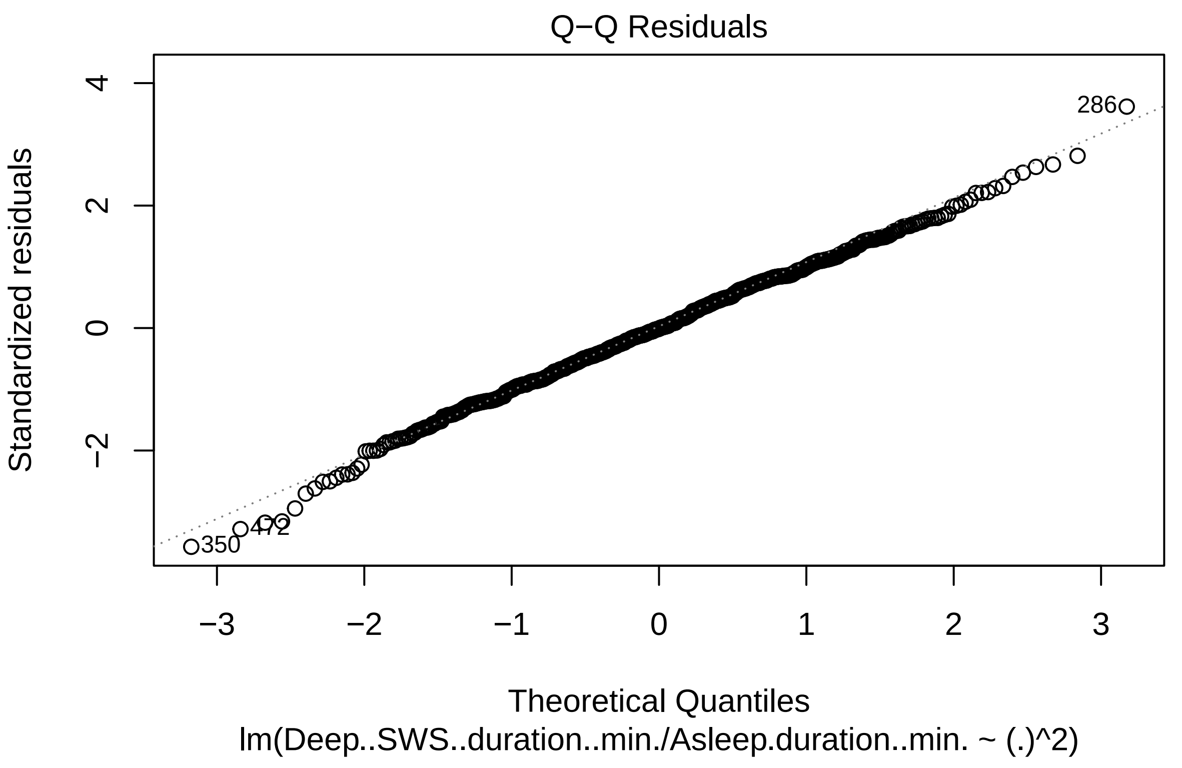
```r
# highly complex model
model_full <- lm(Deep..SWS..duration..min./Asleep.duration..min. ~ (.)^2, data = cleaned_data)
summary(model_full)$adj.r.squared
```

```
## [1] 0.3643356
```

```r
plot(model_full, 1)
```



```r
plot(model_full, 2)
```



6 - Model Selection

```
#Originally the model_full was not running on posit - we examined the dataset again and were able to ta

## Results with Interactive term were:
# Start:  AIC=-4415.95)

## Results with full model term were:
# Start:  AIC=-4295.39)


# Remove rows with missing values
cleaned_data <- na.omit(merged_data_1)

model_backward <- step(model_rr, direction = "backward")
# Intercept-only model with the cleaned data
model_intercept <- lm(Deep..SWS..duration..min./Asleep.duration..min. ~ 1, data = cleaned_data)
model_forward <- step(model_intercept, direction = "forward", scope = formula(model_rr))
```

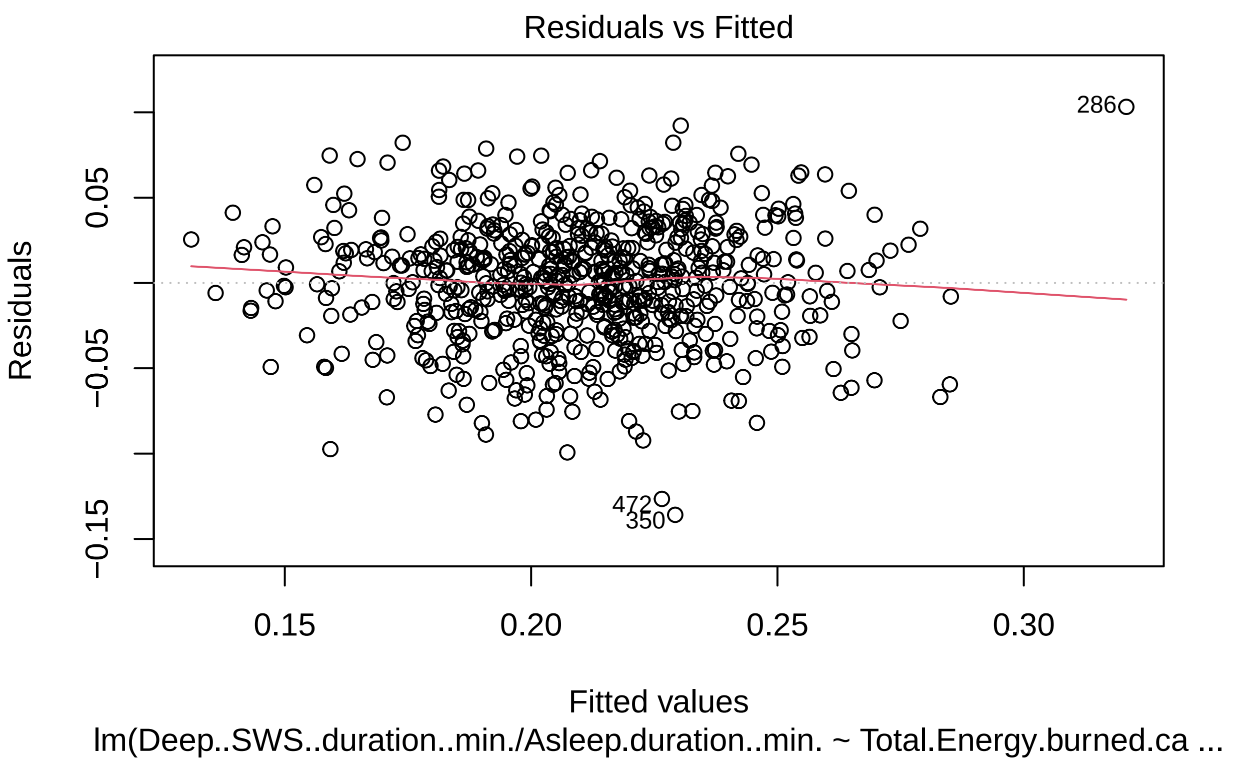
```
#Best Adj-R^2
formula(model_backward)
```

```
## Deep..SWS..duration..min./Asleep.duration..min. ~ Total.Energy.burned.cal +
##     Light.sleep.duration..min. + REM.duration..min. + Awake.duration..min. +
##     Sleep.need..min. + Sleep.consistency.. + Total.Duration.min +
##     Weighted.Avg.HR + Weighted.Avg.HR.Zone.1 + Weighted.Avg.HR.Zone.2 +
##     Weighted.Avg.HR.Zone.3 + Max.HR.bpm + Weighted.Avg.HR.Zone.0 +
##     Total.Energy.burned.cal:Total.Duration.min + Total.Energy.burned.cal:Weighted.Avg.HR +
##     Total.Energy.burned.cal:Weighted.Avg.HR.Zone.1 + Total.Energy.burned.cal:Weighted.Avg.HR.Zone.2
##     Total.Energy.burned.cal:Weighted.Avg.HR.Zone.3 + Total.Energy.burned.cal:Max.HR.bpm +
##     Total.Energy.burned.cal:Weighted.Avg.HR.Zone.0
```

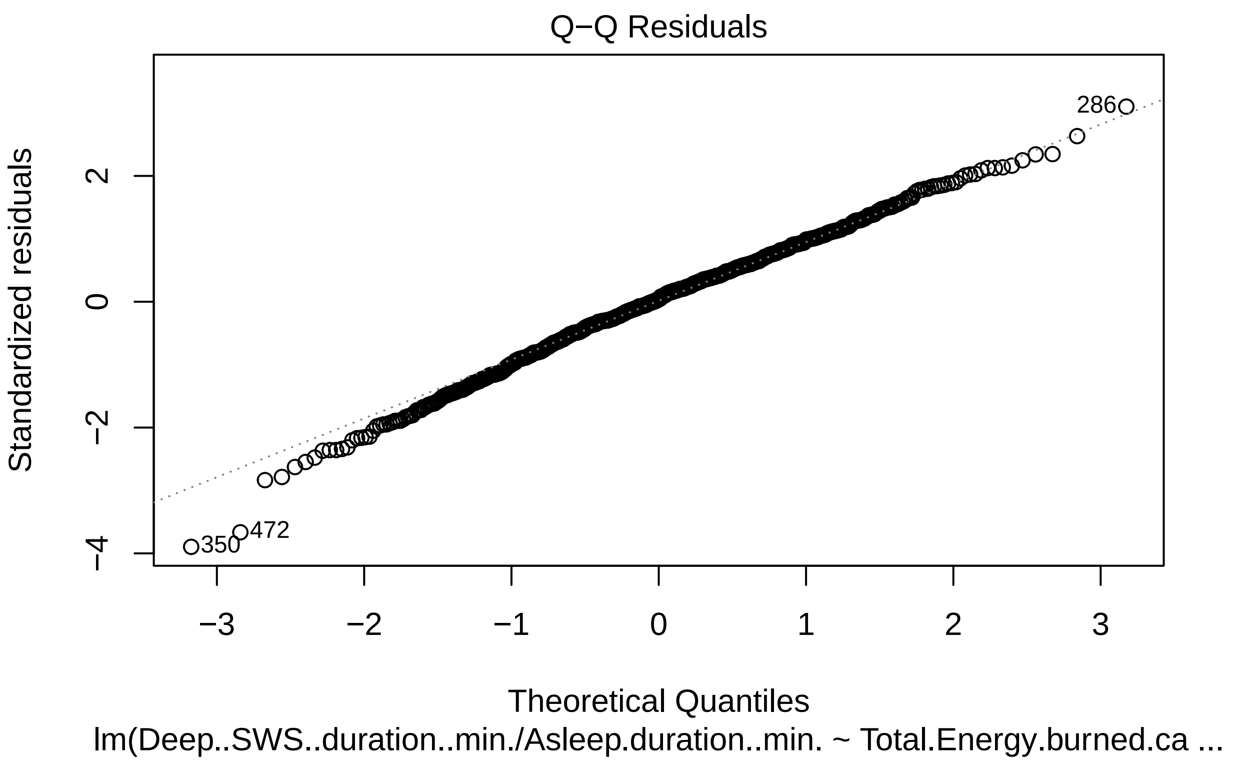
```
summary(model_backward)$adj.r.squared
```

```
## [1] 0.3626397
```

```
plot(model_backward, 1)
```



```r
plot(model_backward, 2)
```



