Predicting Fitbit Sleep Score with Machine Learning

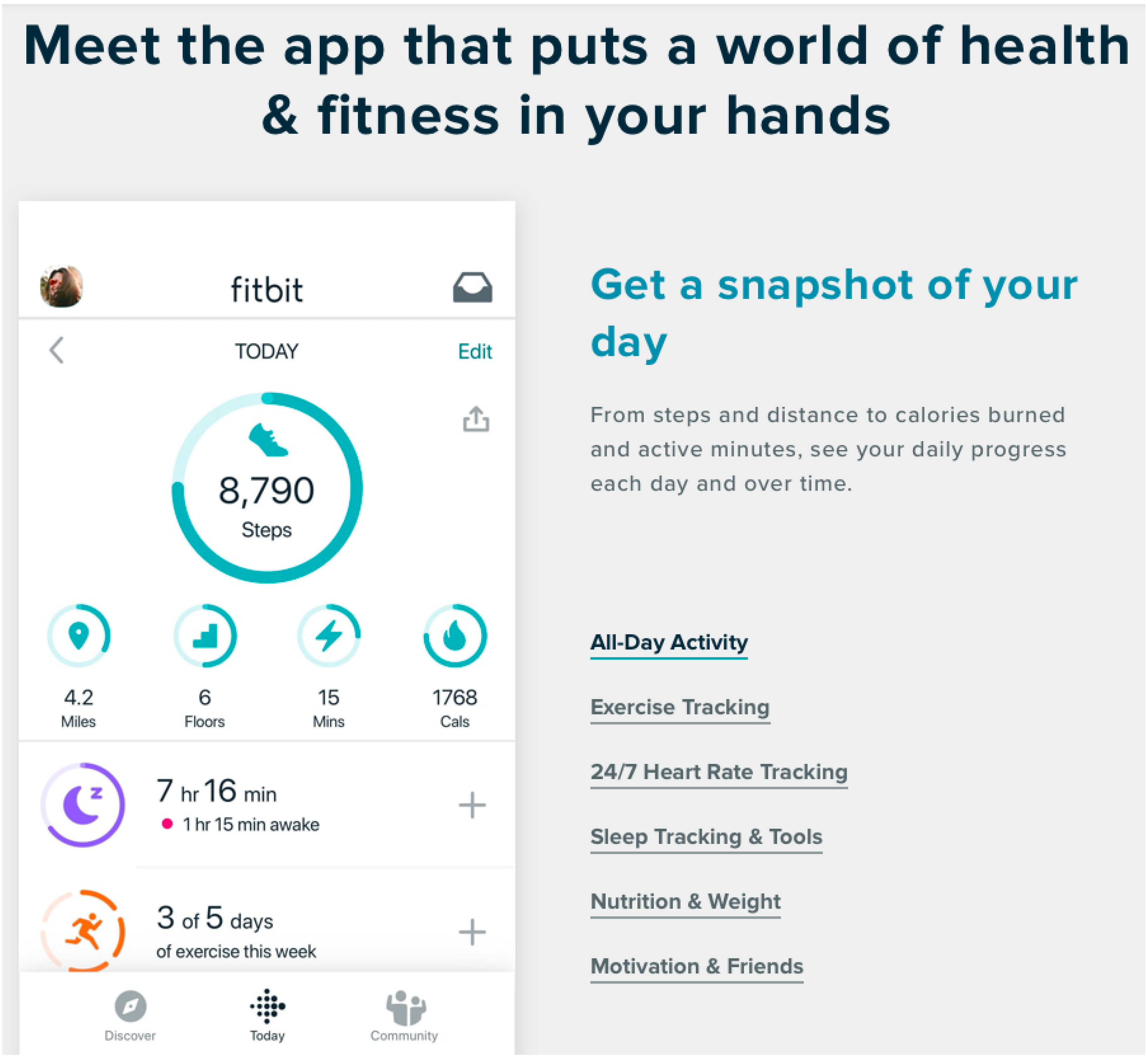
Amy Nguyen

Introduction

In this project, I will be implementing Machine Learning to create a model that predicts Fitbit Sleep Scores.

