Final project on Readiness Score Report

AUTHOR
Jatin Adya and Sharath Reddy Muthyala

```
rm(list = ls())
```

Load necessary libraries

```
library(readr)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
library(ggplot2)
library(forecast)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
```

```
library(rsample)
library(lattice)
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
library(randomForest)
```

```
randomForest 4.7-1.1

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


Attaching package: 'randomForest'

The following object is masked from 'package:ggplot2':

    margin

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

    combine
```

```
library(rpart)
library(rpart.plot)
library(caret)
library(pdp)
library(Matrix)
library(lubridate)
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ forcats 1.0.0     ✔ tibble  3.2.1
✔ purrr   1.0.2     ✔ tidyr   1.3.1
✔ stringr 1.5.0

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ randomForest::combine() masks dplyr::combine()
✖ tidyr::expand()         masks Matrix::expand()
✖ dplyr::filter()         masks stats::filter()
✖ dplyr::lag()            masks stats::lag()
✖ purrr::lift()           masks caret::lift()
✖ randomForest::margin()  masks ggplot2::margin()
✖ tidyr::pack()           masks Matrix::pack()
✖ purrr::partial()        masks pdp::partial()
✖ tidyr::unpack()         masks Matrix::unpack()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to
become errors
```

Data Import

```
# Set the path to the directory containing your files
path <- "/Users/sharath/Desktop/SPRING 2024/STAT 5125/Project/Daily Readiness"
```

```r
# Create a vector of filenames
filenames <- list.files(path, pattern = "Daily Readiness Score -.*\\.csv", full.names = T

# Read and merge all datasets
data <- filenames %>%
  lapply(function(file) read_csv(file, show_col_types = FALSE)) %>%
  bind_rows()
```

Data Cleaning

```r
# Remove rows with missing values
data <- na.omit(data)
```

```r
# First few rows of the data
head(data)
```

```
# A tibble: 6 × 9
  date       readiness_score_value readiness_state activity_subcomponent
  <date>                     <dbl> <chr>                             <dbl>
1 2023-10-31                    97 HIGH                                 97
2 2023-10-30                    60 MEDIUM                               60
3 2023-10-29                    45 MEDIUM                               45
4 2023-10-28                    58 MEDIUM                               58
5 2023-10-27                    92 HIGH                                 92
6 2023-10-26                    74 HIGH                                 74
# ℹ 5 more variables: sleep_subcomponent <dbl>, hrv_subcomponent <dbl>,
#   activity_state <chr>, sleep_state <chr>, hrv_state <chr>
```

Data Description

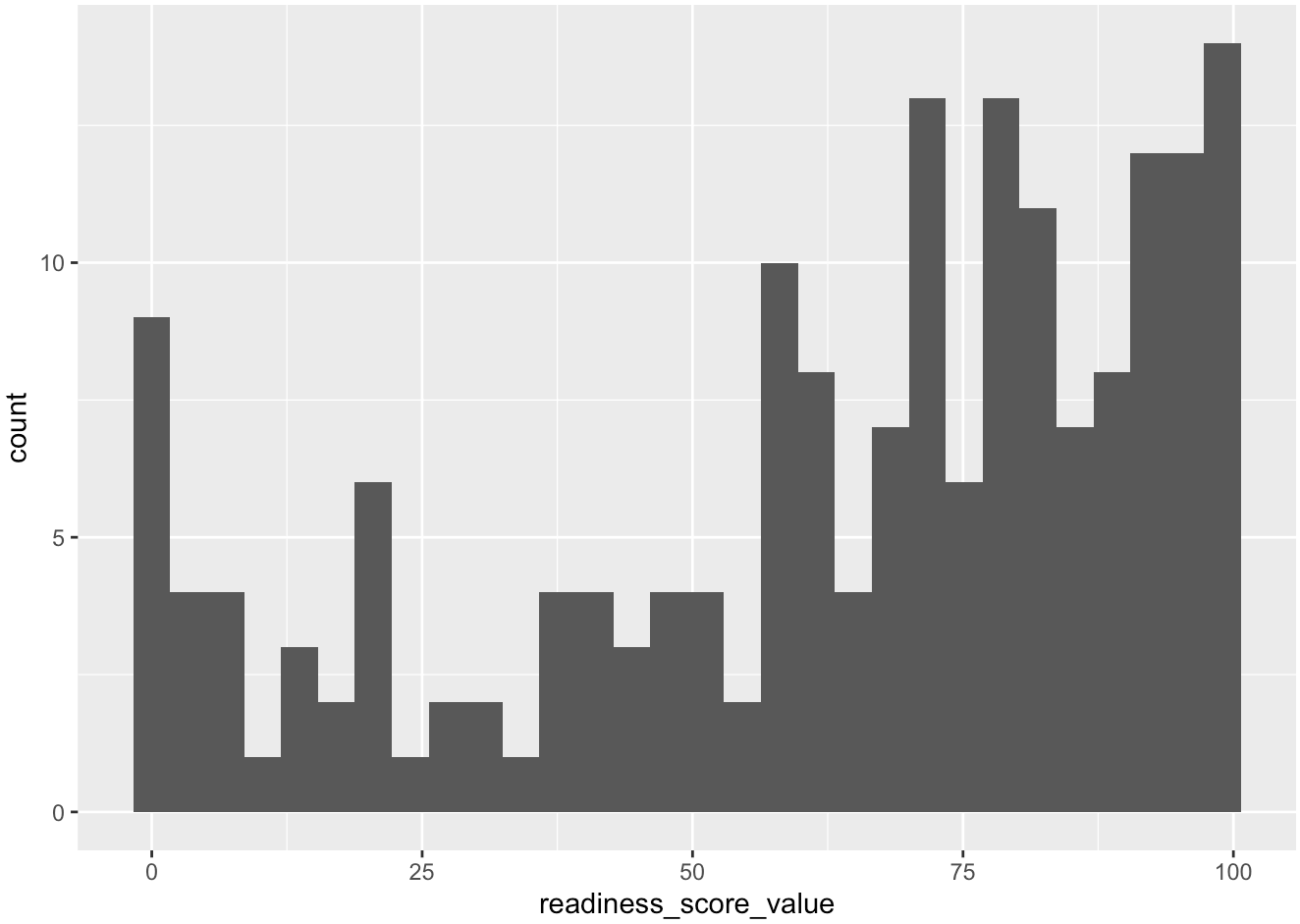
- **Date**: The date of the score.
- **Readiness Score Value**: The final daily readiness score.
- **Readiness State**: Description of the overall readiness state.
- **Activity Subcomponent**: The activity subcomponent.
- **Sleep Subcomponent**: The recent sleep subcomponent.
- **HRV Subcomponent**: The heart rate variability subcomponent.
- **Activity State**: Description of the activity state.
- **Sleep State**: Description of the recent sleep state.
- **HRV State**: Description of the heart rate variability state.

Data Exploration

```r
# Histogram plot of the readiness score
ggplot(data, aes(x = readiness_score_value)) +
```

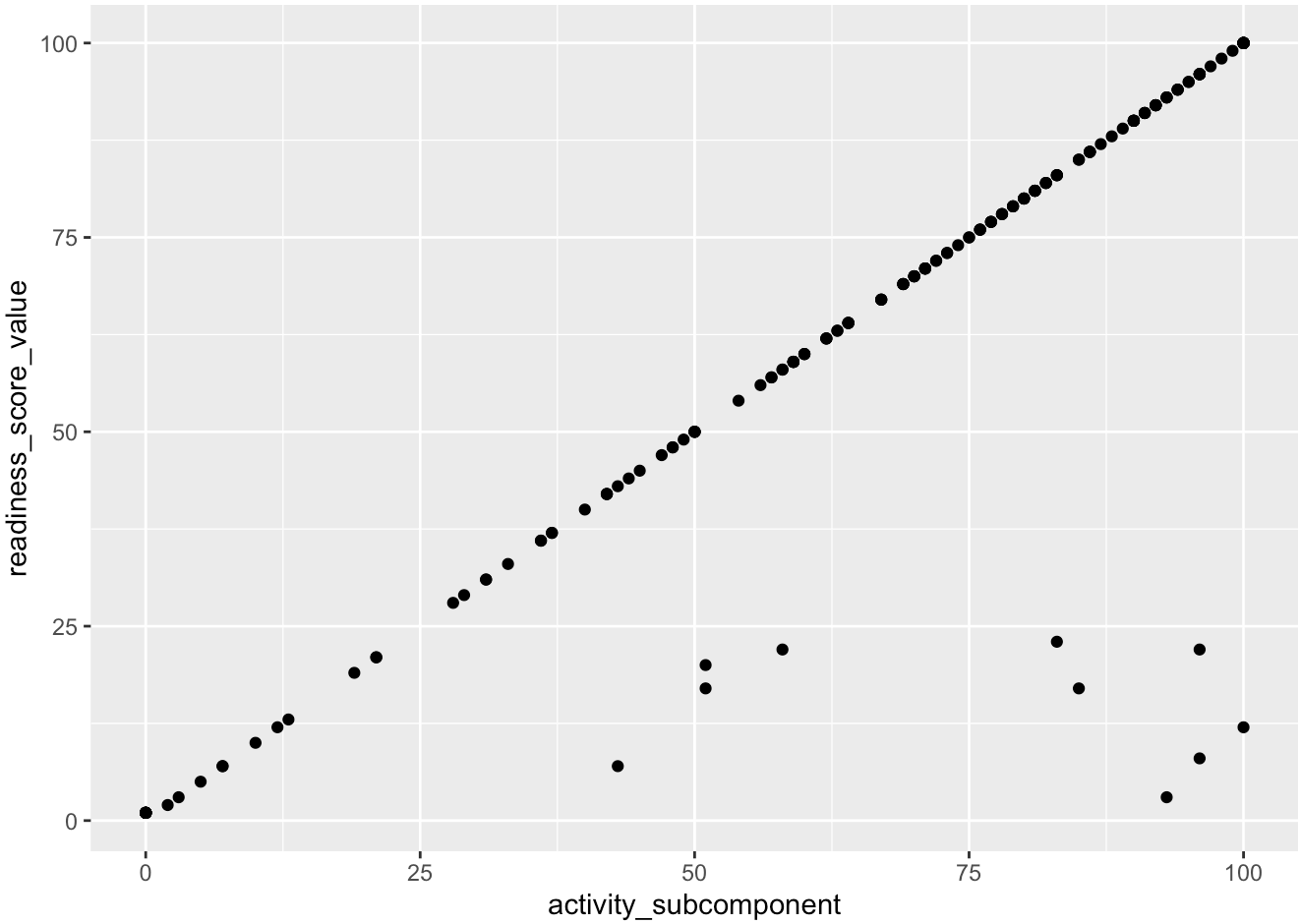
```
  geom_histogram()
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The histogram of readiness score values displays a bimodal distribution with peaks around scores of 10 and 90, suggesting that instances of readiness are often at the extremes—either very low or very high. There are fewer moderate scores around the 50 range. This pattern indicates that the conditions measured tend to be distinctly polarized, either indicating very good or very poor readiness.