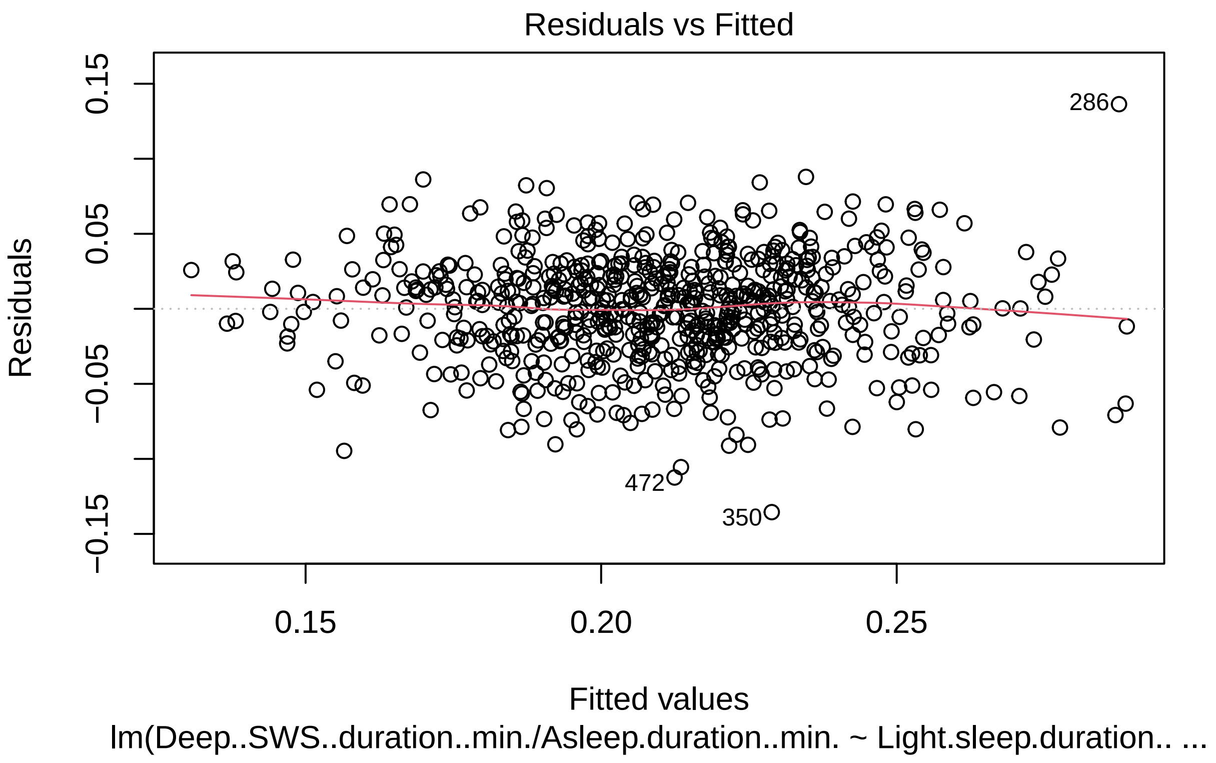
```r
formula(model_forward)
```

```
## Deep..SWS..duration..min./Asleep.duration..min. ~ Light.sleep.duration..min. +
##     Max.HR.bpm + Sleep.consistency.. + Weighted.Avg.HR.Zone.4 +
##     Awake.duration..min. + REM.duration..min.
```

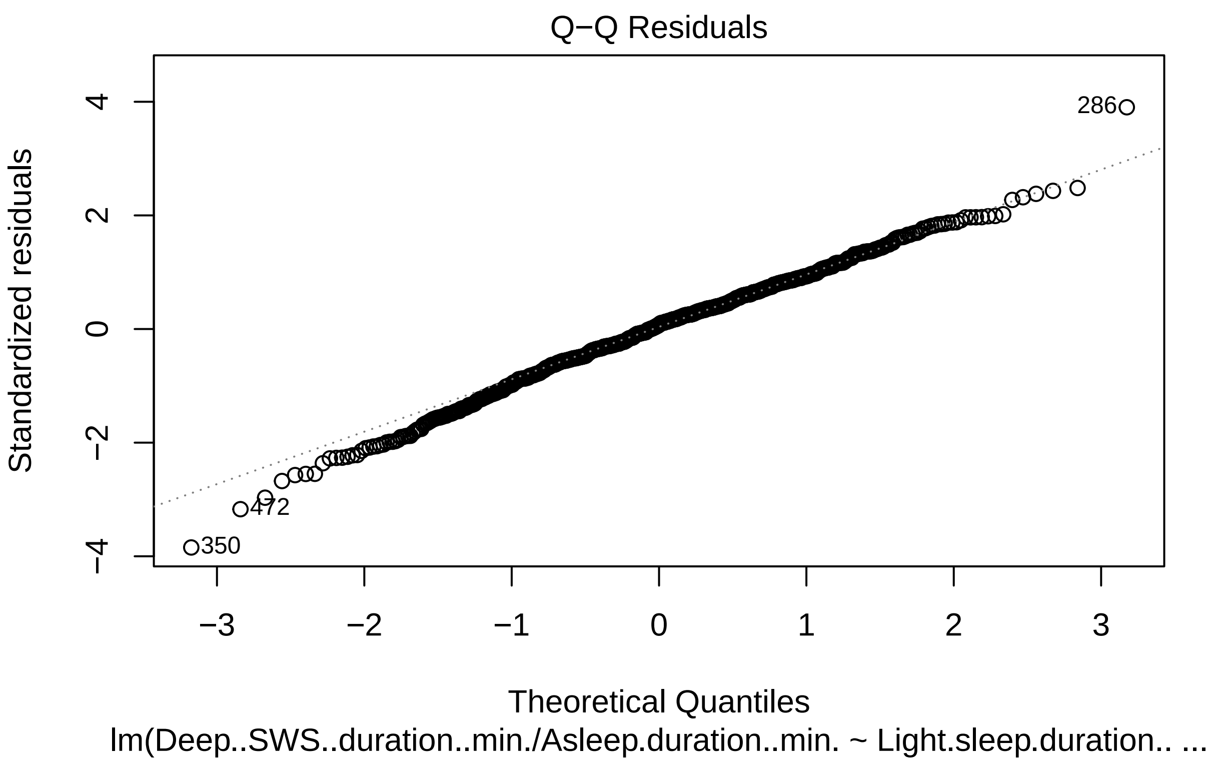
```
summary(model_forward)$adj.r.squared
```

```
## [1] 0.3525724
```

```
plot(model_forward, 1)
```



```
plot(model_forward, 2)
```

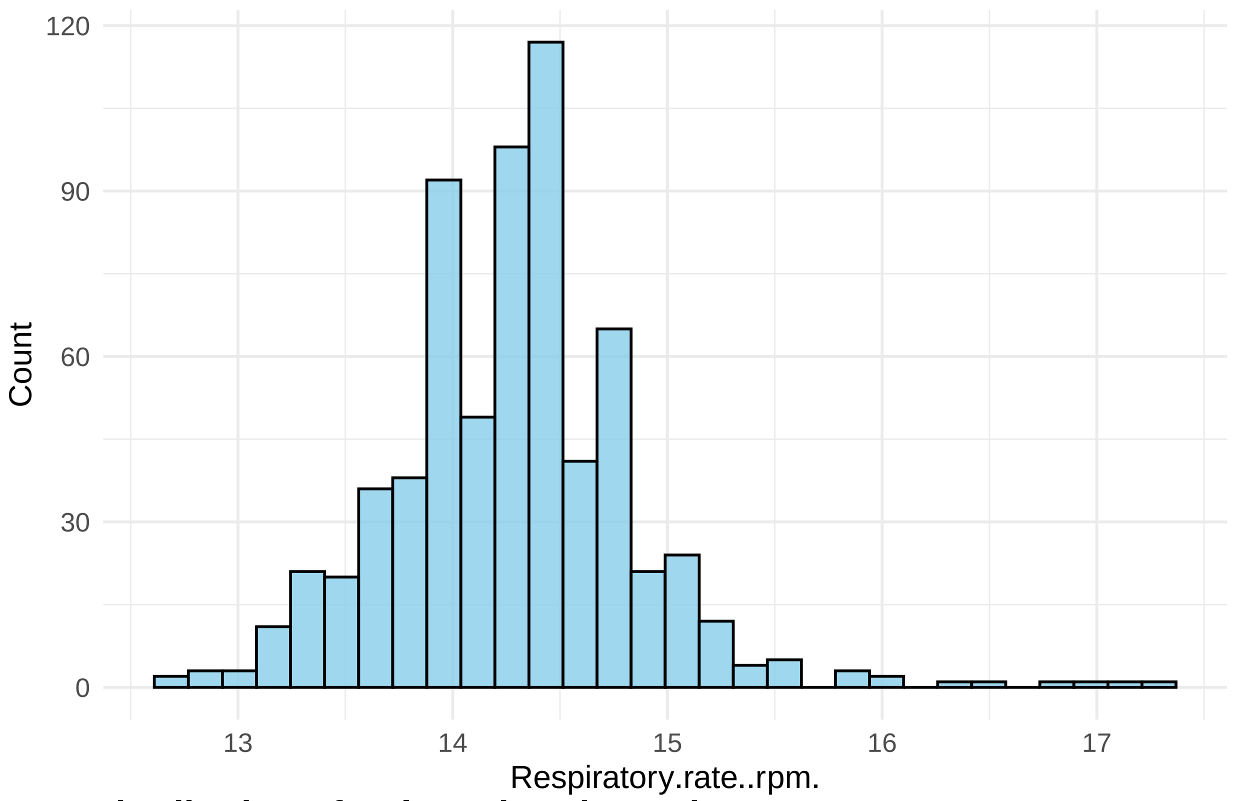


7 - Assumption Checking for Linear Models

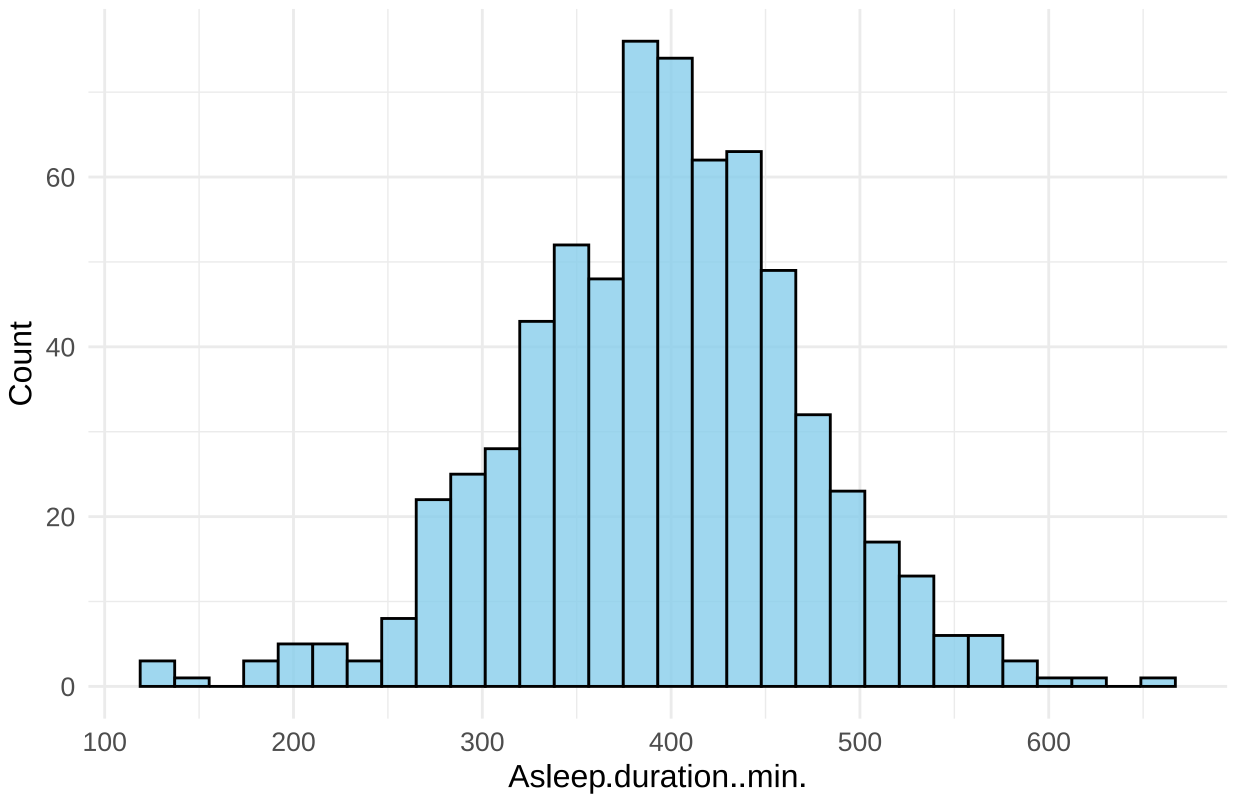
```r
library(ggplot2)

predictors <- merged_data_1 %>%
  select(-Deep..SWS..duration..min.)

for (predictor in names(predictors)) {
  print(
    ggplot(data = predictors, aes_string(x = predictor)) +
      geom_histogram(fill = "skyblue", color = "black", bins = 30, alpha = 0.8) +
      theme_minimal() +
      labs(title = paste("Distribution of", predictor),
           x = predictor, y = "Count") +
      theme(
        plot.title = element_text(size = 14, face = "bold"),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 10)
      )
  )
}
```