About Fitbit

Fitbit is one of the leading American consumer electronics company, best known for their fitness trackers and smartwatches. Alongside their line of products is their Fitbit mobile application where users can sync their Fitbit device via bluetooth.


Meet the app that puts a world of health & fitness in your hands
fitbit
TODAY
Edit
8,790 Steps
4.2 Miles
6 Floors
15 Mins
1768 Cals
7 hr 16 min
1 hr 15 min awake
3 of 5 days of exercise this week
Discover
Today
Community
Get a snapshot of your day
From steps and distance to calories burned and active minutes, see your daily progress each day and over time.
All-Day Activity
Exercise Tracking
24/7 Heart Rate Tracking
Sleep Tracking & Tools
Nutrition & Weight
Motivation & Friends

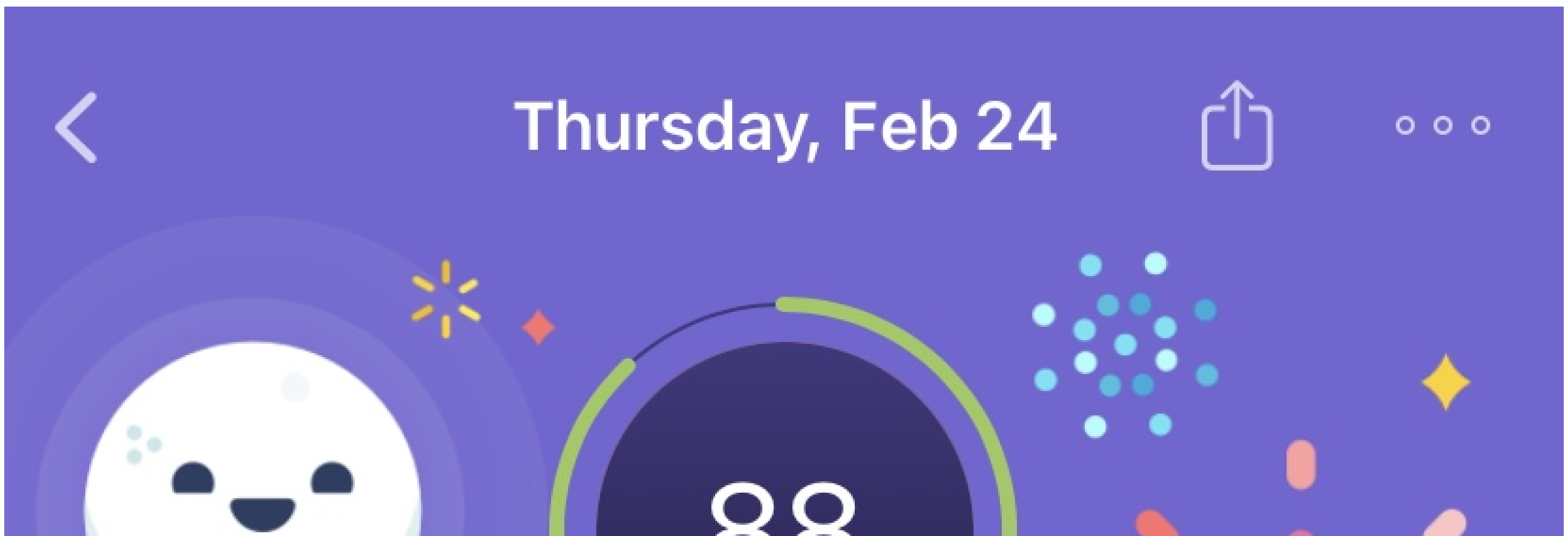
The app displays a detailed summary of the tracker's data such as daily steps, activity, calories, sleep, heart rate, etc. This project will primarily focus on Fitbit's sleep tracking capabilities, and more notably Sleep Score.