```
ggplot(data, aes(x = activity_subcomponent, y = readiness_score_value)) +
  geom_point()
```



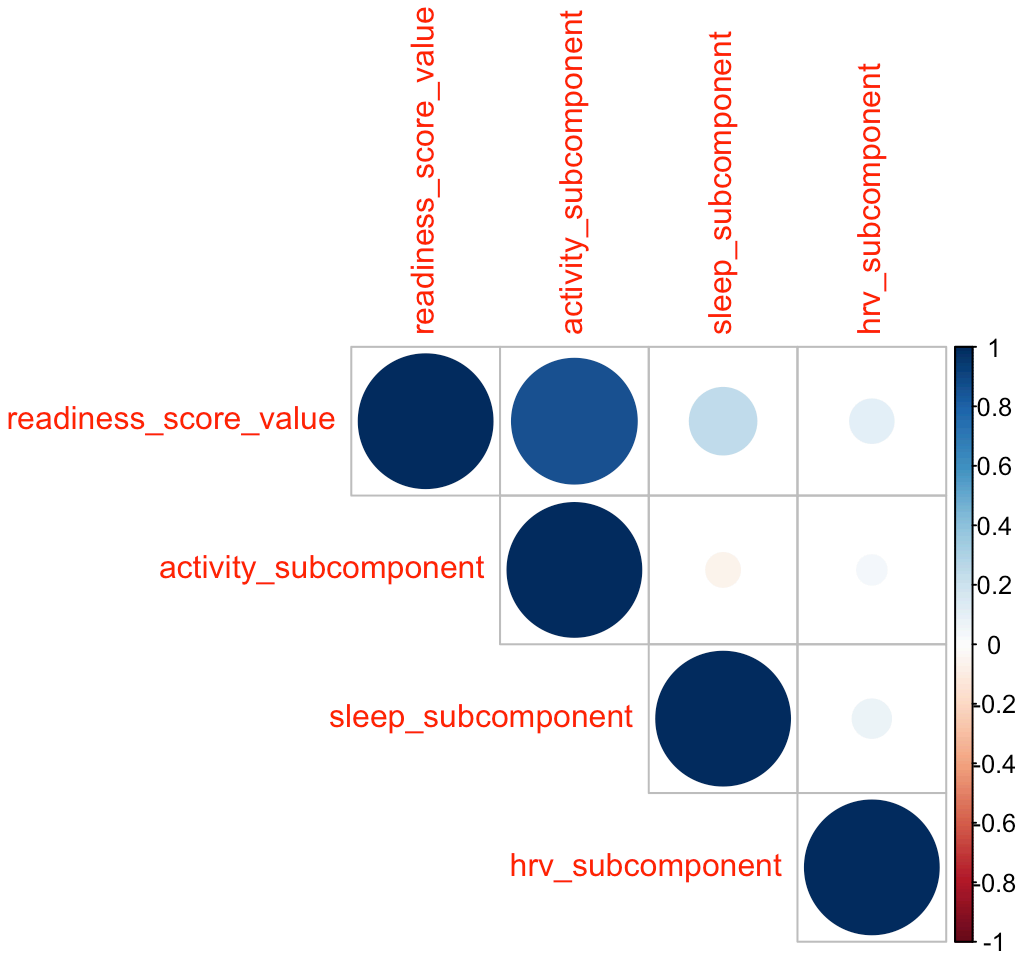
The scatter plot shows the relationship between the activity subcomponent scores and the overall readiness score values. Here are the key observations:

1. **Positive Linear Relationship**: There is a clear positive linear trend for most of the data points, indicating that higher activity subcomponent scores generally correspond to higher readiness scores. This suggests that increased activity levels might contribute positively to overall readiness.

2. **Cluster of Outliers**: There is a distinct cluster of outliers at higher activity subcomponent scores (near 100) where the readiness scores are significantly lower than the trend would suggest. This could indicate scenarios where high activity levels do not translate to high readiness, possibly due to overtraining or insufficient recovery.

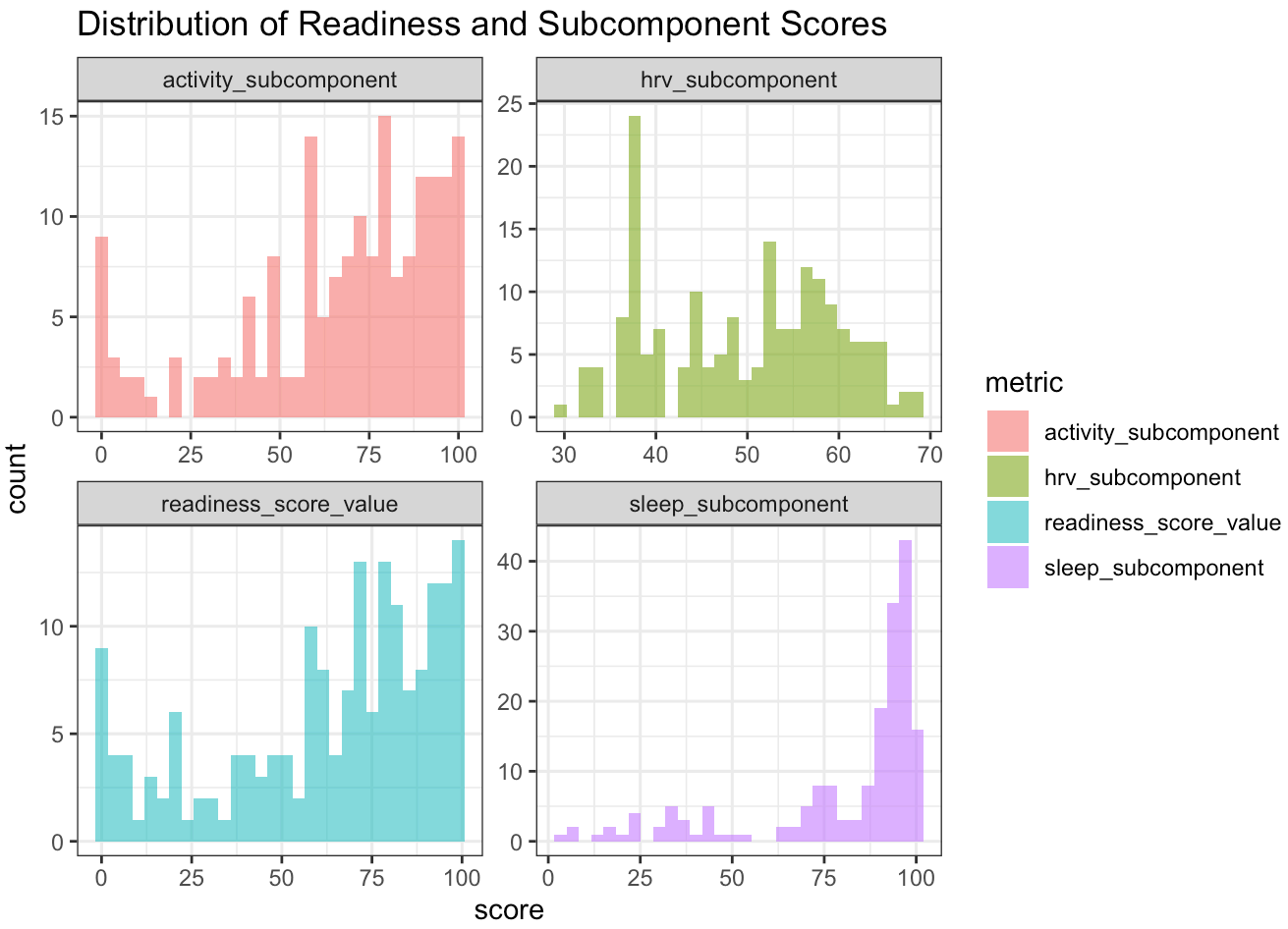
3. **Data Clustering**: The data points are densely packed along the diagonal, which strengthens the indication of a strong linear relationship between these two variables.

```
# Correlation plot of the data
corrplot(cor(data[, sapply(data, is.numeric)]), type = "upper")
```



1. **Strong positive correlations** between readiness scores and both activity and sleep subcomponents, indicating that higher activity levels and better sleep quality are associated with higher readiness scores.
2. **Moderate correlations** between sleep quality and HRV, as well as between activity and sleep, suggesting interdependencies among these factors.
3. **Weaker correlation** between activity levels and HRV, implying less direct association between these two metrics.

```
# Convert 'date' to Date type
data$date <- ymd(data$date)

# Histograms of scores
data %>%
  select(readiness_score_value, activity_subcomponent, sleep_subcomponent, hrv_subcompone
  pivot_longer(cols = c(readiness_score_value, activity_subcomponent, sleep_subcomponent,
               names_to = "metric", values_to = "score") %>%
  ggplot(aes(x = score, fill = metric)) +
  geom_histogram(bins = 30, alpha = 0.6) +
  facet_wrap(~metric, scales = "free") +
  theme_bw() +
  labs(title = "Distribution of Readiness and Subcomponent Scores")
```

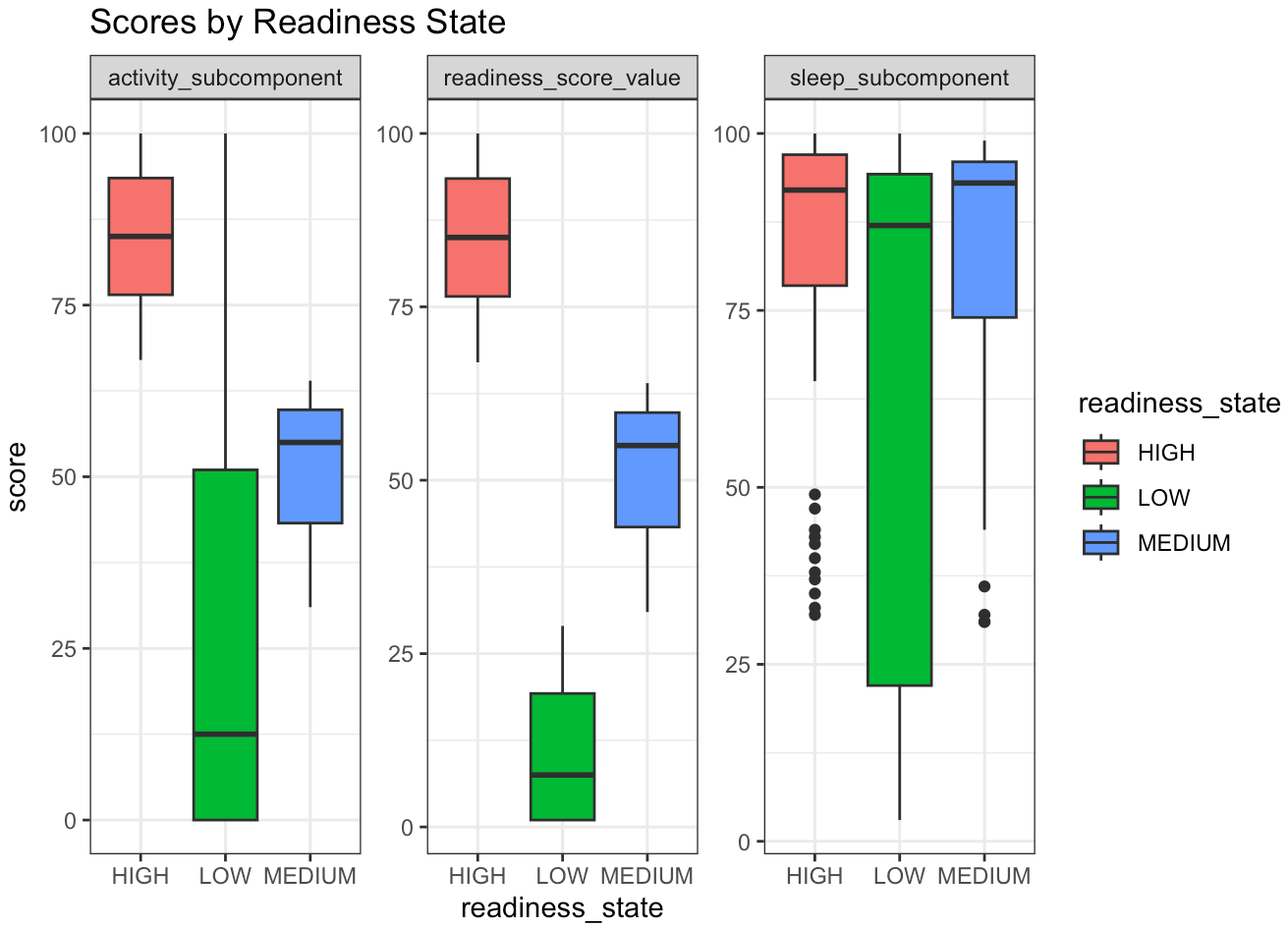


1. **Readiness Score (Red)**: The distribution shows a bimodal pattern, with peaks at the lower and higher ends of the score range, indicating that most scores are either quite low or quite high.
2. **Activity Subcomponent (Cyan)**: This histogram also exhibits a bimodal distribution, with significant groupings at lower and higher score values, suggesting periods of either low or high activity levels.
3. **HRV Subcomponent (Green)**: The distribution is more uniform but slightly skewed towards higher values, suggesting that most HRV scores are moderate to high.
4. **Sleep Subcomponent (Purple)**: Shows a heavily right-skewed distribution, indicating that high scores are much more prevalent, which could suggest generally good sleep quality among the subjects.

Each histogram highlights different aspects of the data, revealing variations in how each subcomponent contributes to overall readiness scores.