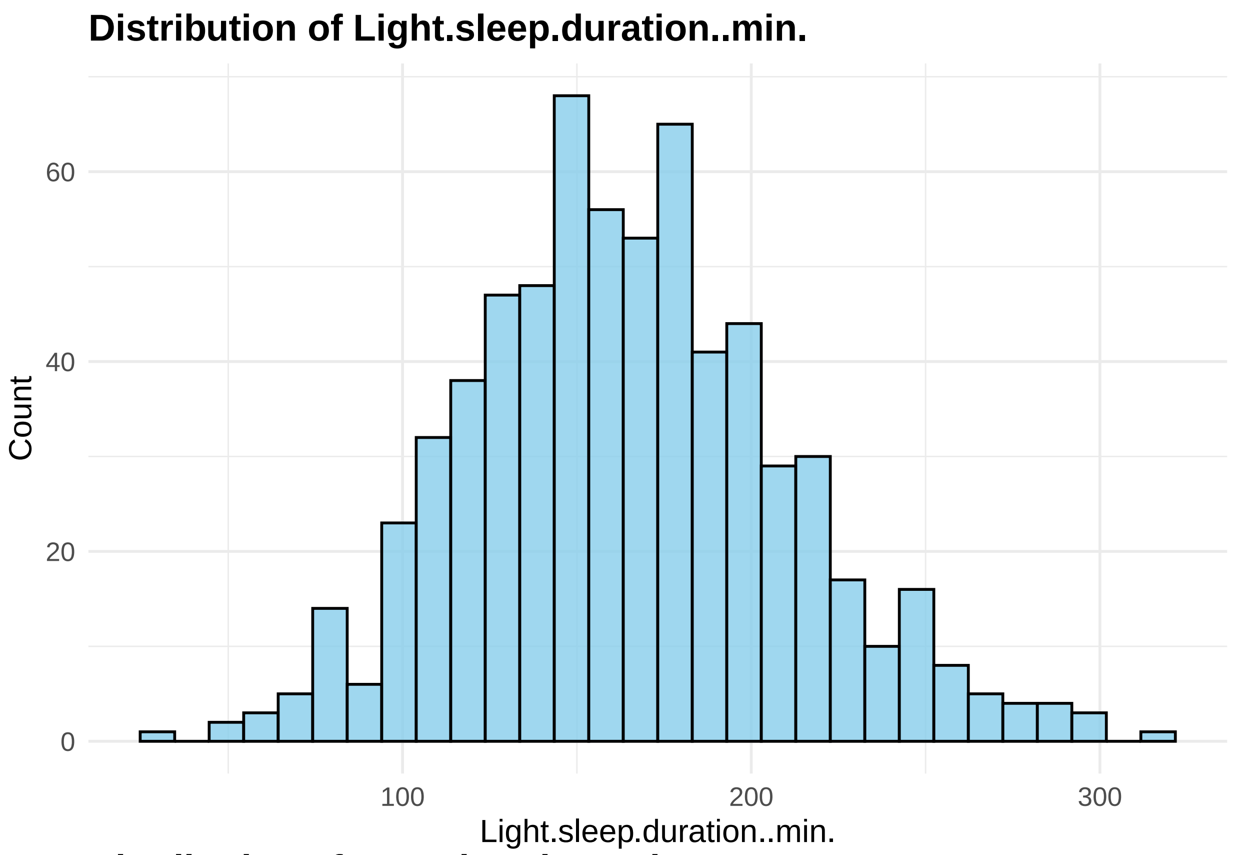
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

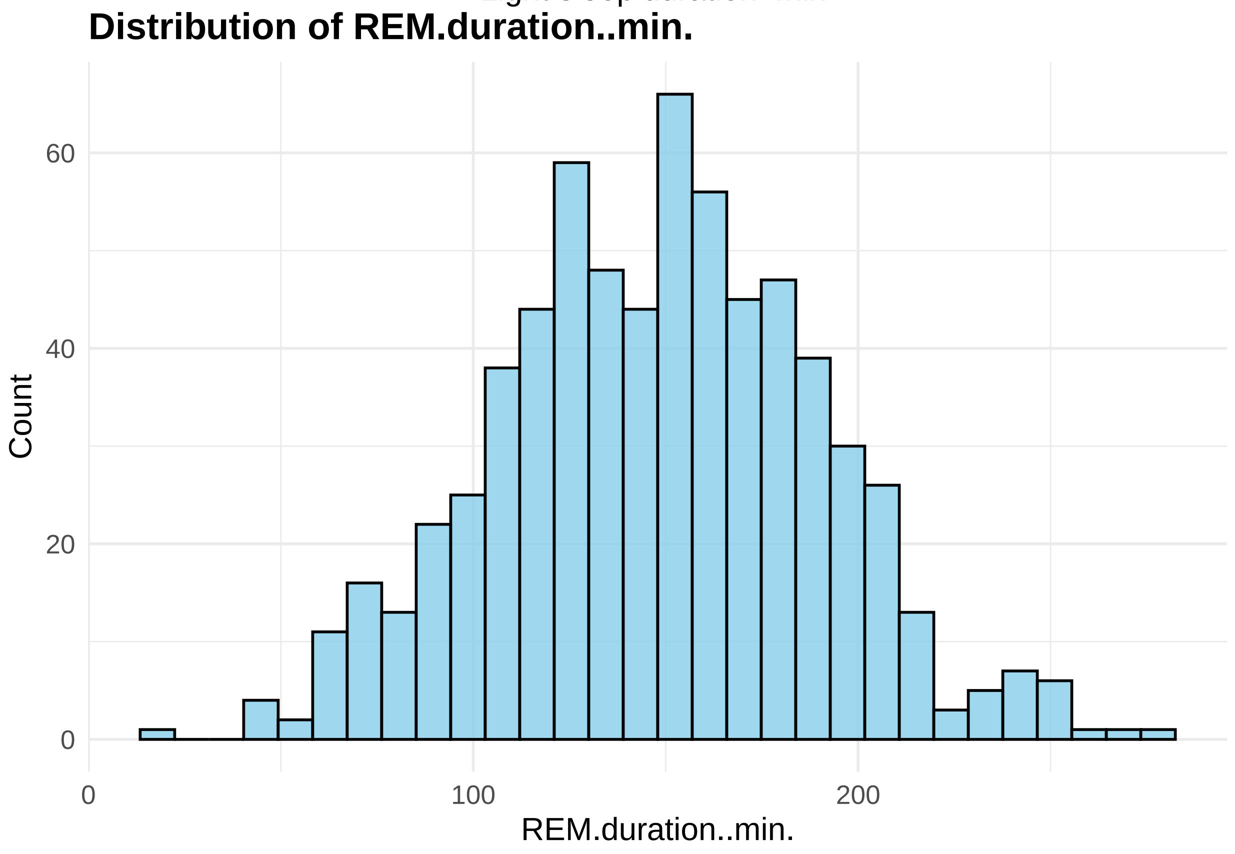
Distribution of Respiratory.rate..rpm.