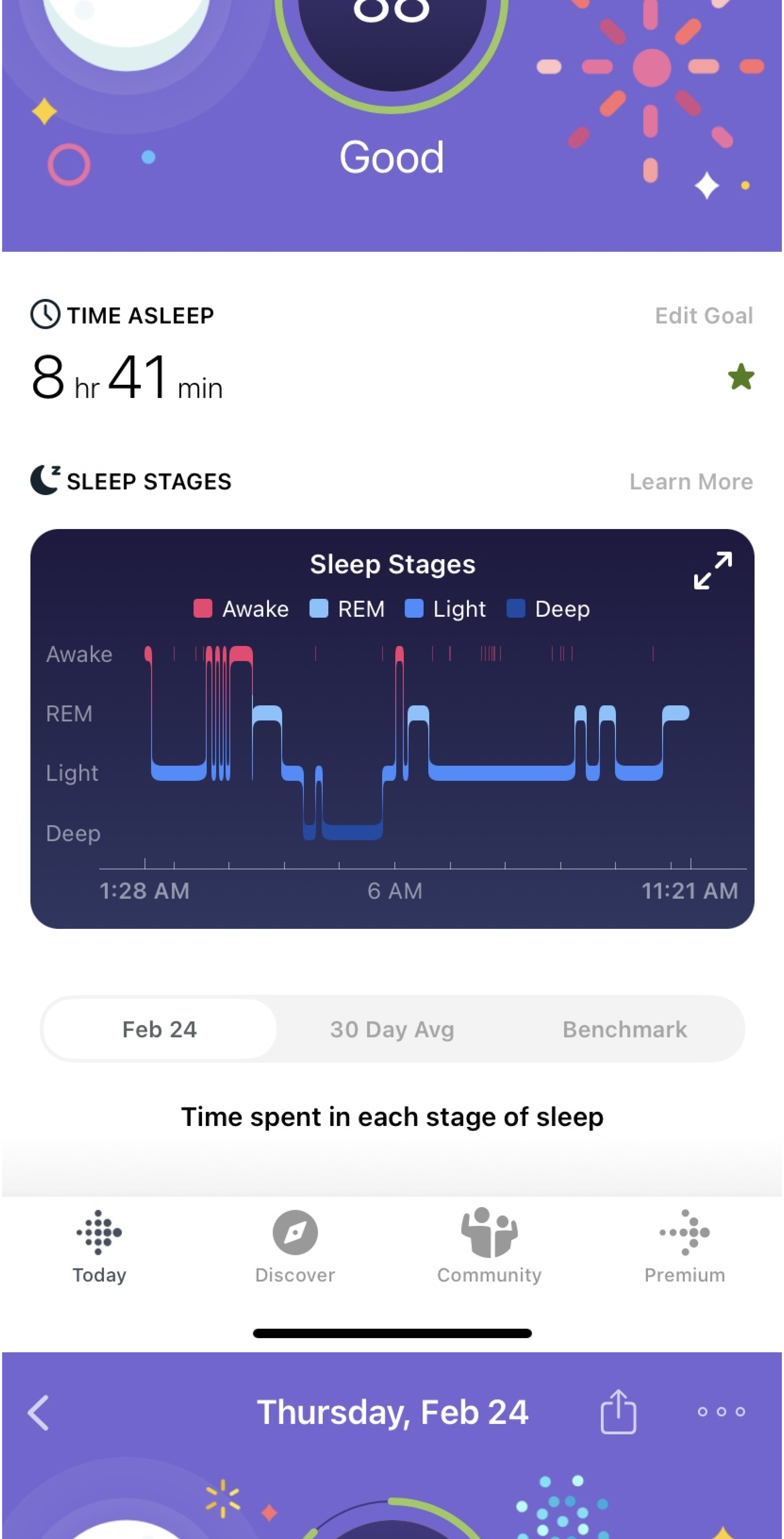
What is a Sleep Score?