```
# Box plots of scores by readiness_state
data %>%
  select(readiness_score_value, activity_subcomponent, sleep_subcomponent, readiness_stat
  pivot_longer(cols = c(readiness_score_value, activity_subcomponent, sleep_subcomponent)
               names_to = "metric", values_to = "score", values_drop_na = TRUE) %>%
  ggplot(aes(x = readiness_state, y = score, fill = readiness_state)) +
  geom_boxplot() +
```

```
  facet_wrap(~metric, scales = "free") +
  theme_bw() +
  labs(title = "Scores by Readiness State")
```



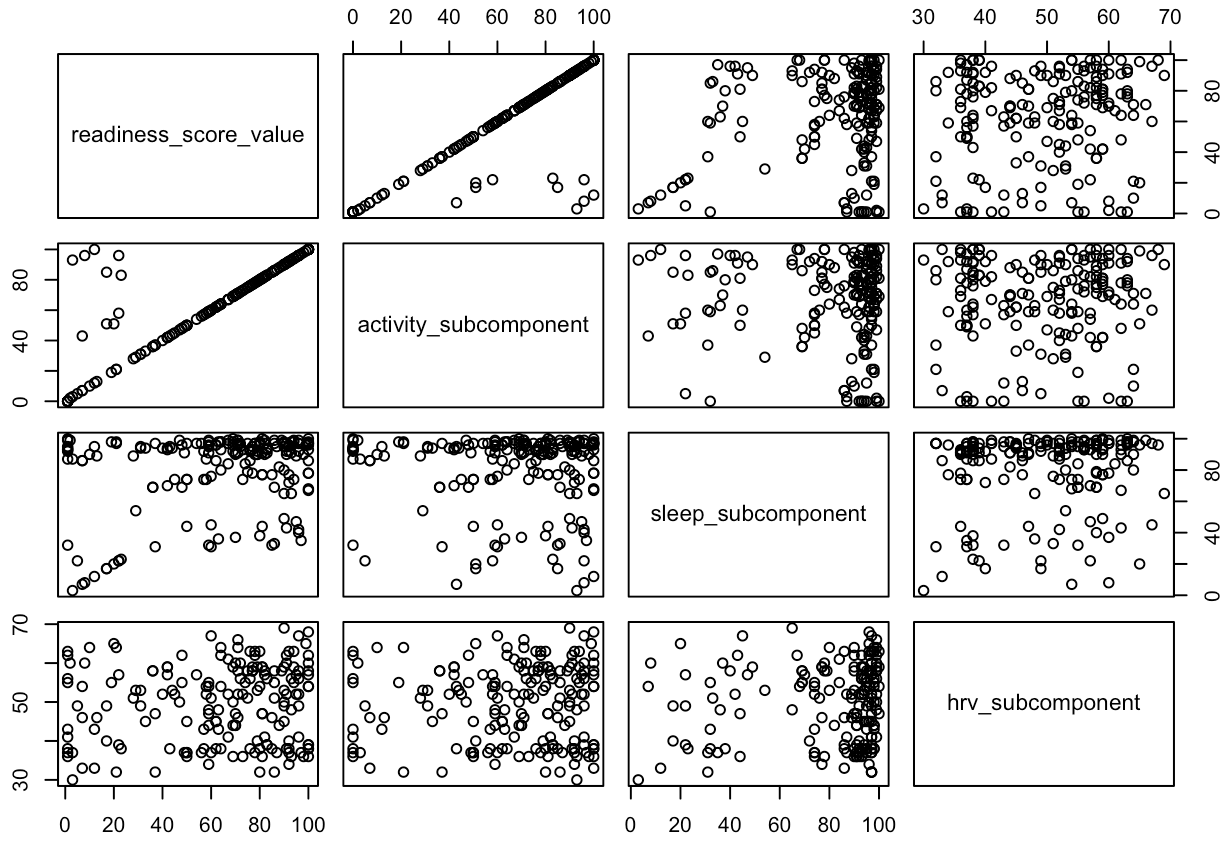
The box plot shows the distribution of scores for different readiness subcomponents across three readiness states: high, medium, and low.

1. **Activity Subcomponent**:
   - High readiness state has high activity scores, mostly around 75-100.
   - Medium readiness shows moderate activity scores.
   - Low readiness state is associated with the lowest activity scores, widely spread from 0 to around 50.
2. **Readiness Score Value**:
   - High readiness scores are tightly clustered at high values.
   - Medium and low readiness scores show a broader range, with low scores extending to very low values.
3. **Sleep Subcomponent**:
   - High readiness is correlated with high sleep scores.
   - Medium and low readiness states show lower sleep scores, with medium showing a wider range than high.

This visualization clearly demonstrates that higher values in the activity and sleep subcomponents are associated with higher overall readiness states. Conversely, lower readiness states correlate with lower

scores in these subcomponents.

```
# Create a pairs plot for selected numeric variables
pairs(select(data, readiness_score_value, activity_subcomponent, sleep_subcomponent, hrv_
```



The scatterplot matrix shows the relationships between readiness scores, activity, sleep, and HRV subcomponents:

1. **Readiness Score**: Strong positive correlations with both activity and sleep subcomponents, indicating higher scores are associated with increased activity and better sleep.
2. **Activity and Sleep**: No clear correlation, suggesting activity levels do not consistently predict sleep quality.
3. **HRV**: Shows weak or no clear correlations with other metrics, indicating it may be influenced by different factors not captured here.

This visualization highlights the importance of activity and sleep in influencing overall readiness while showing that HRV behaves independently of these factors.

```
library(tidymodels)
```

```
── Attaching packages ──────────────────────────────────── tidymodels 1.2.0 ──
```

```
✔ broom        1.0.5      ✔ recipes      1.0.10
✔ dials        1.2.1      ✔ tune         1.2.0
✔ infer        1.0.6      ✔ workflows    1.1.4
✔ modeldata    1.3.0      ✔ workflowsets 1.1.0
✔ parsnip      1.2.1      ✔ yardstick    1.3.1

── Conflicts ──────────────────────────────────────── tidymodels_conflicts() ──
✖ yardstick::accuracy()    masks forecast::accuracy()
✖ randomForest::combine()  masks dplyr::combine()
✖ scales::discard()        masks purrr::discard()
✖ tidyr::expand()          masks Matrix::expand()
✖ dplyr::filter()          masks stats::filter()
✖ recipes::fixed()         masks stringr::fixed()
✖ dplyr::lag()             masks stats::lag()
✖ purrr::lift()            masks caret::lift()
✖ randomForest::margin()   masks ggplot2::margin()
✖ tidyr::pack()            masks Matrix::pack()
✖ purrr::partial()         masks pdp::partial()
✖ yardstick::precision()   masks caret::precision()
✖ dials::prune()           masks rpart::prune()
✖ yardstick::recall()      masks caret::recall()
✖ yardstick::sensitivity() masks caret::sensitivity()
✖ yardstick::spec()        masks readr::spec()
✖ yardstick::specificity() masks caret::specificity()
✖ recipes::step()          masks stats::step()
✖ tidyr::unpack()          masks Matrix::unpack()
✖ recipes::update()        masks Matrix::update(), stats::update()
• Use tidymodels_prefer() to resolve common conflicts.
```

```r
set.seed(123)

# Step 1: Identify Numerical Columns
numerical_cols <- names(data)[sapply(data, is.numeric)]

# Step 2: Subset the Data (Only Numerical Predictors)
data_numeric <- data[, numerical_cols]

# Step 3: Create the Split
split <- initial_split(data_numeric, prop = 0.8)

# Step 4: Get Train and Test Sets
train_data <- training(split)
test_data <- testing(split)
```

Modeling

Linear Regression

```r
# 1. Fit the Model (assuming your data is in a dataframe named 'data')
lm_fit <- lm(readiness_score_value ~ activity_subcomponent + sleep_subcomponent + hrv_sub
```

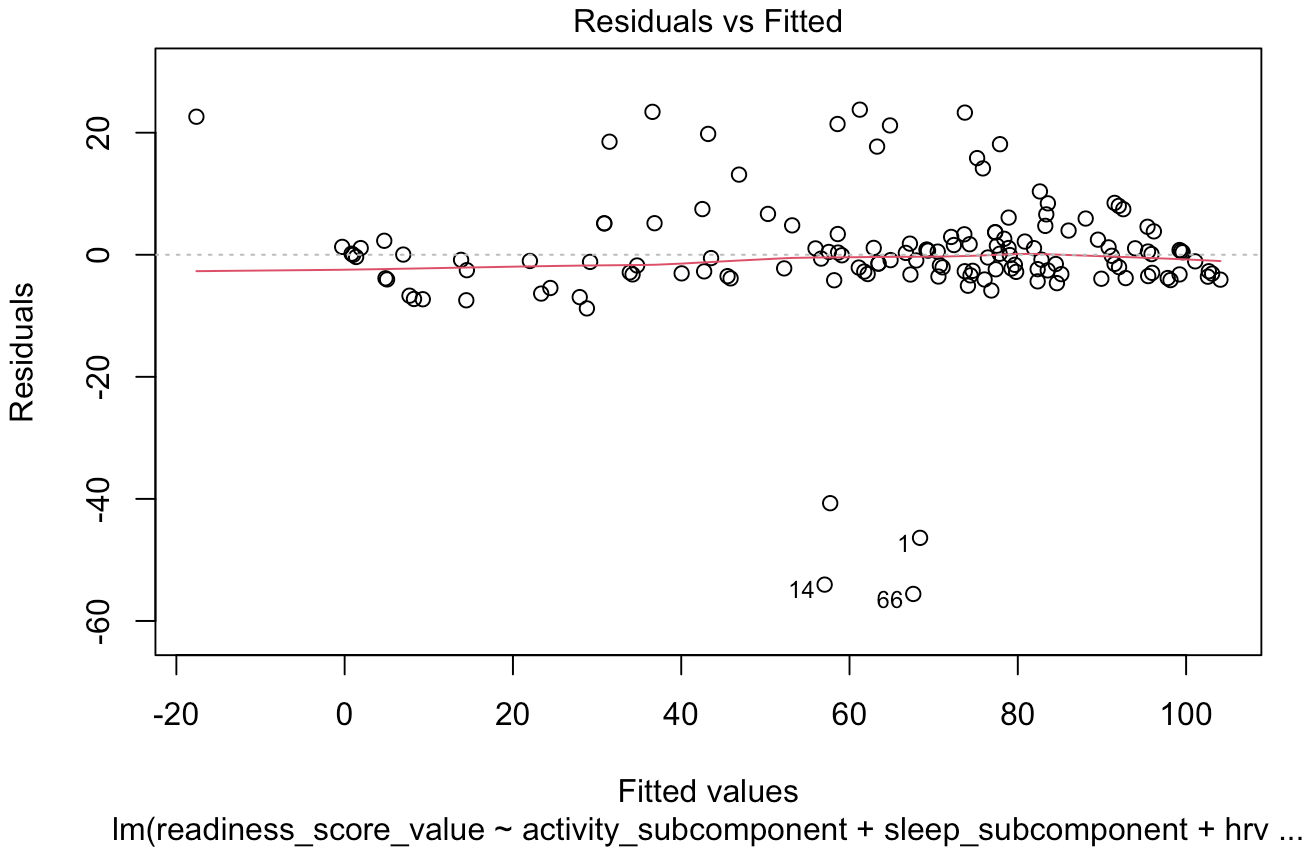
```r
# 2. Save Predictions
lm_predictions <- predict(lm_fit, newdata = test_data)  # Predictions for all data points