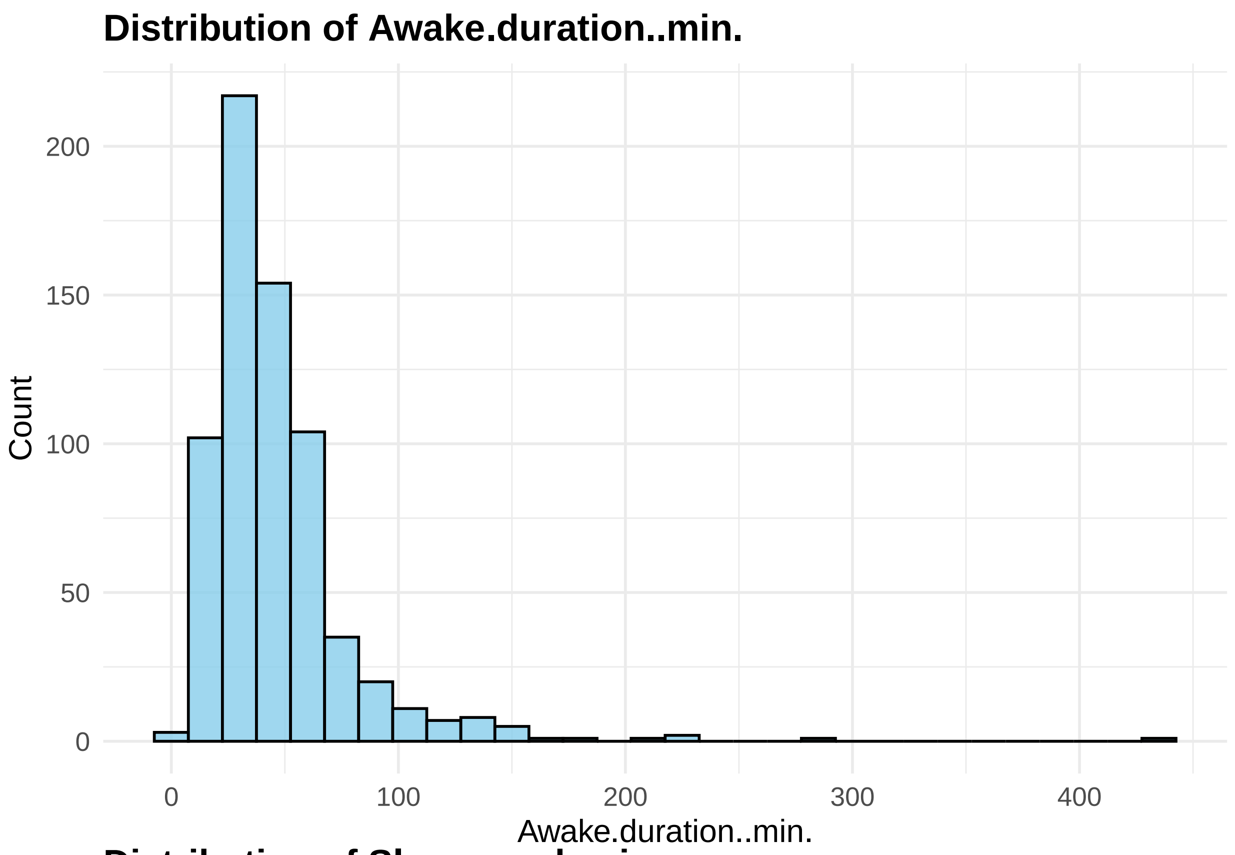


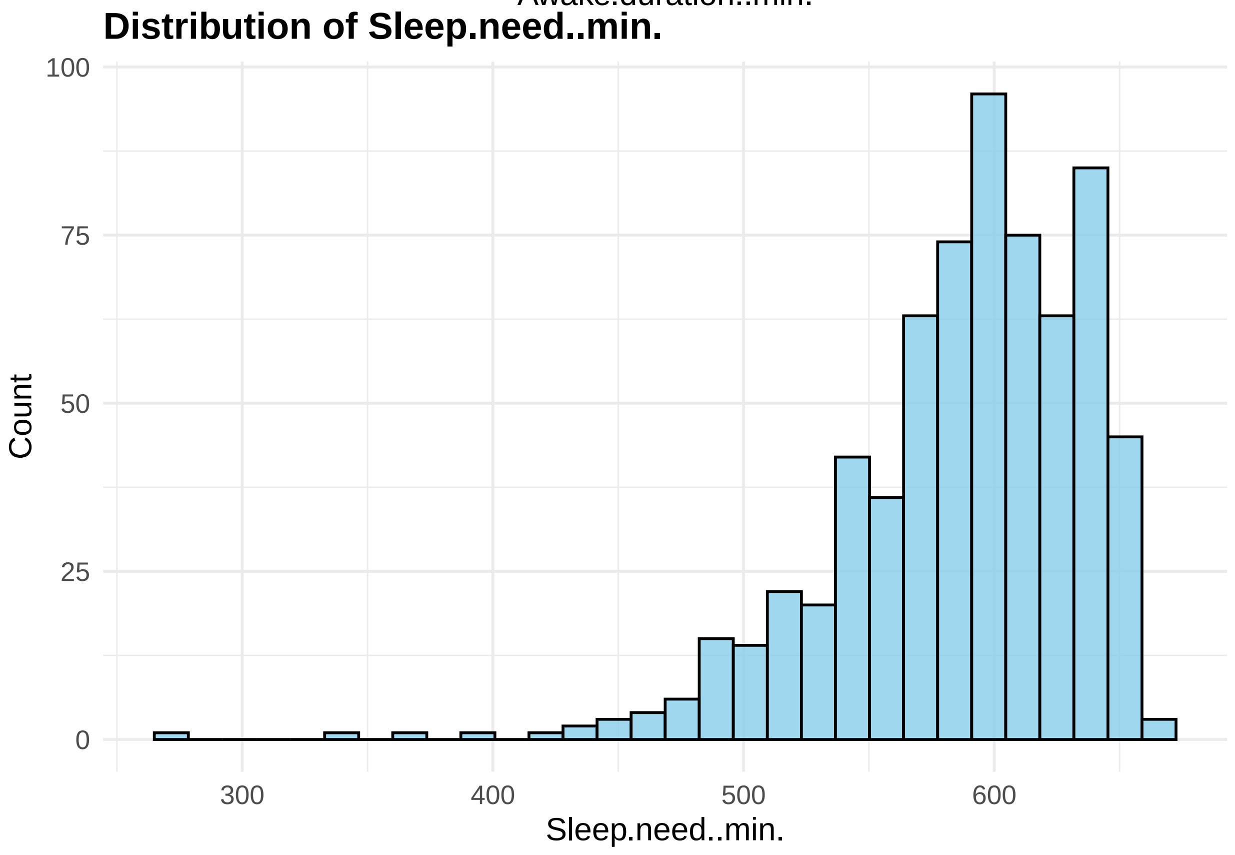
Distribution of Asleep.duration..min.




Distribution of Light.sleep.duration..min.
Count
0
20
40
60
100
200
300
Light.sleep.duration..min.

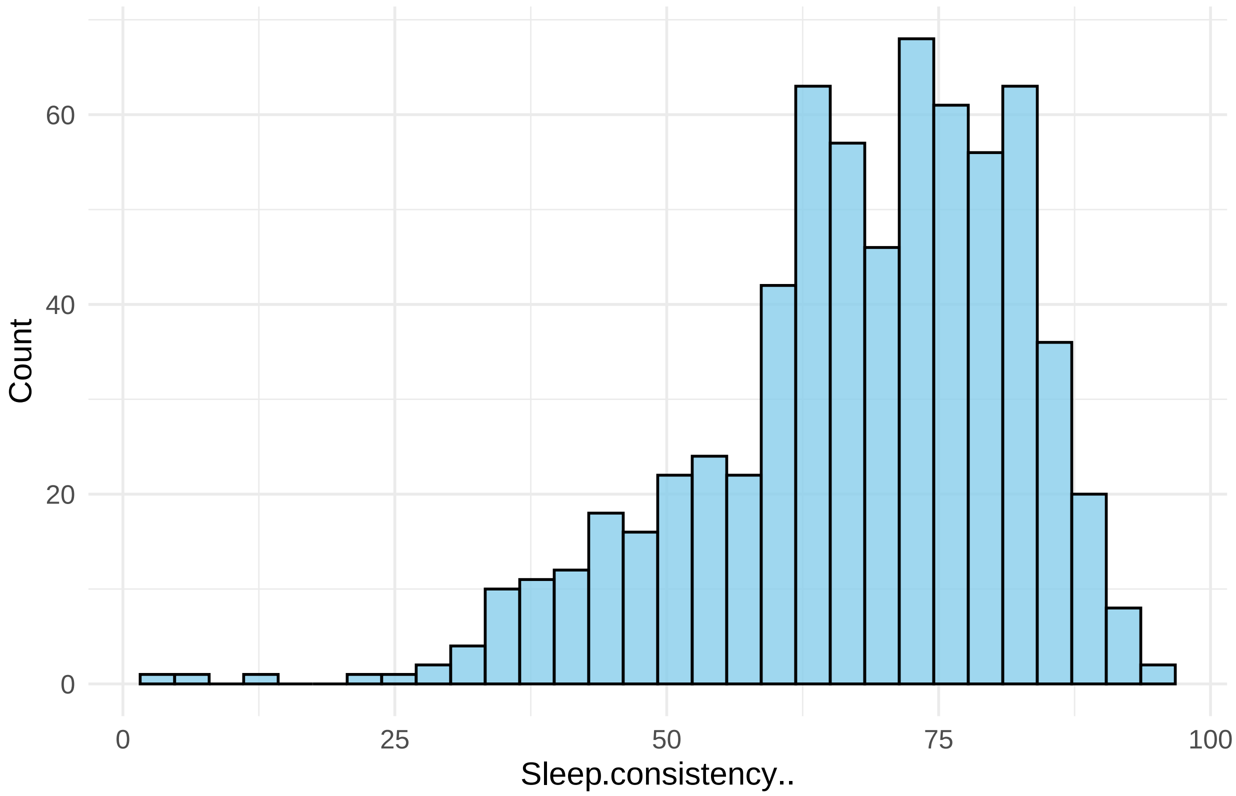

Distribution of REM.duration..min.
Count
0
20
40
60
0
100
200
REM.duration..min.