# 3. Basic Model Evaluation

# a. R-squared (coefficient of determination)
lm_rsquared <- summary(lm_fit)$r.squared
print(paste0("R-squared: ", lm_rsquared))
```

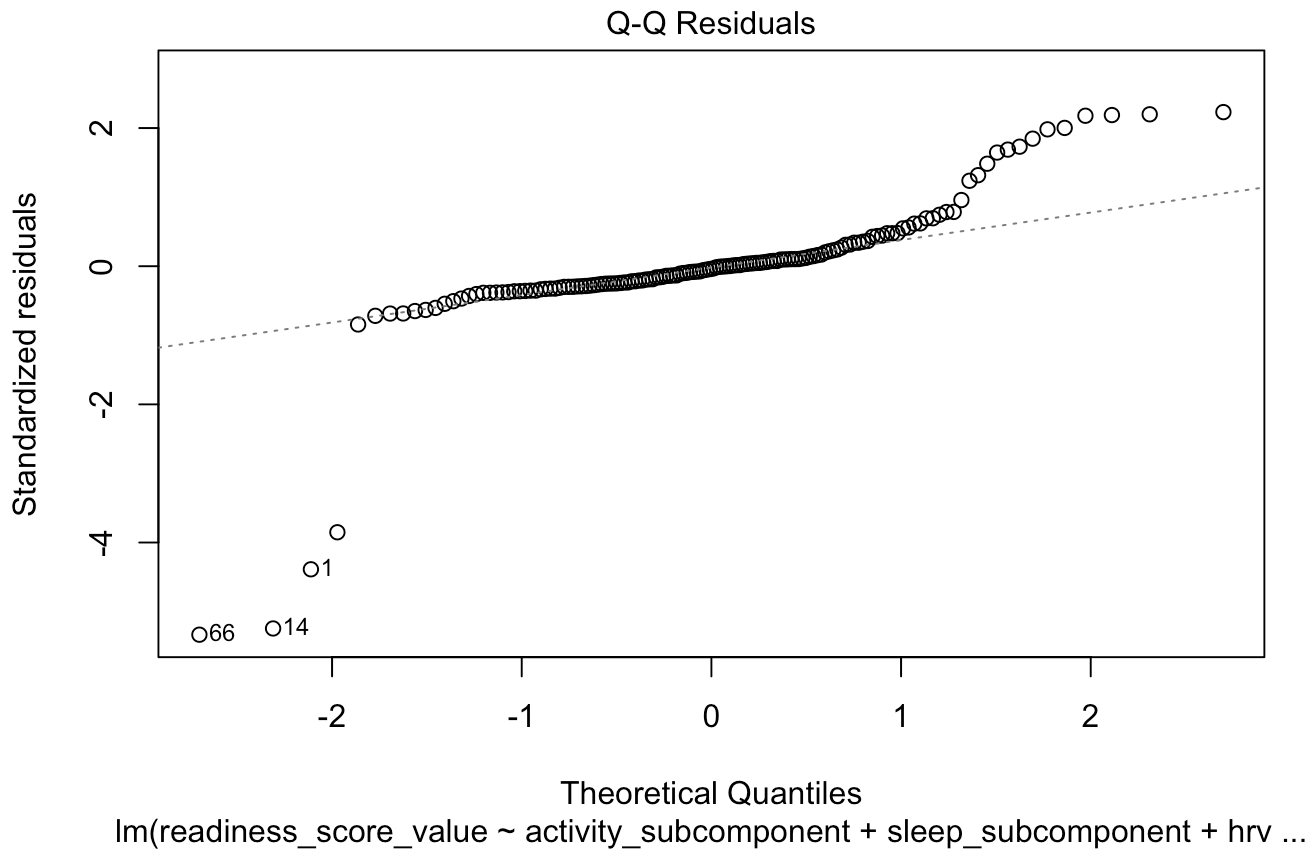
```
[1] "R-squared: 0.875073241073246"
```

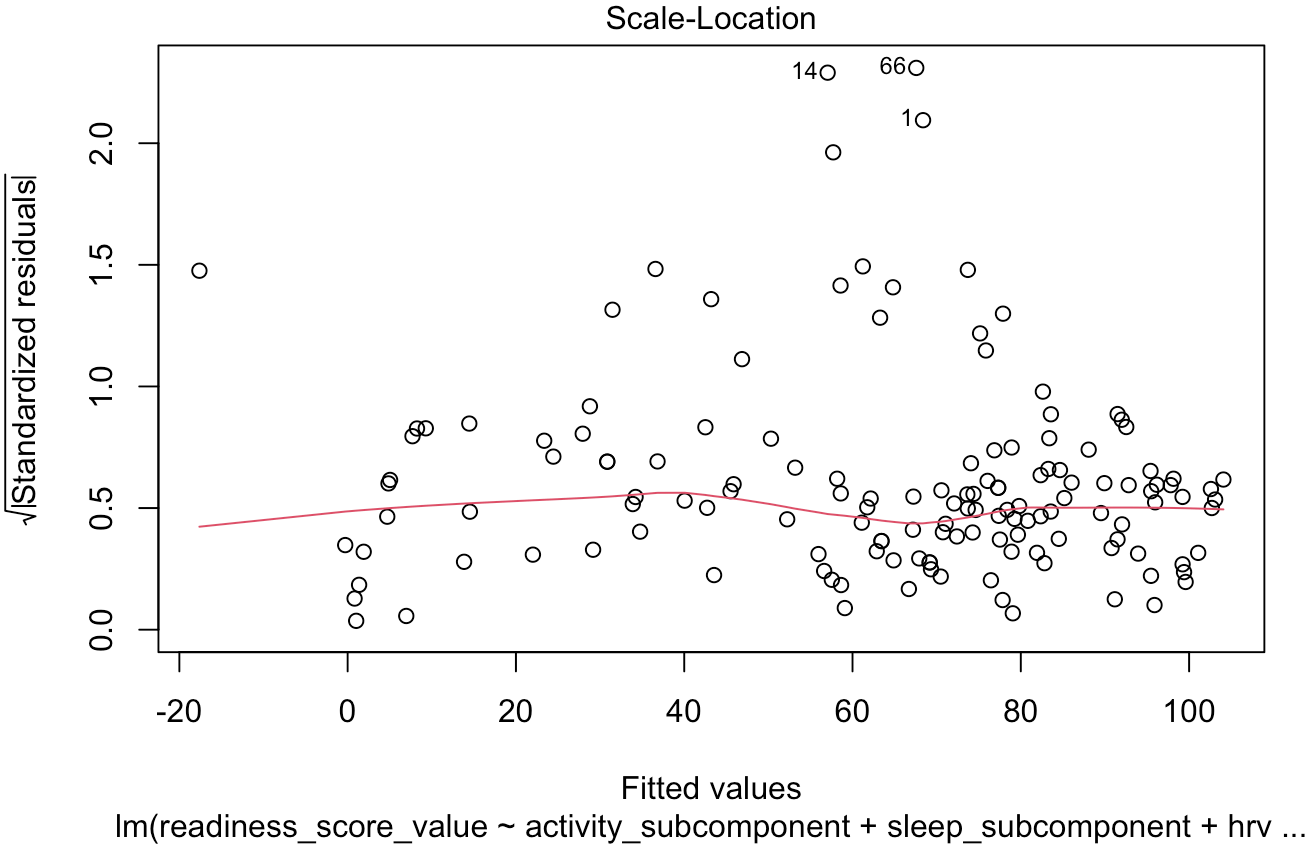
```r
# b. Root Mean Squared Error (RMSE)
# Assuming you have the actual 'readiness_score_value'
lm_rmse <- sqrt(mean((lm_predictions - test_data$readiness_score_value)^2))
print(paste0("RMSE: ", lm_rmse))
```

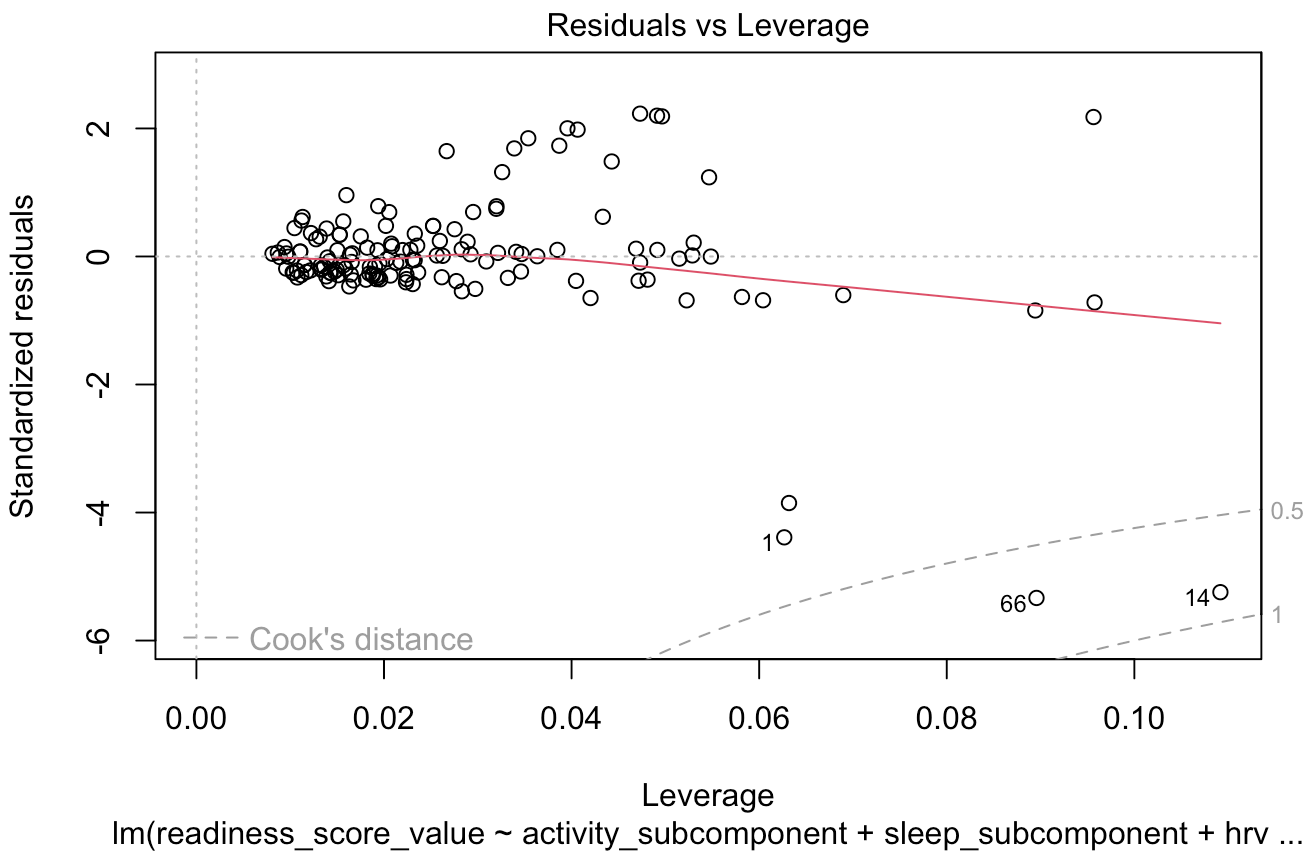
```
[1] "RMSE: 14.3700716254851"
```

```r
plot(lm_fit)
```




Q-Q Residuals
Standardized residuals
2
0
-2
-4
1
66
14
-2
-1
0
1
2
Theoretical Quantiles
lm(readiness_score_value ~ activity_subcomponent + sleep_subcomponent + hrv ...


Scale-Location
√|Standardized residuals|
14
66
1
2.0
1.5
1.0
0.5
0.0
-20
0
20
40
60
80
100
Fitted values
lm(readiness_score_value ~ activity_subcomponent + sleep_subcomponent + hrv ...




```
summary(lm_fit)
```

```
Call:
lm(formula = readiness_score_value ~ activity_subcomponent +
    sleep_subcomponent + hrv_subcomponent, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-55.574  -3.108  -0.404   2.691  23.777

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)          -38.92069    5.80140  -6.709 4.48e-10 ***
activity_subcomponent   0.96414    0.03175  30.364  < 2e-16 ***
sleep_subcomponent      0.36214    0.03980   9.100 8.15e-16 ***
hrv_subcomponent        0.17377    0.09428   1.843   0.0674 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.92 on 140 degrees of freedom
Multiple R-squared:  0.8751,    Adjusted R-squared:  0.8724
F-statistic: 326.9 on 3 and 140 DF,  p-value: < 2.2e-16
```

Coefficients

The coefficients represent the estimated effect of each predictor on the readiness score:

- For every one unit increase in `activity_subcomponent`, the readiness score is expected to increase by approximately 0.96414 units.
- For every one unit increase in `sleep_subcomponent`, the readiness score is expected to increase by approximately 0.36214 units.
- The coefficient for `hrv_subcomponent` is not statistically significant at conventional levels (p-value > 0.05), suggesting that its effect on readiness score may not be reliable.

Model Performance

- **R-squared**: 0.8751
- **Adjusted R-squared**: 0.8724
- **Residual standard error**: 10.92
- **F-statistic**: 326.9 on 3 and 140 degrees of freedom
- **p-value**: < 2.2e-16

The high R-squared value indicates that a large proportion of the variability in readiness score is explained by the predictors.

Polynomial regression model

```
# Fit a polynomial regression model
poly_fit <- lm(readiness_score_value ~ poly(activity_subcomponent, 2) +
               sleep_subcomponent + hrv_subcomponent, data = train_data)
```

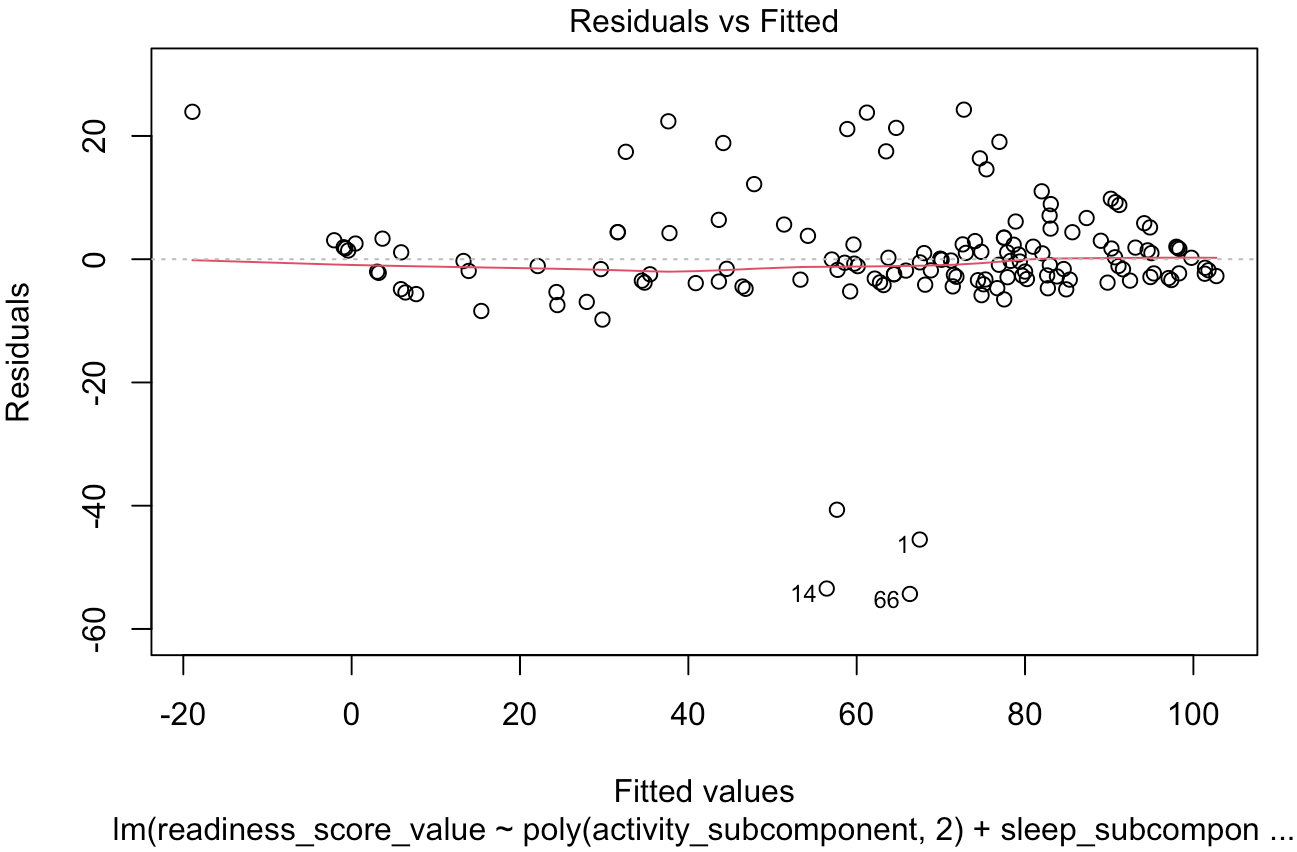
```
# Make predictions on the test set
poly_predictions <- predict(poly_fit, newdata = test_data)