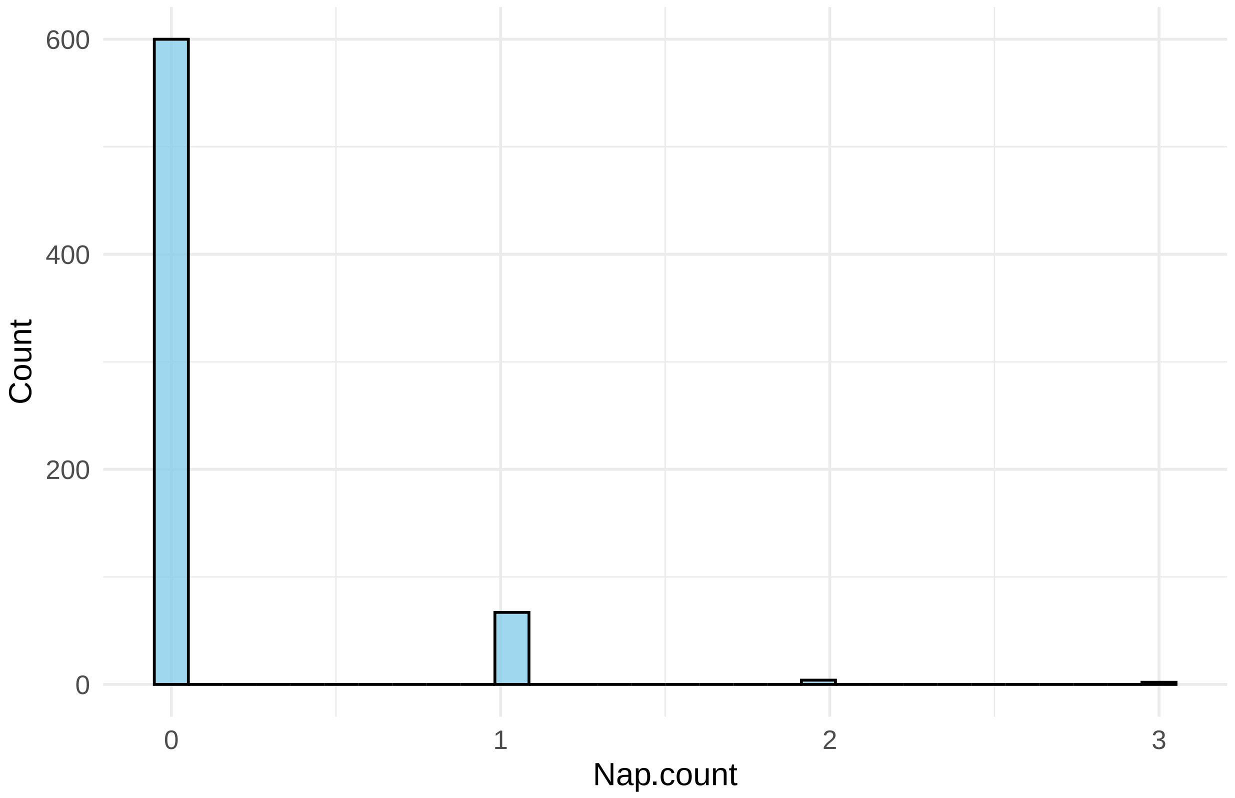


```
## Warning: Removed 5 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

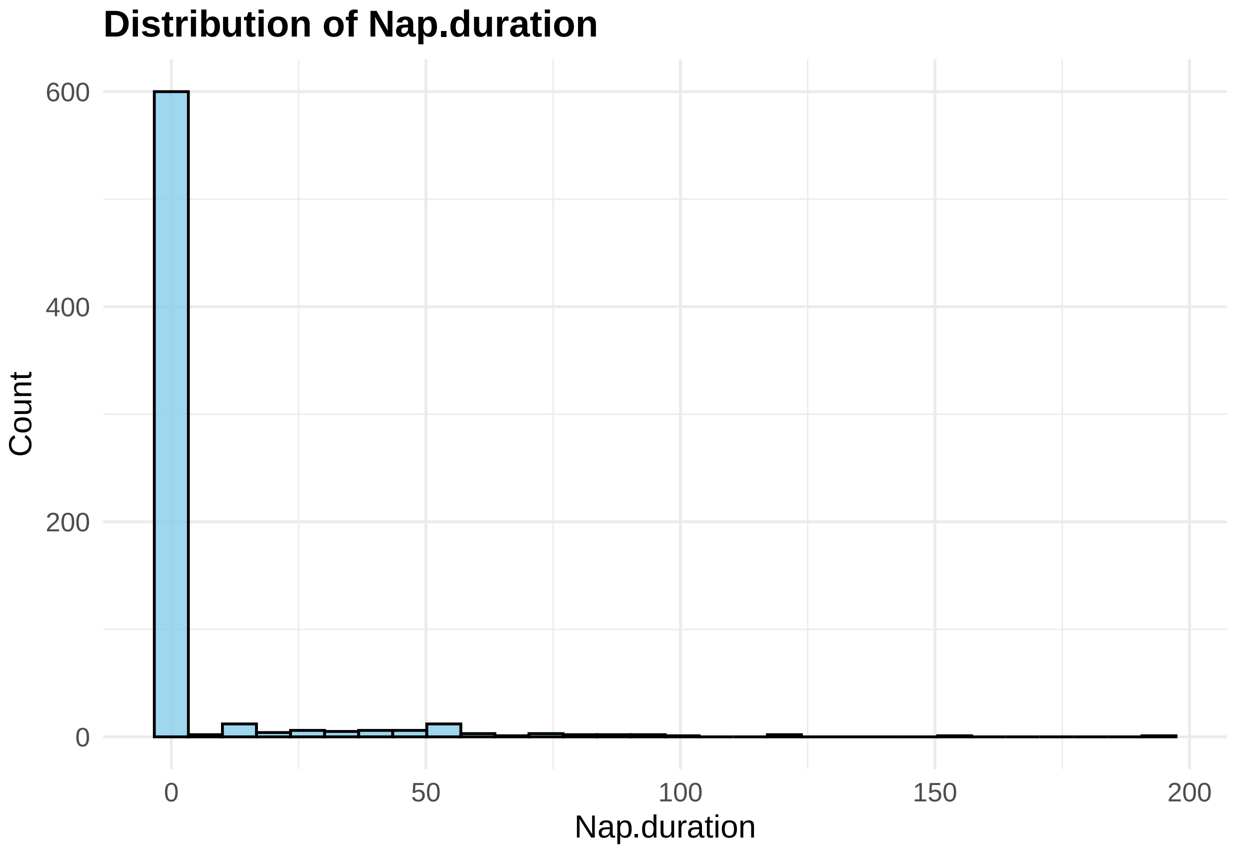
Distribution of Sleep.consistency..

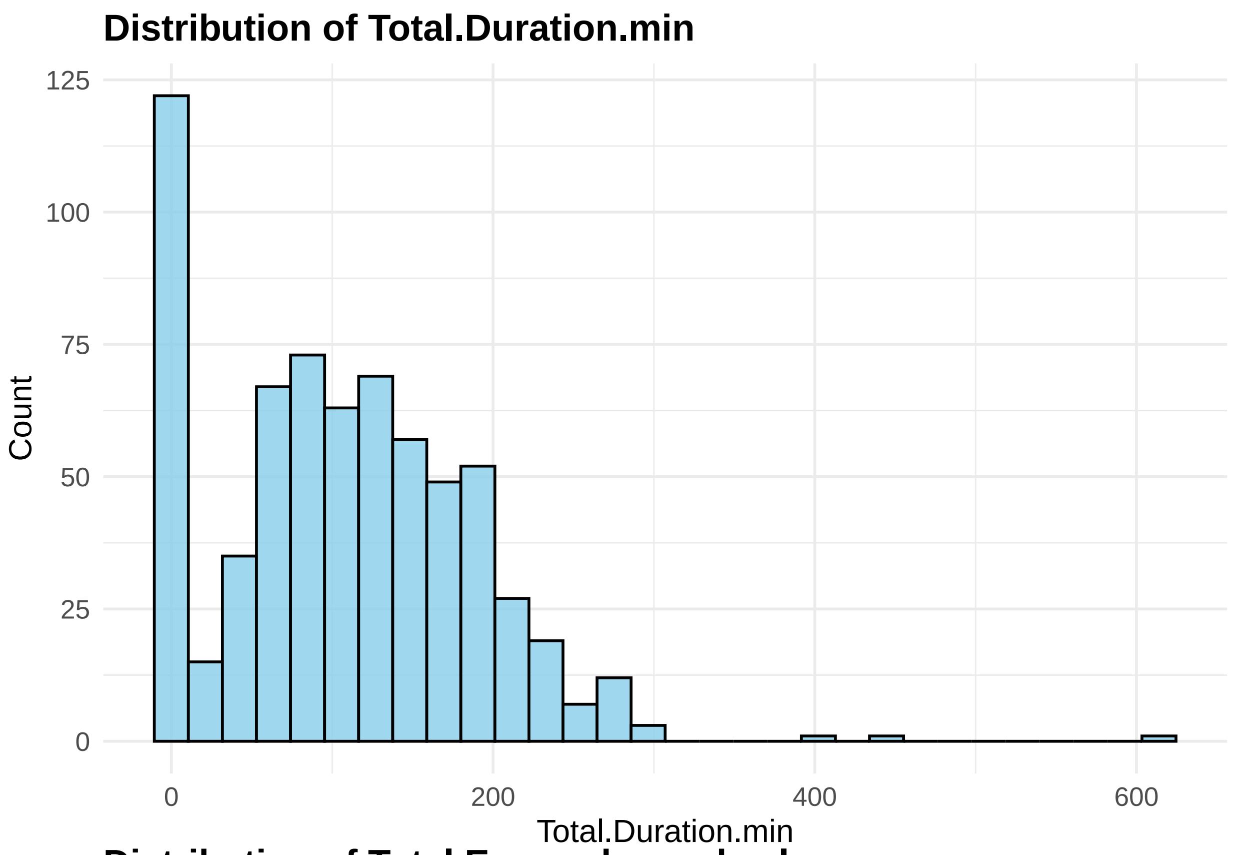


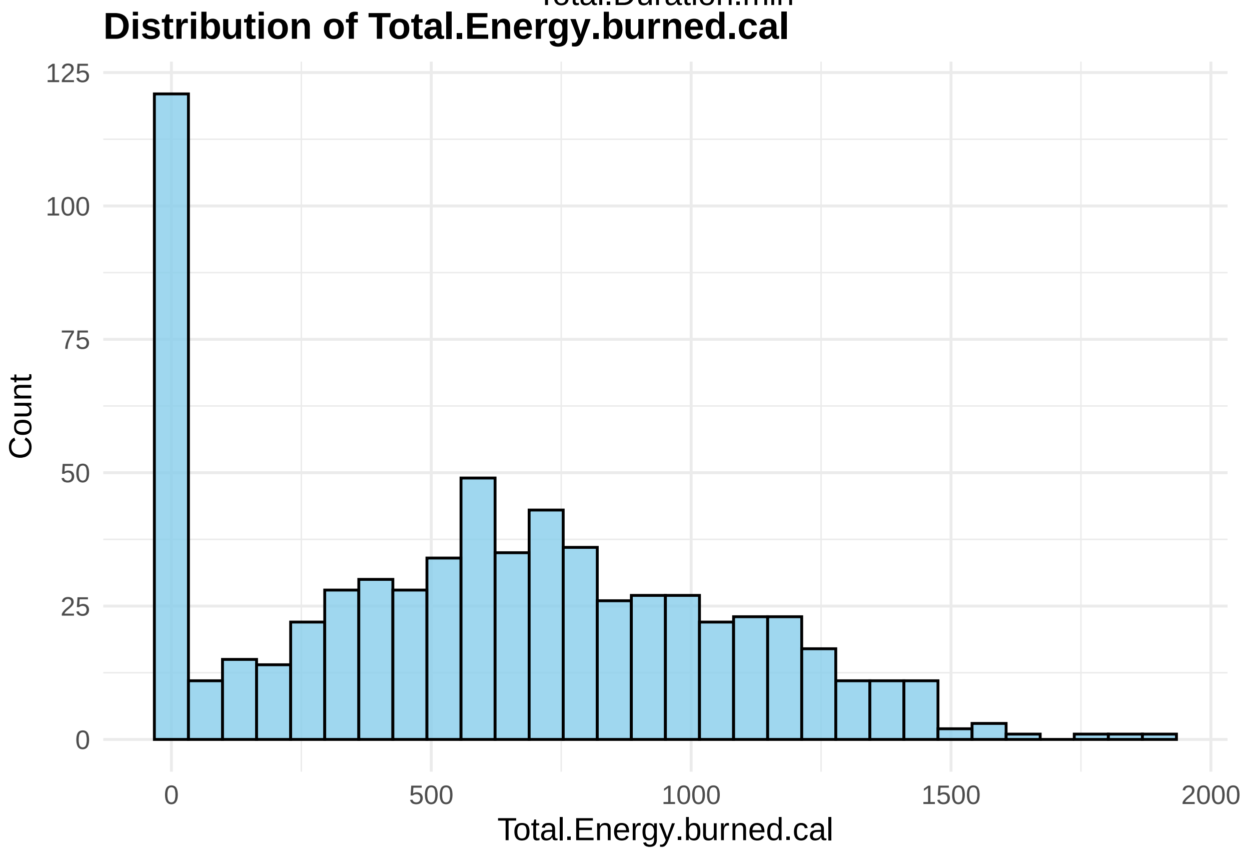
Distribution of Nap.count



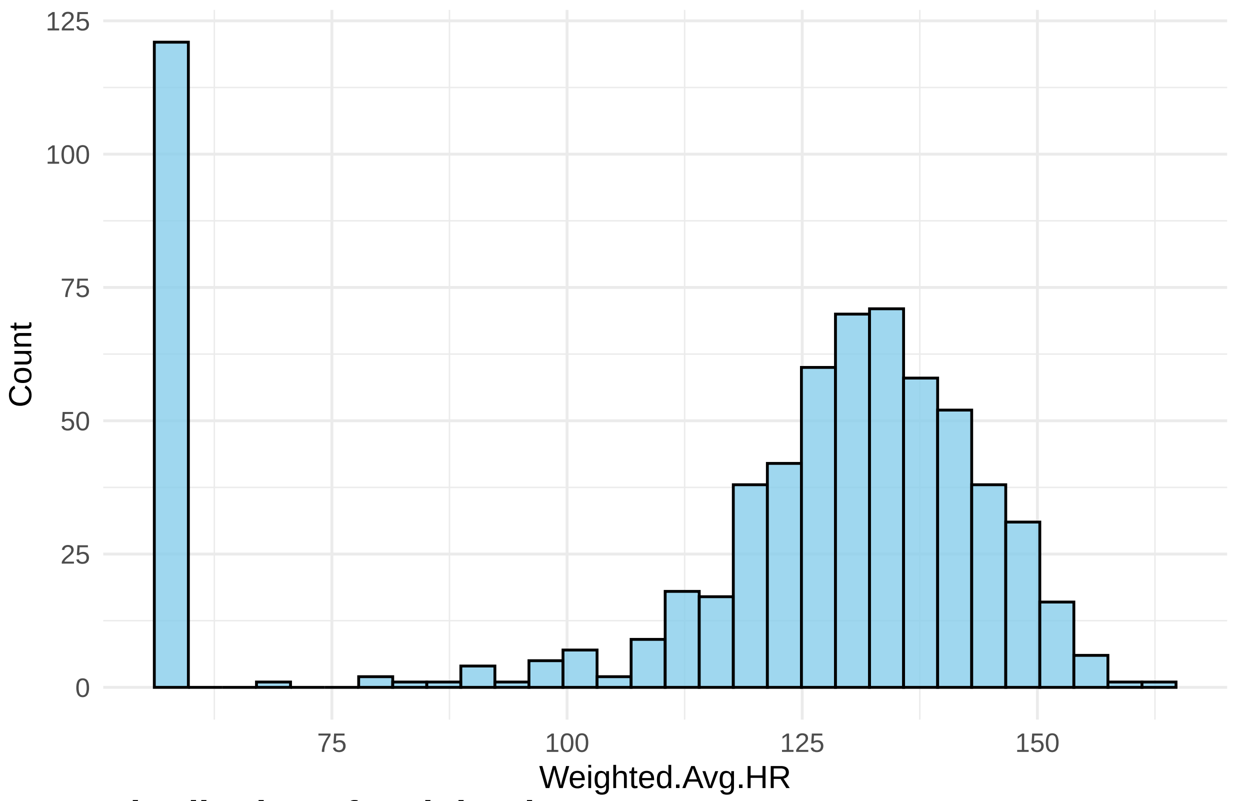
```
## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