# Calculate Root Mean Squared Error (RMSE)
poly_rmse <- sqrt(mean((poly_predictions - test_data$readiness_score_value)^2))
cat("Root Mean Squared Error on test set: ", poly_rmse, "\n")
```

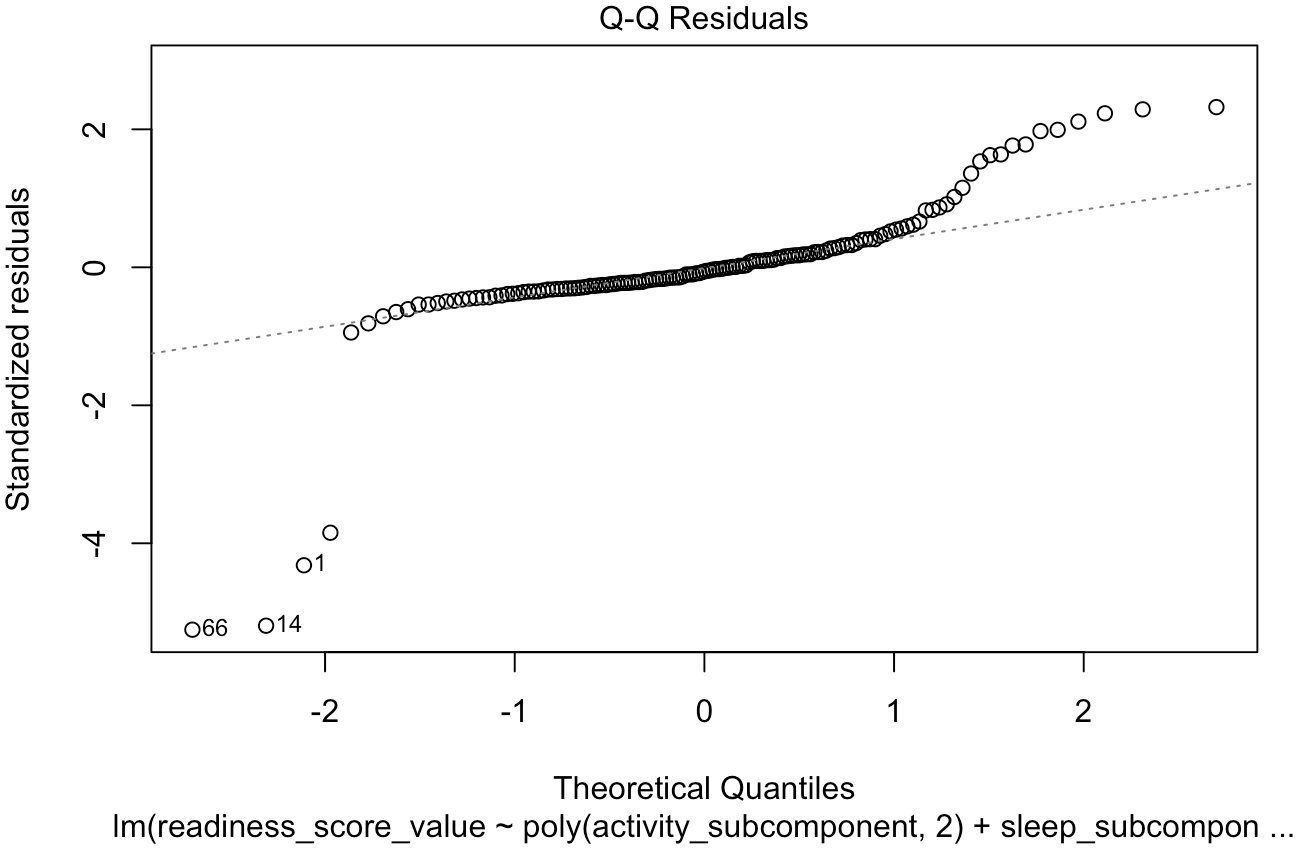
```
Root Mean Squared Error on test set:  14.32769
```

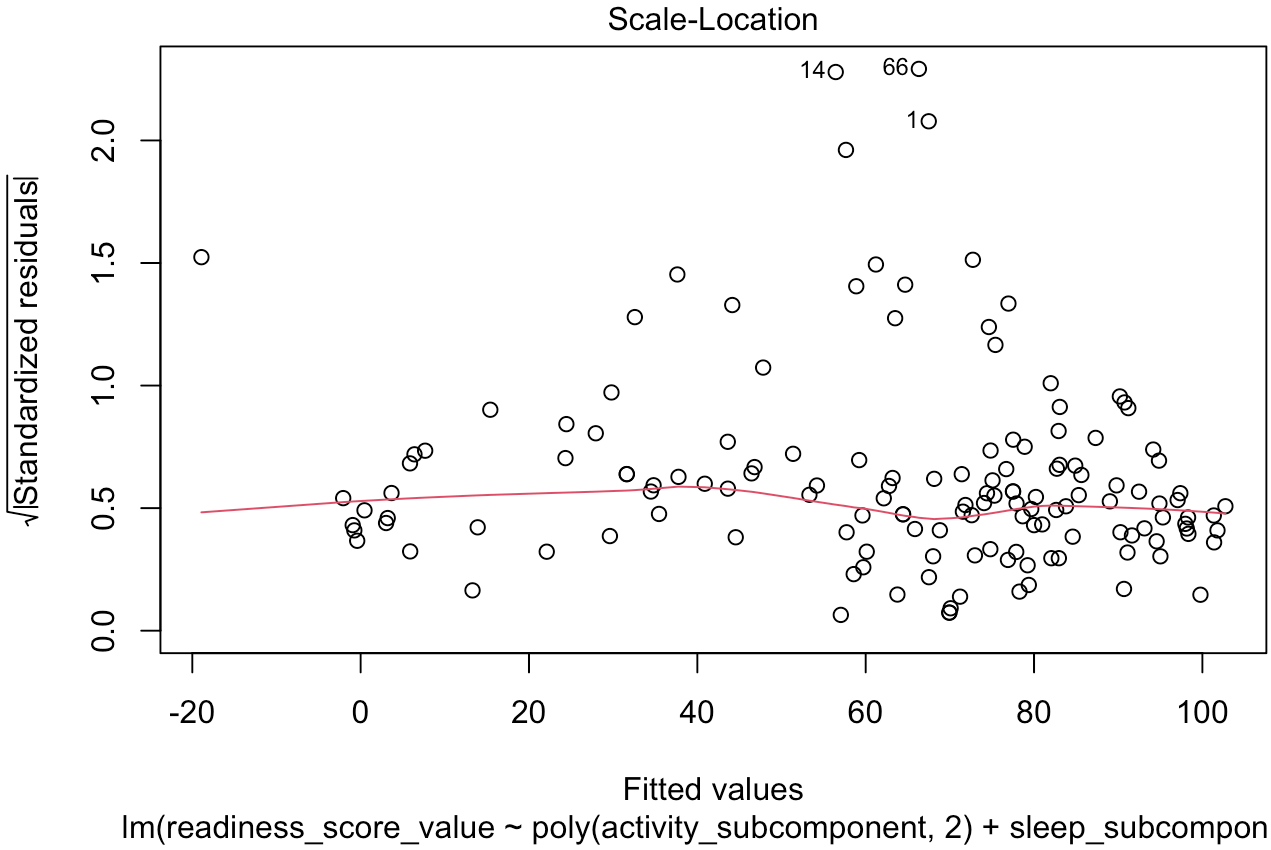
```
# R-squared (coefficient of determination)
poly_rsquared <- summary(poly_fit)$r.squared
print(paste0("R-squared: ", poly_rsquared))
```

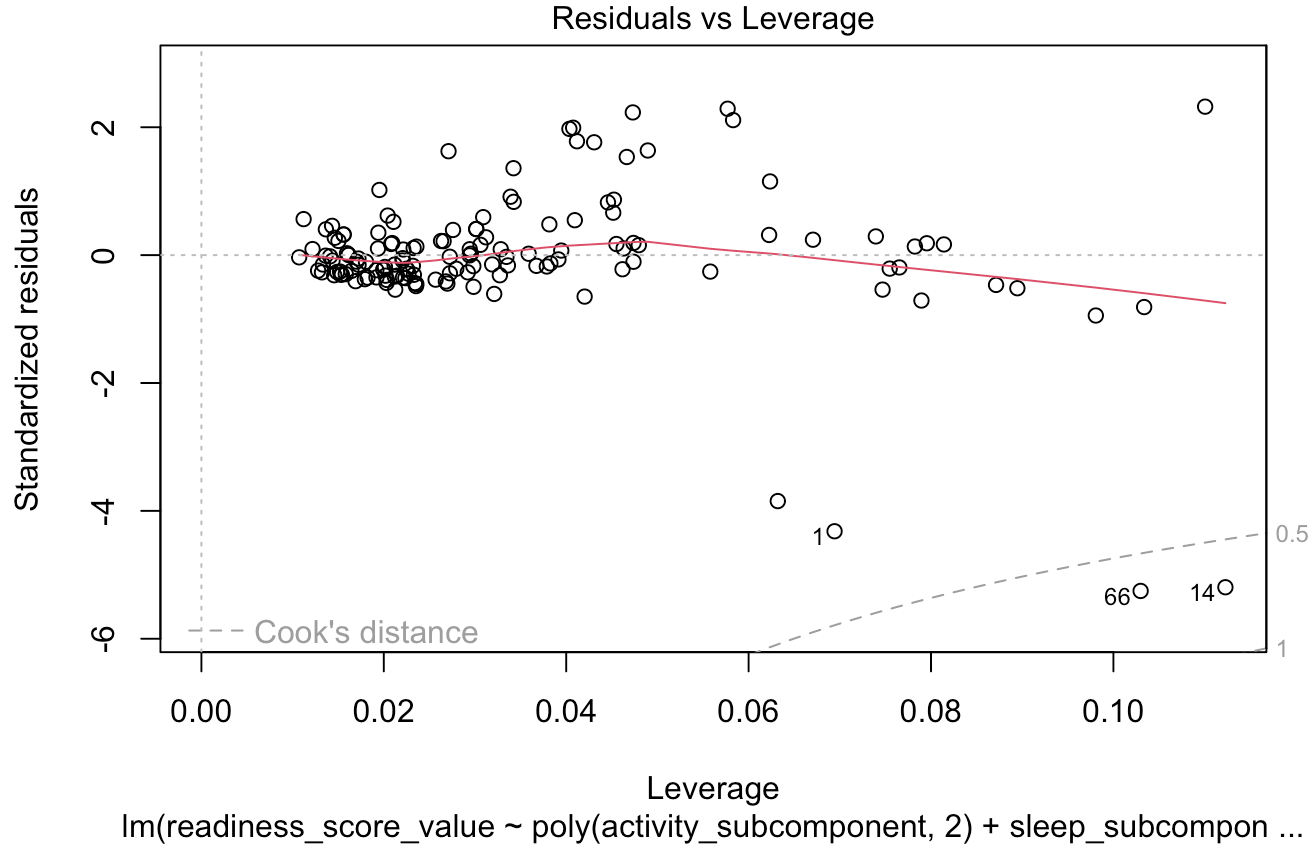
```
[1] "R-squared: 0.875952269755738"
```

```
# Plot the polynomial fit
plot(poly_fit)
```


Residuals vs Fitted
Residuals
20
0
-20
-40
-60
-20
0
20
40
60
80
100
1
14
66
Fitted values
lm(readiness_score_value ~ poly(activity_subcomponent, 2) + sleep_subcompon ...


Q-Q Residuals
Standardized residuals
2
0
-2
-4
-2
-1
0
1
2
1
66
14
Theoretical Quantiles
lm(readiness_score_value ~ poly(activity_subcomponent, 2) + sleep_subcompon ...


Scale-Location
√|Standardized residuals|
14
66
1
0.0
0.5
1.0
1.5
2.0
-20
0
20
40
60
80
100
Fitted values
lm(readiness_score_value ~ poly(activity_subcomponent, 2) + sleep_subcompon ...


Residuals vs Leverage
Standardized residuals
2
0
-2
-4
-6
1
66
14
Cook's distance
0.5
1
0.00
0.02
0.04
0.06
0.08
0.10
Leverage
lm(readiness_score_value ~ poly(activity_subcomponent, 2) + sleep_subcompon ...

```r
# Summary of the polynomial fit
summary(poly_fit)
```

```
Call:
lm(formula = readiness_score_value ~ poly(activity_subcomponent,
    2) + sleep_subcomponent + hrv_subcomponent, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-54.319  -3.239  -0.646   2.952  24.267

Coefficients:
                                  Estimate Std. Error t value Pr(>|t|)
(Intercept)                       25.01852    5.41989   4.616 8.81e-06 ***
poly(activity_subcomponent, 2)1 334.84824   11.02718  30.366  < 2e-16 ***
poly(activity_subcomponent, 2)2 -10.84293   10.92527  -0.992   0.3227
sleep_subcomponent                0.36239    0.03980   9.106 8.28e-16 ***
hrv_subcomponent                  0.17071    0.09433   1.810   0.0725 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10.92 on 139 degrees of freedom
Multiple R-squared:  0.876, Adjusted R-squared:  0.8724
F-statistic: 245.4 on 4 and 139 DF,  p-value: < 2.2e-16
```

Coefficients

The coefficients represent the estimated effect of each predictor on the readiness score:

- For every one unit increase in `activity_subcomponent`, the readiness score is expected to increase by approximately 334.84824 units when the relationship is linear (poly(activity_subcomponent, 2)1).
- The quadratic term (poly(activity_subcomponent, 2)2) is not statistically significant at conventional levels (p-value > 0.05), suggesting that the relationship between `activity_subcomponent` and readiness score may not be well described by a quadratic function.
- For every one unit increase in `sleep_subcomponent`, the readiness score is expected to increase by approximately 0.36239 units.
- The coefficient for `hrv_subcomponent` is marginally statistically significant (p-value = 0.0725), suggesting a possible but weak effect on readiness score.

Model Performance

- **R-squared**: 0.876
- **Adjusted R-squared**: 0.8724
- **Residual standard error**: 10.92
- **F-statistic**: 245.4 on 4 and 139 degrees of freedom
- **p-value**: < 2.2e-16

The high R-squared value indicates that a large proportion of the variability in readiness score is explained by the predictors.

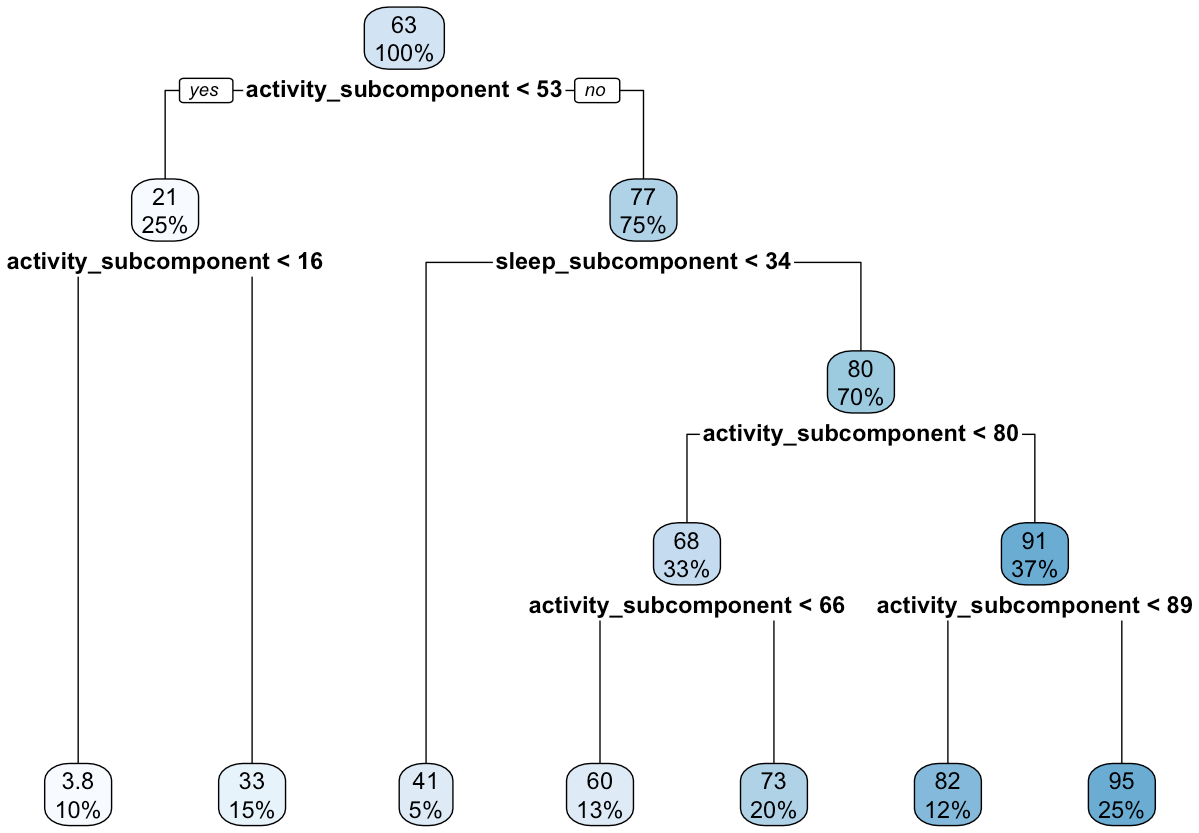
Decision Tree