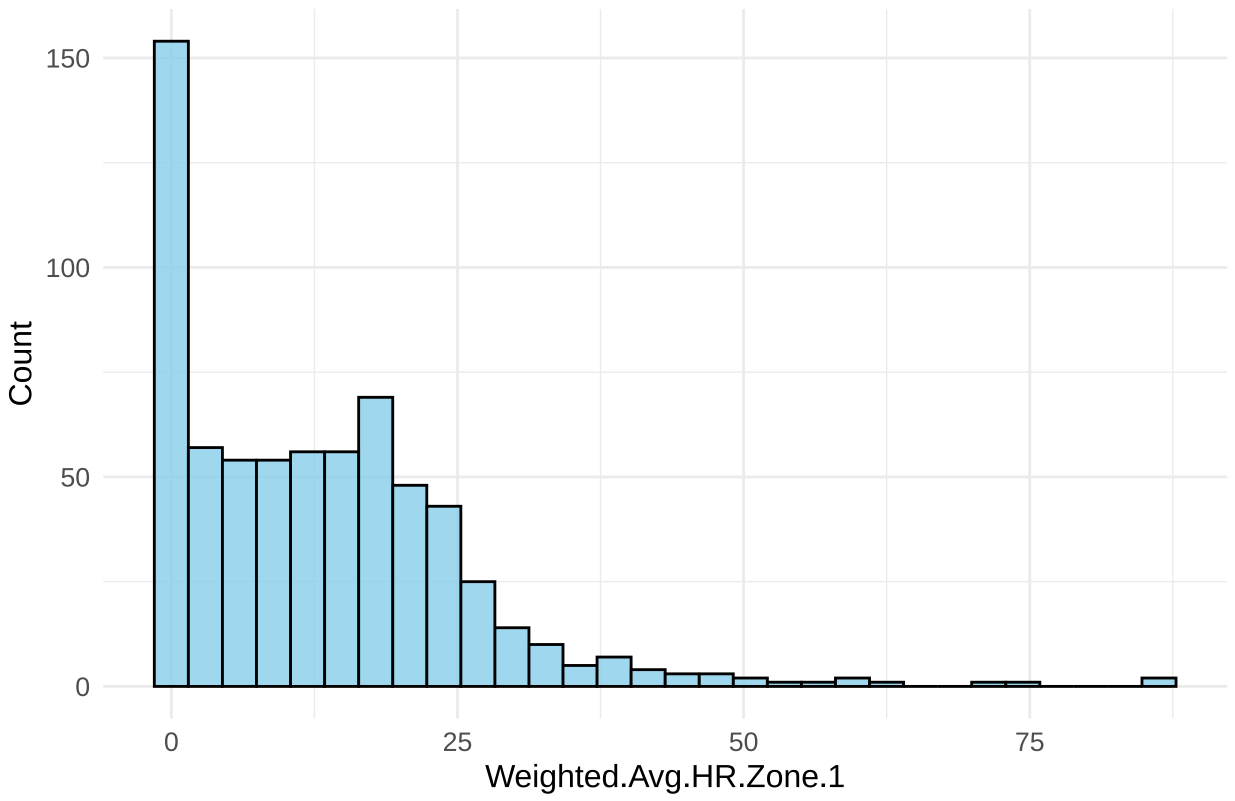

Distribution of Total.Duration.min
Count
125
100
75
50
25
0
0
200
400
600
Total.Duration.min


Distribution of Total.Energy.burned.cal
Count
125
100
75
50
25
0
0
500
1000
1500
2000
Total.Energy.burned.cal

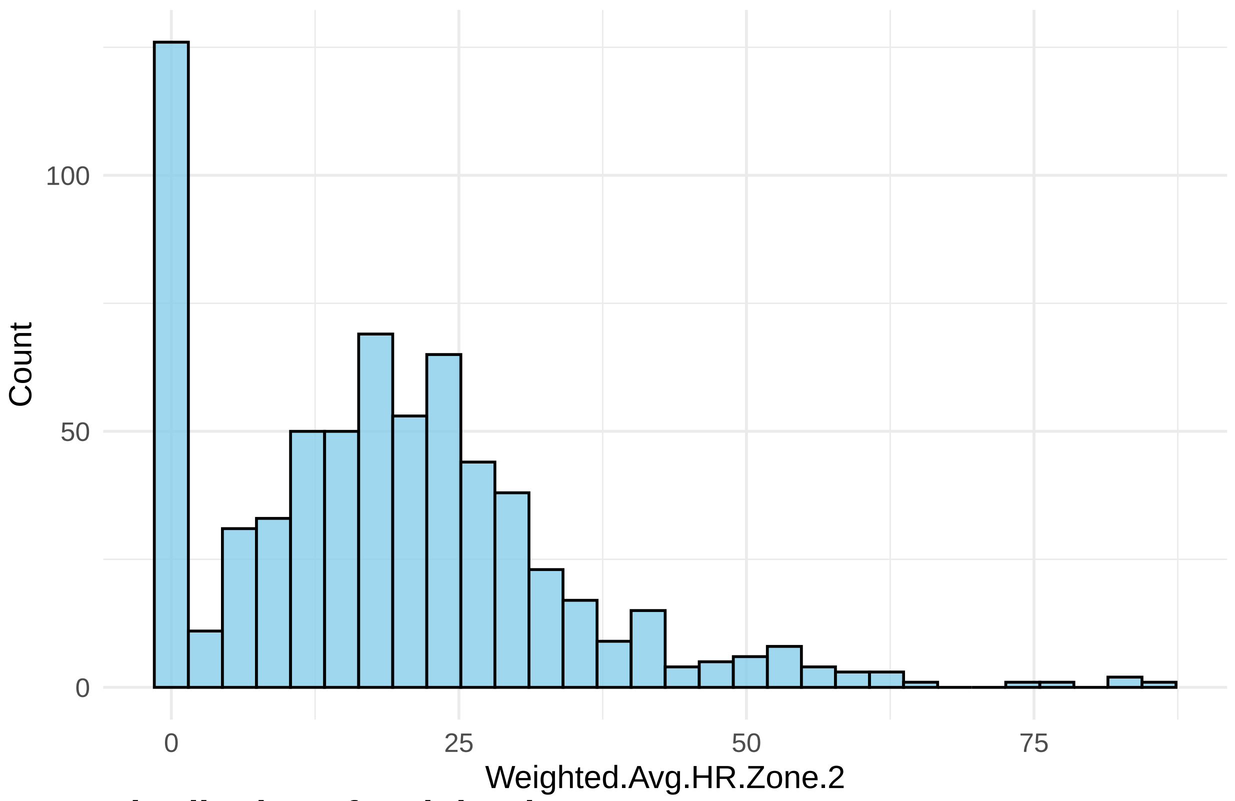
Distribution of Weighted.Avg.HR



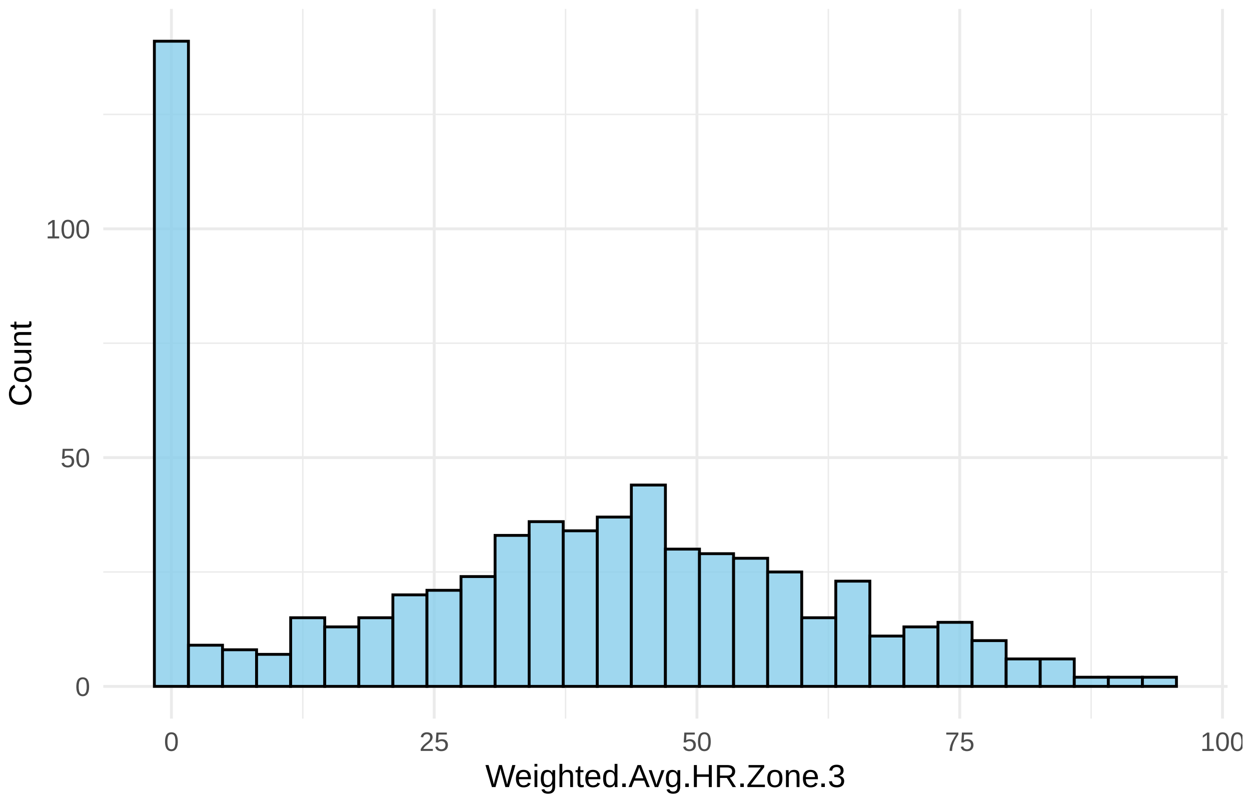
Distribution of Weighted.Avg.HR.Zone.1

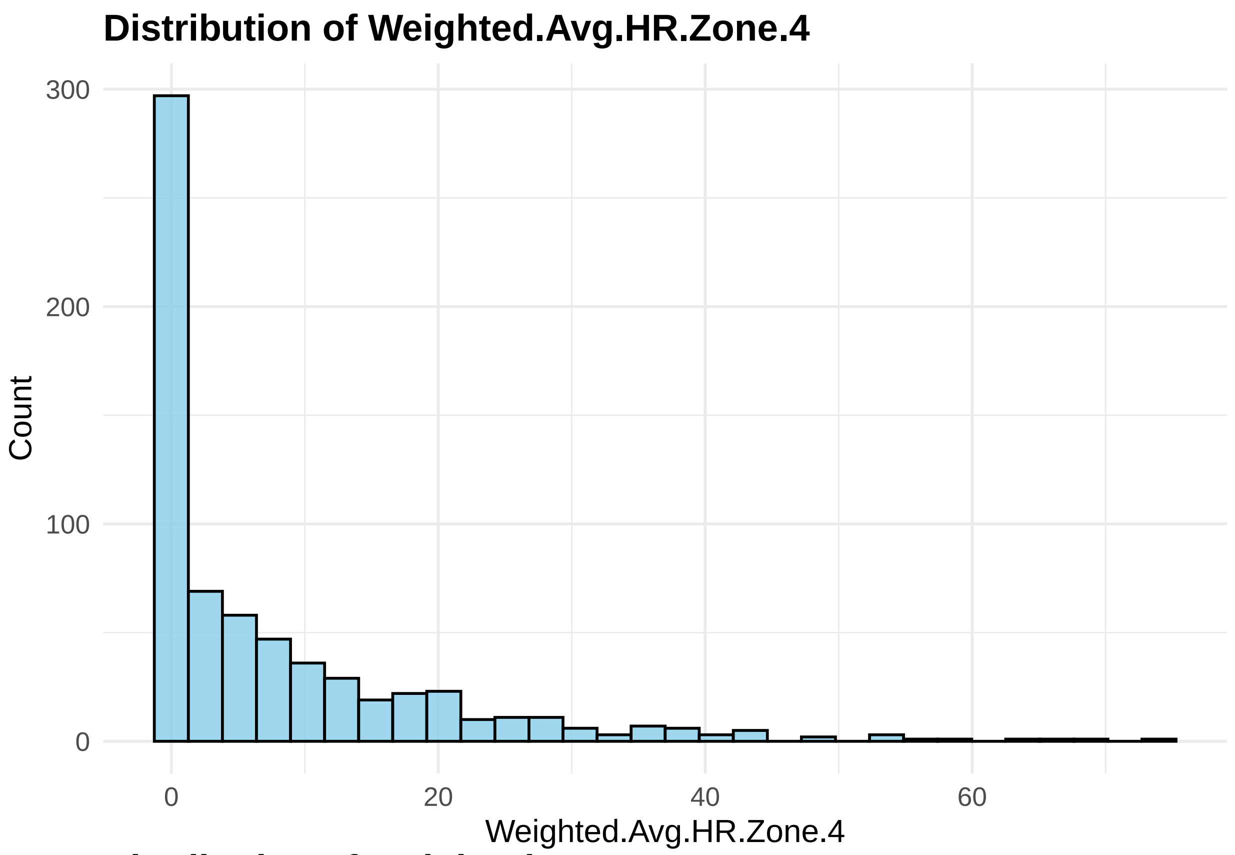


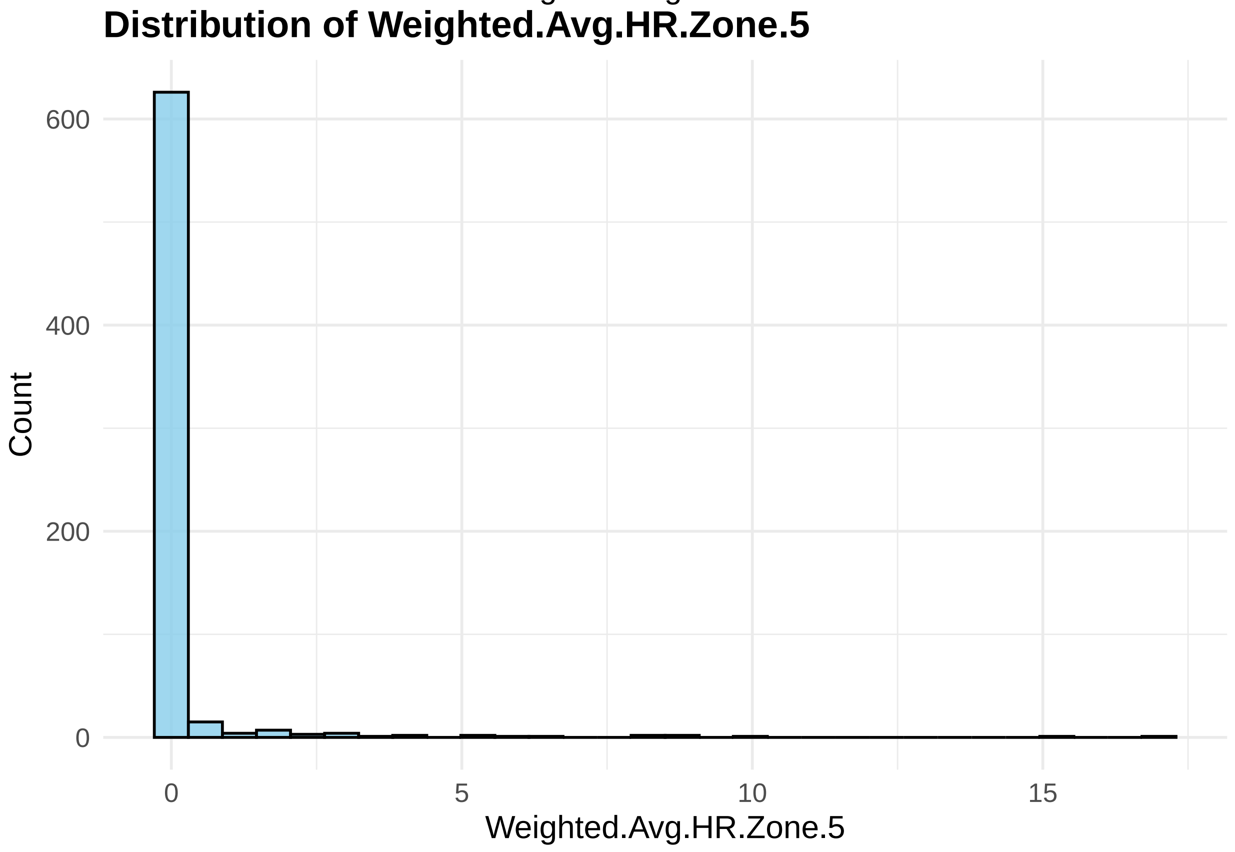
Distribution of Weighted.Avg.HR.Zone.2

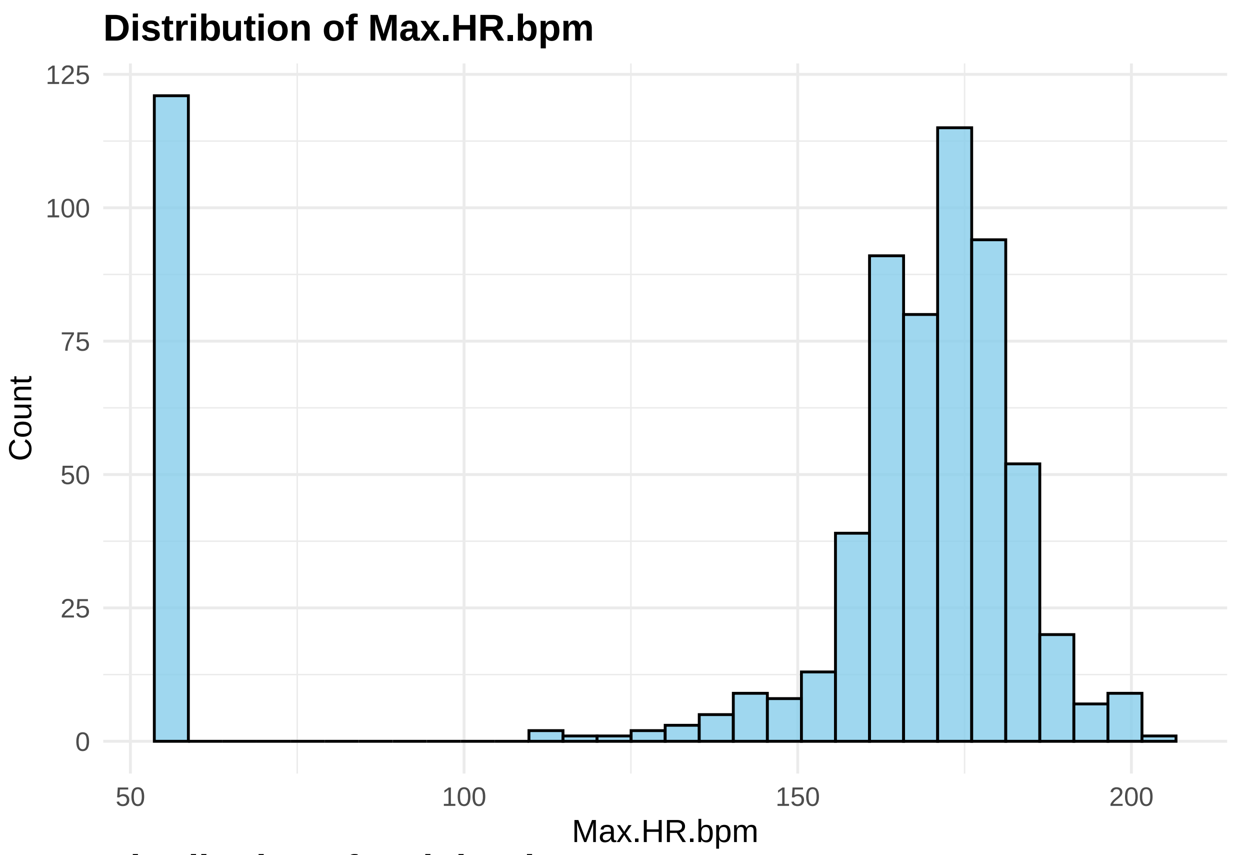


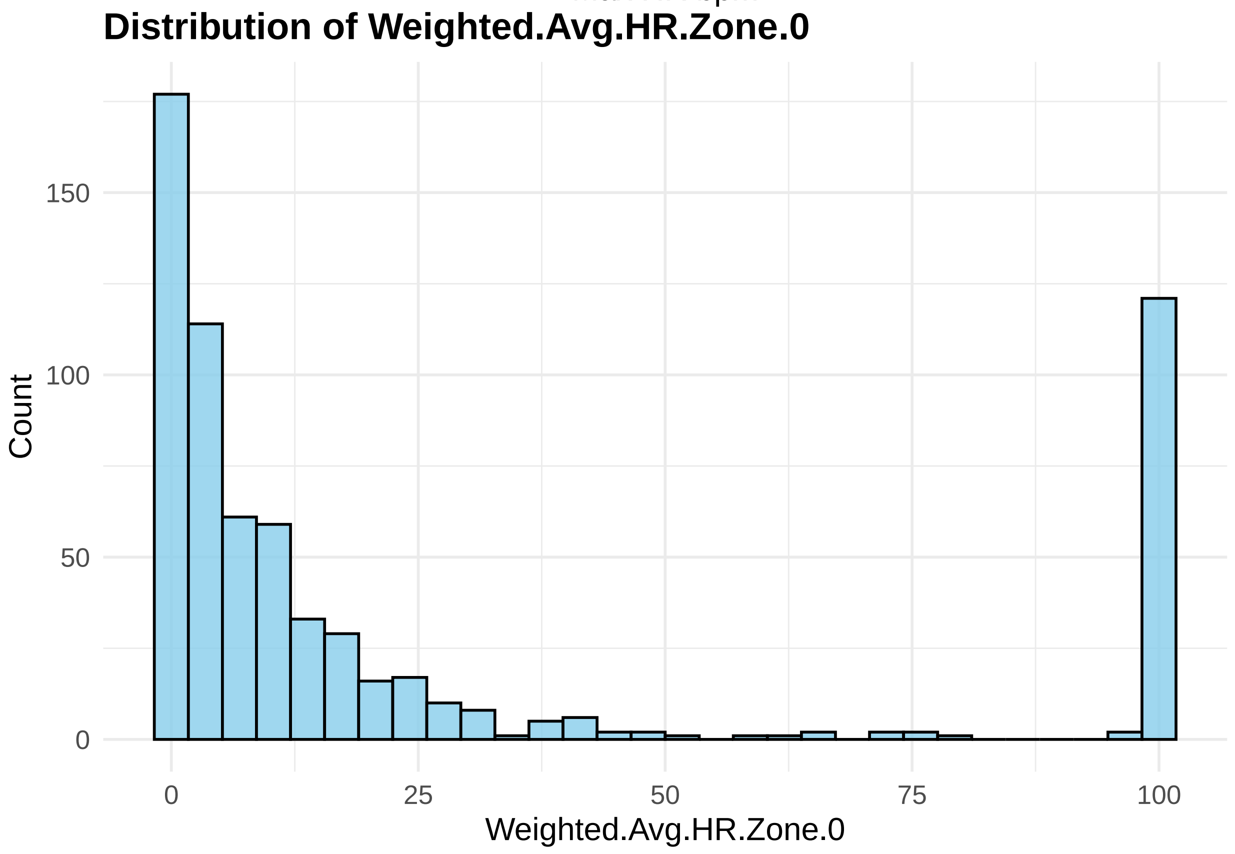
Distribution of Weighted.Avg.HR.Zone.3



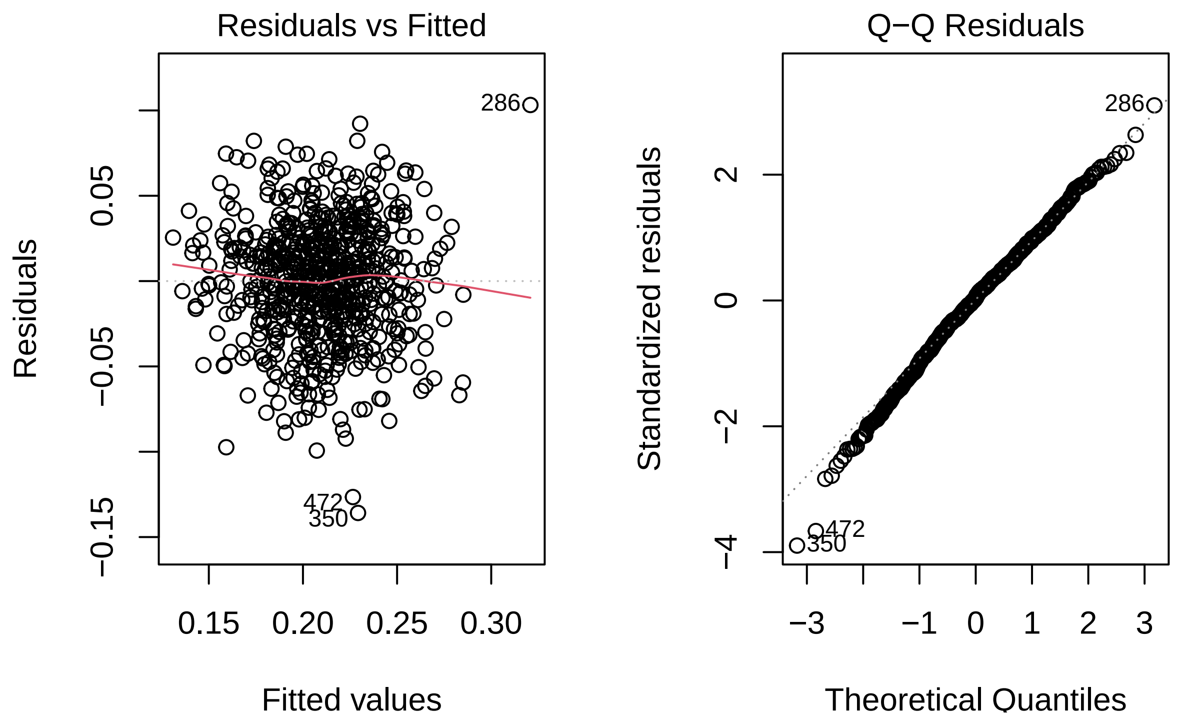

Distribution of Weighted.Avg.HR.Zone.4
Count
300
200
100
0
0
20
40
60
Weighted.Avg.HR.Zone.4


Distribution of Weighted.Avg.HR.Zone.5
Count
600
400
200
0
0
5
10
15
Weighted.Avg.HR.Zone.5

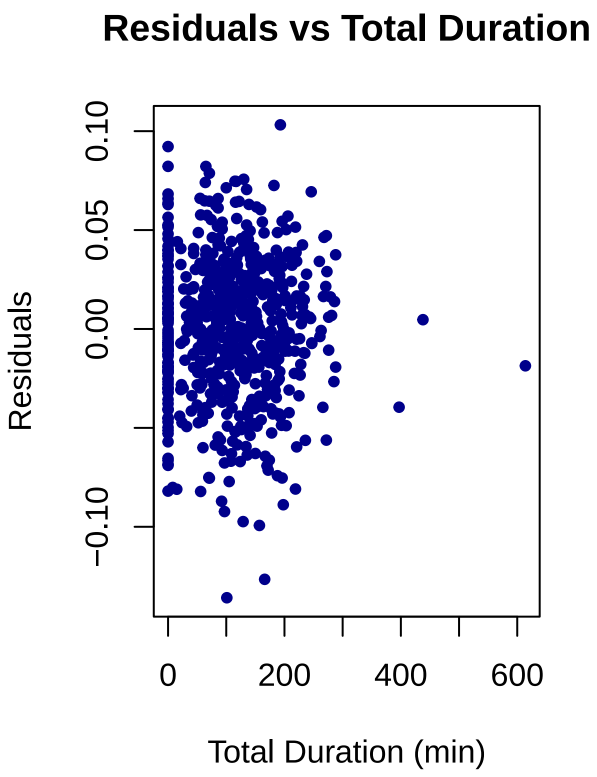

Distribution of Max.HR.bpm
Count
125
100
75
50
25
0
50
100
150
200
Max.HR.bpm

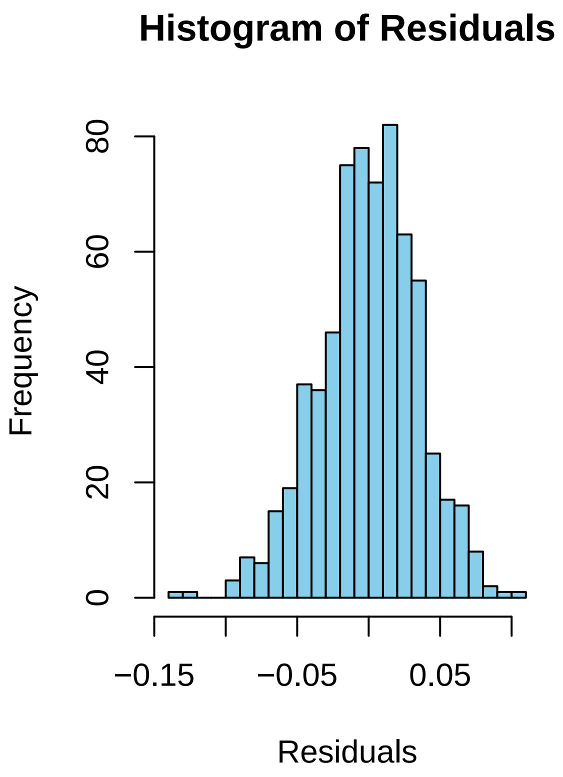

Distribution of Weighted.Avg.HR.Zone.0
Count
150
100
50
0
0
25
50
75
100
Weighted.Avg.HR.Zone.0

```r
# 1. Linearity and Homoscedasticity - Residuals vs Fitted
par(mfrow = c(1, 2))  # Split screen into 2 plots
plot(model_backward, which = 1)  # Residuals vs Fitted
plot(model_backward, which = 2)  # Q-Q plot for residuals
```



```r
# Extract the model's data for consistent lengths
model_data <- model.frame(model_backward)
residuals_backward <- residuals(model_backward)

# 2. Independence - Residuals over Total Duration
plot(model_data$Total.Duration.min, residuals_backward,
     main = "Residuals vs Total Duration",
     xlab = "Total Duration (min)", ylab = "Residuals",
     col = "darkblue", pch = 20)

# 3. Residual Histogram
hist(residuals_backward, breaks = 30, col = "skyblue",
     main = "Histogram of Residuals", xlab = "Residuals")
```





8 - Prediction

8.1 - Baseline

```
# Why? A simple model to establish a baseline for predictive performance.
# Linear regression helps identify if the relationship between predictors and response is linear. We co

library("Metrics")
set.seed(139)
split <- sample(1:nrow(merged_data_1), 0.7 * nrow(merged_data_1))  # 70/30 split

train <- merged_data_1[split, ]
test <- merged_data_1[-split, ]

# Baseline Linear Regression
baseline_model <- lm(Deep..SWS..duration..min. / Asleep.duration..min. ~ ., data = train)
baseline_predictions <- predict(baseline_model, test)

# Model evaluation (RMSE and R-squared)
library(Metrics)
rmse_baseline <- rmse(test$Deep..SWS..duration..min. / test$Asleep.duration..min., baseline_predictions)
r2_baseline <- cor(test$Deep..SWS..duration..min. / test$Asleep.duration..min., baseline_predictions)^2

print(rmse_baseline)
```

```
## [1] NA
```

```
print(r2_baseline)
```

```
## [1] NA
```

8.2 - LASSO

```r
# Lasso Regression
# Why? Useful for datasets with many predictors to automatically perform feature selection.
# Lasso helps reduce overfitting and simplifies the model by shrinking less important coefficients to z

set.seed(139)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```r
library(Metrics)

train_clean <- na.omit(train)
test_clean <- na.omit(test)

x_train <- as.matrix(train_clean %>% select(-Deep..SWS..duration..min.))
y_train <- train_clean$Deep..SWS..duration..min. / train_clean$Asleep.duration..min.