```
# Fit the decision tree model (using default parameters)
dt_fit <- rpart(readiness_score_value ~ . , data = train_data, method = "anova")
```

```
# Predictions on the test set
dt_predictions <- predict(dt_fit, newdata = test_data)

# Wrapper function
predict_dt_wrapper <- function(object, newdata) {
  predict(object, newdata = test_data, type="response")
}

# 1. Calculate Squared Errors
squared_errors <- (dt_predictions - test_data$readiness_score_value)^2

# 2. Calculate Mean Squared Error (MSE)
dt_rmse <- mean(squared_errors)

# Print the calculated RMSE
print(paste0("Decision tree RMSE: ", round(dt_rmse, digits = 3)))
```

```
[1] "Decision tree RMSE: 102.829"
```

```
# Visualize the tree
rpart.plot(dt_fit)
```




```
summary(dt_fit)
```

```
Call:
rpart(formula = readiness_score_value ~ ., data = train_data,
    method = "anova")
  n= 144

          CP nsplit  rel error    xerror       xstd
1 0.64575099      0 1.00000000 1.0193957 0.10200414
2 0.08942784      1 0.35424901 0.3687318 0.06332577
3 0.05638847      3 0.17539334 0.3260996 0.08852764
4 0.01496090      4 0.11900486 0.2168624 0.07810262
5 0.01476045      5 0.10404396 0.1934371 0.07835836
6 0.01000000      6 0.08928351 0.1886996 0.07841701

Variable importance
activity_subcomponent    sleep_subcomponent      hrv_subcomponent
                   87                    11                     3

Node number 1: 144 observations,    complexity param=0.645751
  mean=63.3125, MSE=927.8121
  left son=2 (36 obs) right son=3 (108 obs)
  Primary splits:
```

```
      activity_subcomponent < 52.5 to the left,  improve=0.64575100, (0 missing)
      sleep_subcomponent    < 26.5 to the left,  improve=0.16128660, (0 missing)
      hrv_subcomponent      < 35   to the left,  improve=0.03217051, (0 missing)

Node number 2: 36 observations,    complexity param=0.05638847
  mean=20.91667, MSE=296.7986
  left son=4 (15 obs) right son=5 (21 obs)
  Primary splits:
      activity_subcomponent < 16   to the left,  improve=0.70509640, (0 missing)
      sleep_subcomponent    < 80   to the right, improve=0.07747369, (0 missing)
      hrv_subcomponent      < 44   to the left,  improve=0.02264910, (0 missing)
  Surrogate splits:
      sleep_subcomponent < 98.5 to the right, agree=0.667, adj=0.200, (0 split)
      hrv_subcomponent   < 44   to the left,  agree=0.639, adj=0.133, (0 split)

Node number 3: 108 observations,    complexity param=0.08942784
  mean=77.44444, MSE=339.3025
  left son=6 (7 obs) right son=7 (101 obs)
  Primary splits:
      sleep_subcomponent    < 34   to the left,  improve=0.27557210, (0 missing)
      activity_subcomponent < 74.5 to the left,  improve=0.25896390, (0 missing)
      hrv_subcomponent      < 49.5 to the left,  improve=0.06111314, (0 missing)
  Surrogate splits:
      hrv_subcomponent < 33.5 to the left,  agree=0.944, adj=0.143, (0 split)

Node number 4: 15 observations
  mean=3.8, MSE=16.02667

Node number 5: 21 observations
  mean=33.14286, MSE=138.5986

Node number 6: 7 observations
  mean=40.71429, MSE=1077.633

Node number 7: 101 observations,    complexity param=0.08942784
  mean=79.9901, MSE=188.1484
  left son=14 (48 obs) right son=15 (53 obs)
  Primary splits:
      activity_subcomponent < 79.5 to the left,  improve=0.72608320, (0 missing)
      sleep_subcomponent    < 78.5 to the right, improve=0.03450201, (0 missing)
      hrv_subcomponent      < 48.5 to the left,  improve=0.02965485, (0 missing)
  Surrogate splits:
      sleep_subcomponent < 78.5 to the right, agree=0.594, adj=0.146, (0 split)
      hrv_subcomponent   < 48.5 to the left,  agree=0.545, adj=0.042, (0 split)

Node number 14: 48 observations,    complexity param=0.01476045
  mean=67.70833, MSE=53.37326
  left son=28 (19 obs) right son=29 (29 obs)
  Primary splits:
      activity_subcomponent < 65.5 to the left,  improve=0.76976320, (0 missing)
      hrv_subcomponent      < 57.5 to the left,  improve=0.13797020, (0 missing)
```

```
      sleep_subcomponent    < 82   to the left,  improve=0.02600416, (0 missing)
  Surrogate splits:
      sleep_subcomponent < 76   to the left,  agree=0.688, adj=0.211, (0 split)
      hrv_subcomponent   < 39   to the left,  agree=0.646, adj=0.105, (0 split)

Node number 15: 53 observations,    complexity param=0.0149609
  mean=91.11321, MSE=49.87398
  left son=30 (17 obs) right son=31 (36 obs)
  Primary splits:
      activity_subcomponent < 88.5 to the left,  improve=0.756189000, (0 missing)
      hrv_subcomponent      < 59.5 to the left,  improve=0.034178020, (0 missing)
      sleep_subcomponent    < 94.5 to the left,  improve=0.009834095, (0 missing)

Node number 28: 19 observations
  mean=59.78947, MSE=7.955679

Node number 29: 29 observations
  mean=72.89655, MSE=15.12723

Node number 30: 17 observations
  mean=82.17647, MSE=4.968858

Node number 31: 36 observations
  mean=95.33333, MSE=15.55556
```

Model Complexity Parameters

The complexity parameter (CP) is used to control the size of the tree. As the tree grows larger, the CP decreases, indicating an increase in model complexity. The xerror column represents the cross-validated prediction error.

CP	nsplit	rel error	xerror	xstd
0.64575099	0	1.000000	1.0193957	0.10200414
0.08942784	1	0.354249	0.3687318	0.06332577
0.05638847	3	0.175393	0.3260996	0.08852764
0.01496090	4	0.119004	0.2168624	0.07810262
0.01476045	5	0.104044	0.1934371	0.07835836
0.01000000	6	0.089284	0.1886996	0.07841701

Variable Importance

The importance of each predictor variable in the model is indicated by the percentage of times it is chosen for splitting nodes in the tree.

Variable	Importance
activity_subcomponent	87%
sleep_subcomponent	11%
hrv_subcomponent	3%

Nodes

The tree structure is represented by nodes, each containing a subset of observations. Each node is split into two child nodes based on a selected predictor variable and split point.

- Node number 1: The root node with 144 observations.
- Node number 2: Represents a subset of observations (36) with a mean readiness score of 20.92.
- Node number 3: Represents a larger subset of observations (108) with a mean readiness score of 77.44.
- Nodes 4 to 31: Further splits of the data based on predictor variables.

Mean Squared Error (MSE)

The mean squared error (MSE) is used to evaluate the goodness of fit of the model within each node. It represents the average squared difference between predicted and actual readiness scores within a node.

Random Forest

```
# Fit the Random Forest model
rf_fit <- randomForest(readiness_score_value ~ ., data = train_data)

# Predictions on the test set
rf_predictions <- predict(rf_fit, newdata = test_data)

# Calculate Mean Squared Error (MSE) for Random Forest
rf_squared_errors <- (rf_predictions - test_data$readiness_score_value)^2
rf_rmse <- mean(rf_squared_errors)
print(paste0("Random Forest RMSE: ", round(rf_rmse, digits = 3)))
```