x_test <- as.matrix(test_clean %>% select(-Deep..SWS..duration..min.))
y_test <- test_clean$Deep..SWS..duration..min. / test_clean$Asleep.duration..min.

# Fit Lasso model
lasso_model <- cv.glmnet(x_train, y_train, alpha = 1)
best_lambda <- lasso_model$lambda.min

# Predict on test set
lasso_preds <- predict(lasso_model, s = best_lambda, newx = x_test)

# Calculate RMSE
rmse_lasso <- rmse(y_test, lasso_preds)

# Calculate R-squared
rss <- sum((y_test - lasso_preds)^2)
tss <- sum((y_test - mean(y_test))^2)
r2_lasso <- 1 - (rss / tss)

print(paste("Lasso RMSE:", round(rmse_lasso, 4)))
```

```
## [1] "Lasso RMSE: 0.0085"
```

```r
print(paste("Lasso R-squared:", round(r2_lasso, 4)))
```

```
## [1] "Lasso R-squared: 0.961"
```

8.3 - Random Forest

```r
# Random Forest
# Why? Suitable for capturing nonlinear relationships and interactions between predictors.
# Random Forest is also robust to overfitting and can rank feature importance.

library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3

## Loading required package: lattice

##
## Attaching package: 'lattice'

## The following object is masked from 'package:faraway':
##
##     melanoma

##
## Attaching package: 'caret'

## The following objects are masked from 'package:Metrics':
##
##     precision, recall
```

```r
library(Metrics)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.3.3

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
set.seed(139)
train_clean <- na.omit(train)
test_clean <- na.omit(test)

# Define the response variable
y_train <- train_clean$Deep..SWS..duration..min. / train_clean$Asleep.duration..min.
y_test <- test_clean$Deep..SWS..duration..min. / test_clean$Asleep.duration..min.

x_train <- train_clean %>% select(-Deep..SWS..duration..min.)
x_test <- test_clean %>% select(-Deep..SWS..duration..min.)

# Combine predictors and response for training
train_data <- cbind(x_train, y_train)

# Define cross-validation settings
set.seed(139)
cv_control <- trainControl(method = "cv", number = 5)  # 5-fold cross-validation

# Train Random Forest model with cross-validation
rf_cv_model <- train(
  x = x_train,
  y = y_train,
  method = "rf",
  trControl = cv_control,
  tuneGrid = expand.grid(mtry = seq(2, sqrt(ncol(x_train)), length.out = 5)),  # Adjust mtry values
  ntree = 1000
)
```

```
## Warning: Setting row names on a tibble is deprecated.
```

```
## Warning: Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
## Setting row names on a tibble is deprecated.
```

```r
# Print the best hyperparameters
print(rf_cv_model$bestTune)
```

```
##       mtry
## 4 3.769174
```

```r
# Predict on test set using the best model
rf_cv_preds <- predict(rf_cv_model, x_test)

# Evaluate the model: RMSE and R-squared
rmse_rf_cv <- rmse(y_test, rf_cv_preds)
rss_rf_cv <- sum((y_test - rf_cv_preds)^2)
tss_rf_cv <- sum((y_test - mean(y_test))^2)
r2_rf_cv <- 1 - (rss_rf_cv / tss_rf_cv)

print(paste("Cross-Validated Random Forest RMSE:", round(rmse_rf_cv, 4)))
```

```
## [1] "Cross-Validated Random Forest RMSE: 0.0373"
```

```r
print(paste("Cross-Validated Random Forest R-squared:", round(r2_rf_cv, 4)))
```

```
## [1] "Cross-Validated Random Forest R-squared: 0.2516"
```