```
[1] "Random Forest RMSE: 77.964"
```

```
# 1. Print the model summary
print(rf_fit)
```

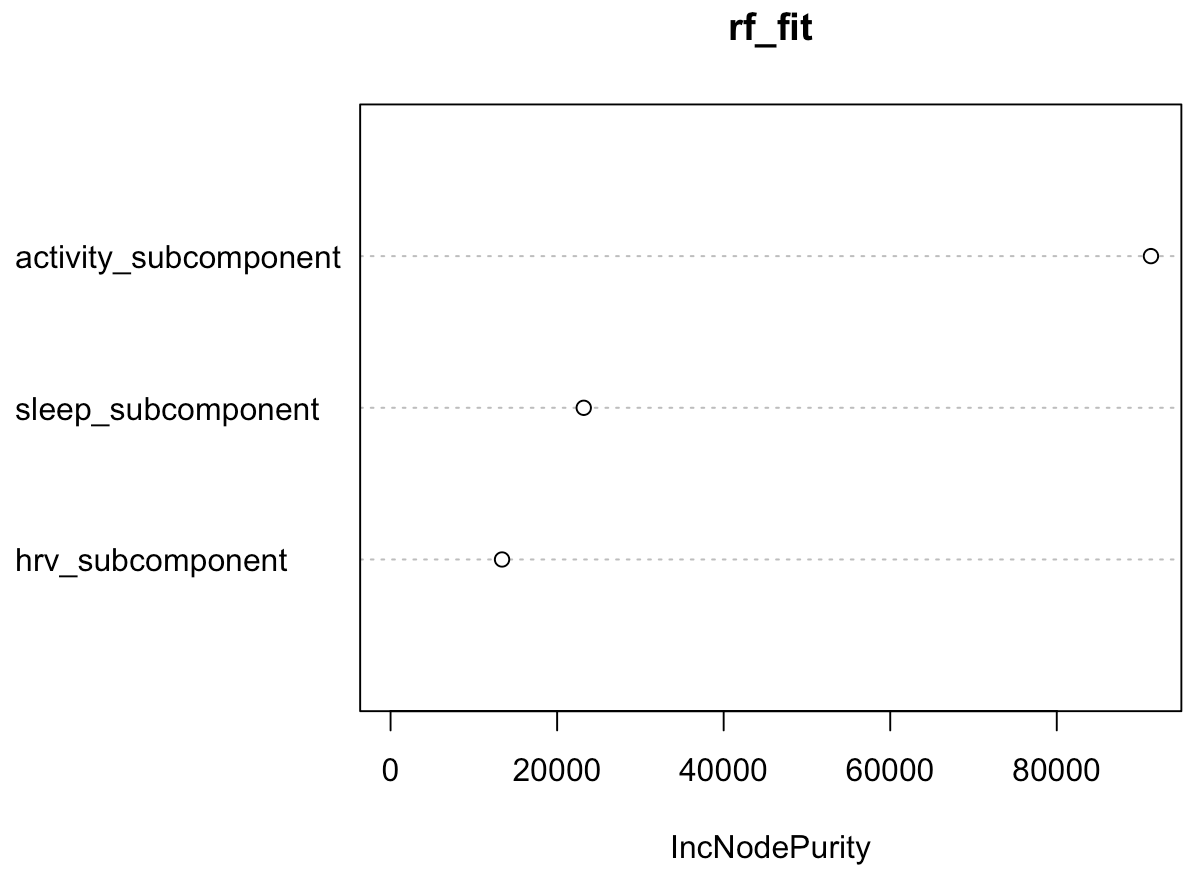
```
Call:
 randomForest(formula = readiness_score_value ~ ., data = train_data)
               Type of random forest: regression
```

```
                    Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 56.9829
                    % Var explained: 93.86
```

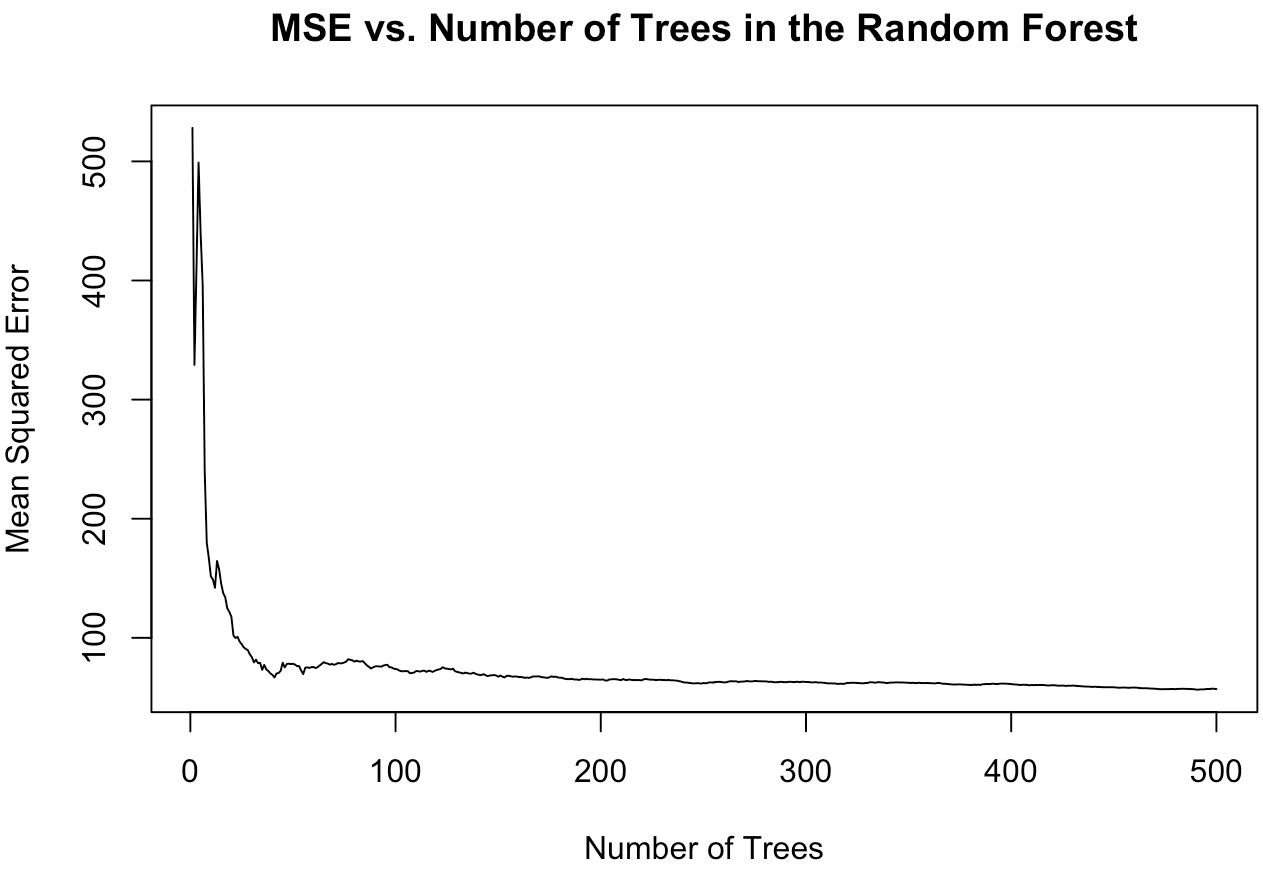
```r
# 2. Show variable importance
importance(rf_fit)
```

```
                      IncNodePurity
activity_subcomponent      91307.38
sleep_subcomponent         23196.82
hrv_subcomponent           13385.59
```

```r
# 3. Plot variable importance
varImpPlot(rf_fit)
```




```r
# 4. Plot MSE or RMSE over the number of trees
plot(rf_fit$mse, type = "l", xlab = "Number of Trees", ylab = "Mean Squared Error",
     main = "MSE vs. Number of Trees in the Random Forest")
```



- **Type**: Regression random forest.
- **Trees**: 500 trees used.
- **Variables per Split**: 1 variable tried at each split.
- **MSE**: Mean of squared residuals is 56.9829.
- **Explained Variance**: 93.86% of the variance in the target variable is explained, indicating high accuracy.
- **Variable Importance**:
  - **activity_subcomponent** is the most influential.
  - **sleep_subcomponent** and **hrv_subcomponent** also contribute significantly, though less than the activity component.

Knn

```
# Model Fitting (Illustrative 'k')
knn_fit <- train(readiness_score_value ~ . , data = train_data,
                 method = "knn",
                 trControl = trainControl(method = "none"), # No resampling yet
                 tuneGrid = data.frame(k = 5)) # Experiment with different 'k' values

# Predictions on the test set
knn_predictions <- predict(knn_fit, newdata = test_data)
```

```r
# Calculate RMSE
knn_rmse <- sqrt(mean((knn_predictions - test_data$readiness_score_value)^2))
print(paste0("RMSE (KNN): ", knn_rmse))
```

```
[1] "RMSE (KNN): 9.06234826941429"
```

```r
# Print the KNN model summary
print(summary(knn_fit))
```

```
            Length Class      Mode
learn       2      -none-     list
k           1      -none-     numeric
theDots     0      -none-     list
xNames      3      -none-     character
problemType 1      -none-     character
tuneValue   1      data.frame list
obsLevels   1      -none-     logical
param       0      -none-     list
```

```r
# Check the model's specific details, such as k value and other parameters
print(paste0("Number of Neighbors (k): ", knn_fit$bestTune$k))
```

```
[1] "Number of Neighbors (k): 5"
```

The KNN model with (k = 5) neighbors resulted in an RMSE of 9.062, indicating the average prediction error. The model uses three predictors. The summary confirms that (k = 5) was used, but does not provide detailed performance metrics or variable importance. This setup suggests the model performs reasonably well with the chosen parameters.

Model Selection

```r
# Assuming you have the following objects (adjust if needed):
model_predictions <- list(lm_predictions, poly_predictions, dt_predictions, knn_predictio
model_names <- c("Linear Model", "Polynomial  Model", "Decision Tree", "KNN", "Random For
model_rmses <- c(lm_rmse, poly_rmse, dt_rmse, knn_rmse, rf_rmse)

# 1. Create a Comparison Dataframe
comparison_df <- data.frame(model = model_names, rmse = model_rmses)

# 2. Arrange by RMSE (Ascending)
comparison_df <- comparison_df[order(comparison_df$rmse), ]

# 3. Print the Results
print(comparison_df)
```

```
              model       rmse
4               KNN   9.062348
```

```
2 Polynomial  Model  14.327693
1      Linear Model  14.370072
5     Random Forest  77.963570
3     Decision Tree 102.829179
```

Based on the RMSE (Root Mean Squared Error) values for each model:

1. **K-Nearest Neighbors (KNN)**:
   - RMSE: 9.062348
2. **Polynomial Model**:
   - RMSE: 14.327693
3. **Linear Model**:
   - RMSE: 14.370072
4. **Random Forest**:
   - RMSE: 77.963570
5. **Decision Tree**:
   - RMSE: 102.829179

Lower RMSE values indicate better model performance in terms of prediction accuracy. Based on the RMSE values:

- **K-Nearest Neighbors (KNN)** has the lowest RMSE among all models, indicating it performs the best in terms of minimizing prediction errors on unseen data.
- The **Polynomial Model** and **Linear Model** have similar RMSE values, but the Polynomial Model slightly outperforms the Linear Model.
- **Random Forest** and **Decision Tree** models have significantly higher RMSE values compared to the other models, indicating poorer performance in terms of prediction accuracy.

Therefore, if the goal is to select the best-performing model based solely on RMSE, the K-Nearest Neighbors (KNN) model would be the preferred choice.

Results communication

Project Objective

The focus of this project was to analyze factors influencing readiness scores by employing various statistical and machine learning models. The dataset included multiple metrics related to daily readiness assessments.

Encountered Challenges

A significant challenge was the limited size of the dataset, containing just a small number of observations. Additionally, the data was too clean from the start, which sometimes can lead to models that are overly fit to a non-representative state of real-world data. This restriction was pivotal in shaping

the outcomes of the modeling efforts, predominantly leading to overfitting across different applied models.

Model Implementation

- **Linear and Polynomial Regression:** Aimed to identify both linear and nonlinear relationships within the data. However, the limited data led to polynomial models fitting excessively to noise.
- **K-Nearest Neighbors (KNN):** Dependent on the choice of neighbors and distance metrics, KNN struggled with parameter tuning due to dataset size limitations.
- **Random Forests and Decision Trees:** Known for handling complex dataset characteristics, both models nevertheless learned the data too precisely, mirroring training data patterns rather than learning to generalize.

Overfitting: A Recurring Issue

The models exhibited excellent performance metrics that were unusually high for training datasets but were indicative of overfitting. This was characterized by the models' tendencies to memorize the training data instead of understanding underlying patterns.

Key Insights

The analysis revealed that high accuracy on training data, while initially seeming positive, masked the models' poor generalization capabilities. This project underscores the need for a larger dataset or refined methodologies tailored for smaller datasets. Potential strategies include implementing data augmentation, opting for simpler models, and enhancing validation strategies to better assess model robustness.