A nightly sleep score is a score out of 100 based on the user's quality of sleep that night. If wearing the device to bed, Fitbit tracks a breakdown of the time spent in different stages of sleep. Within the app, users can view time asleep and time spent in each stage of sleep:

1. Awake
2. REM
3. Light
4. Deep

Below are screenshots from my own Fitbit app to better visualize the interface.





TIME ASLEEP Edit Goal

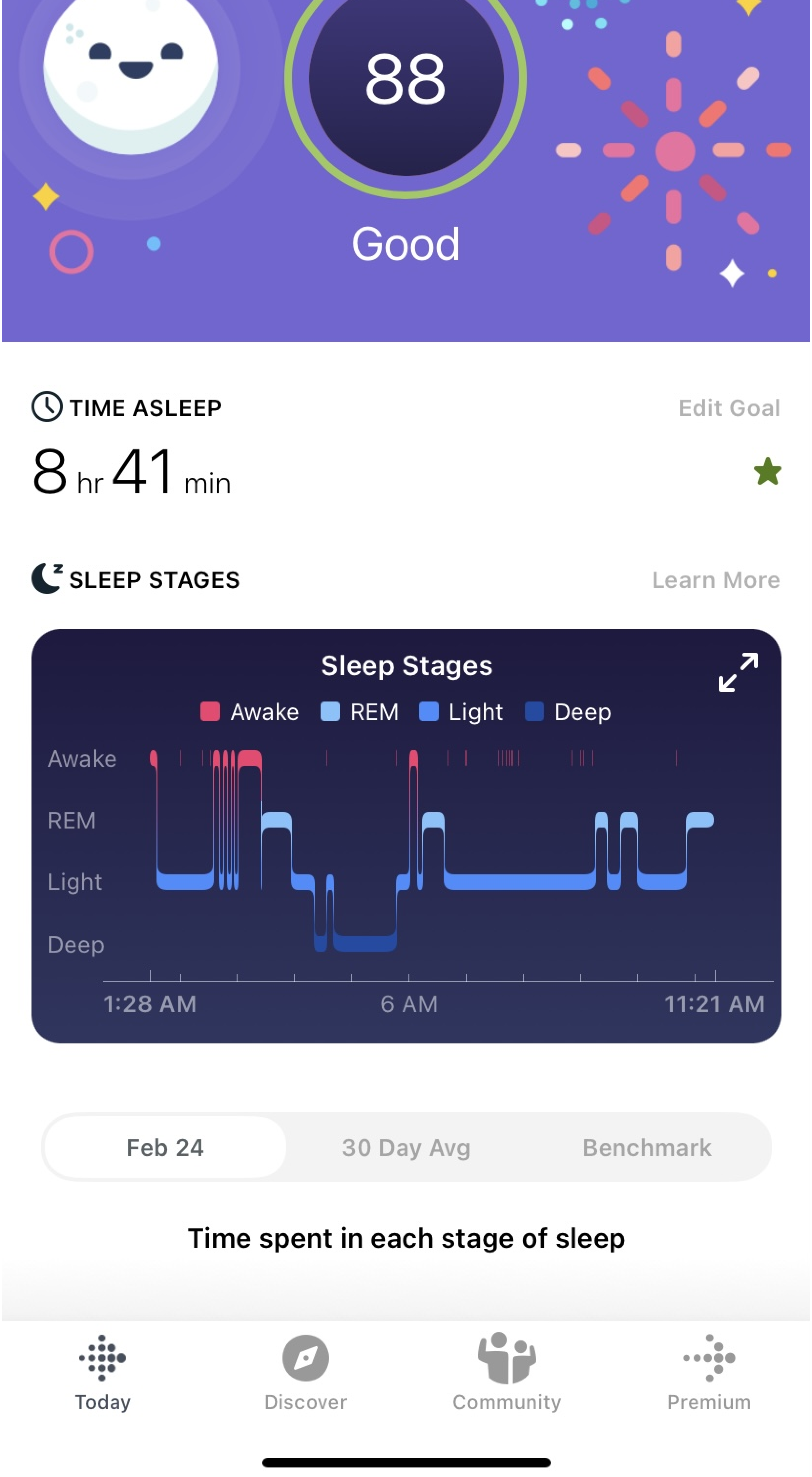
8 hr 41 min

SLEEP STAGES Learn More

Feb 24 | 30 Day Avg | Benchmark

Time spent in each stage of sleep

Thursday, Feb 24



88

Good

TIME ASLEEP

Edit Goal

8 hr 41 min

SLEEP STAGES

Learn More

Time spent in each stage of sleep

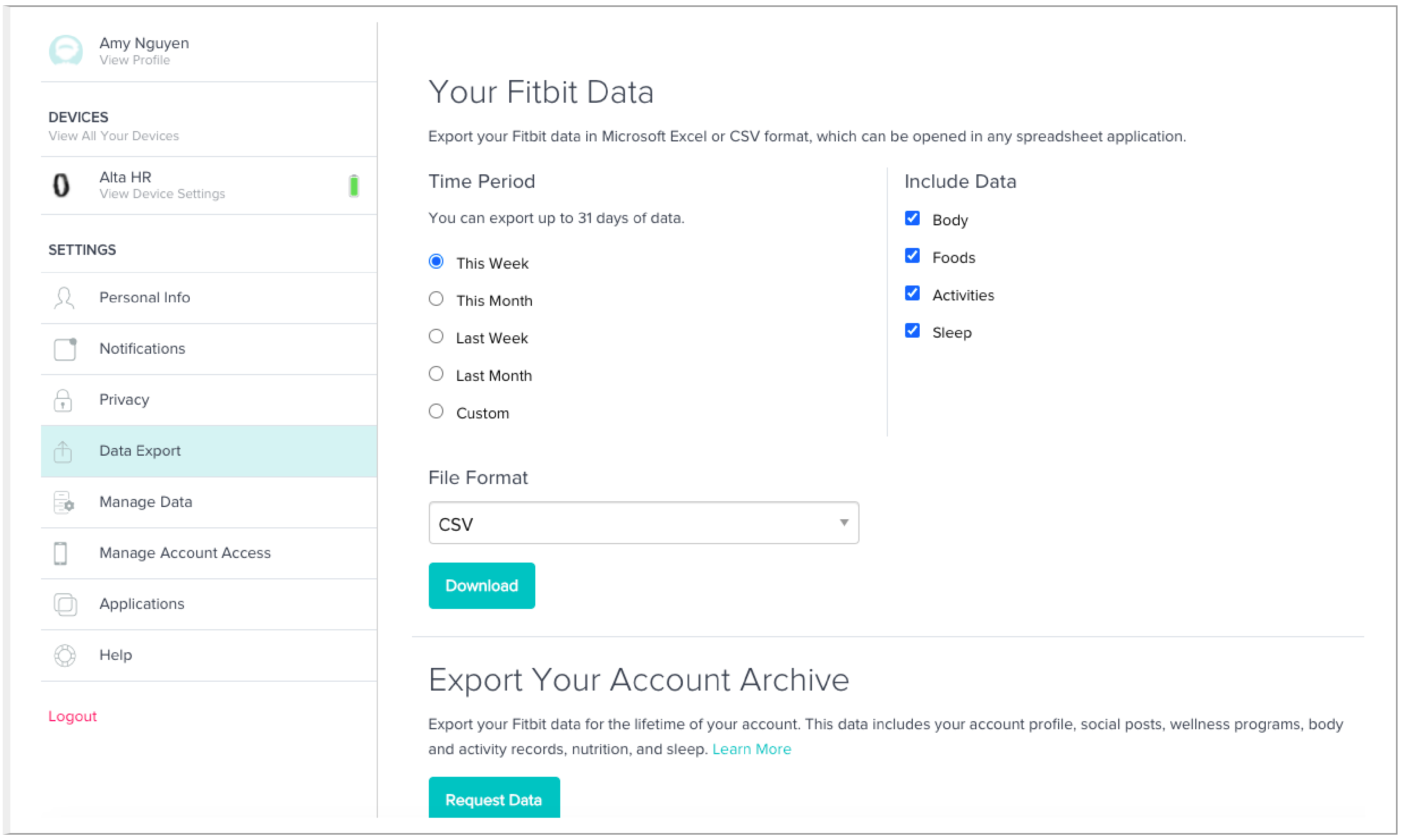
The company website (https://help.fitbit.com/articles/en_US/Help_article/2439.htm) defines sleep score:

"Your overall nightly sleep score is based on your heart rate, the time you spend awake or restless, and your sleep stages."

My Personal Dataset

Fitbit Alta HR

For this project, I chose to use my own personal dataset from my own Fitbit Alta HR device. While there are extensive Fitbit datasets available, I couldn't find any datasets on Fitbit users' sleep data with associated sleep scores. Fitbit allows users to download personal data collected by their Fitbit tracker into CSV files, which ultimately inspired me to analyze and build a model on my own personal Fitbit data.



Amy Nguyen
View Profile

DEVICES
View All Your Devices

Alta HR
View Device Settings

SETTINGS

Personal Info

Notifications

Privacy

Data Export

Manage Data

Manage Account Access

Applications

Help

Logout

Your Fitbit Data

Export your Fitbit data in Microsoft Excel or CSV format, which can be opened in any spreadsheet application.

Time Period

You can export up to 31 days of data.

- [✓] This Week
- This Month
- Last Week
- Last Month
- Custom

Include Data

- [✓] Body
- [✓] Foods
- [✓] Activities
- [✓] Sleep

File Format

CSV

Download

Export Your Account Archive

Export your Fitbit data for the lifetime of your account. This data includes your account profile, social posts, wellness programs, body and activity records, nutrition, and sleep. Learn More

Request Data

I was able to download my tracker's dataset on my sleep for the entire lifetime of my tracker, but this is where things got complicated and laborous. Frustratingly enough, the only relevant statistic provided was sleep score, resting heart rate, and restlessnessless. Other relevant data such as time asleep in the different stages of sleep was not part of this dataset. In order to retrieve this information,, I had to manually export my sleep data, but Fitbit only allows a maximum of 31 days of data to be exported a time. I exported the CSV files month by month (dating back to August 2019) and saved all the sleep statistics data saved into one CSV file as my_sleep.csv.

Data Cleaning & Wrangling

Now that I finally had both CSV files, I could begin cleaning up the data. To start, I loaded the following packages:

```
library(dplyr)
library(knitr)
library(ggplot2)
library(tidyverse)
library(tidyr)
library(randomForest)
library(gbm)
library(glmnet)
library(lubridate)
library(Metrics)
library(tree)
library(caret)
library(maptree)
library(class)
```

The dataset ranges from Aug 2019 - March 2022, however, it is important to note that there were gaps where I did not wear device for nearly a year because silly me lost and didn't replace the charging cord. Other explanations for missing values in the dataset are also be due to charging my Fitbit overnight, dead battery in the middle of the day, and therefore having no sleep data recorded for the night.

```
sleep_score_data = read_csv('sleep_score.csv')
my_sleep = read_csv('my_sleep.csv')
```

The dates in `sleep_score_data` and `my_sleep` are in difference formats. To fix this, I took a substring of `timestamp` and of `End Time`, only keeping the year, month and year, and added to a new column `date` in both dataframes.

```
# Take substring of timestamp to format dates the same in both dataframes
# Add substrings to new column 'date'
sleep_score_data$date = substr(sleep_score_data$timestamp, 1, 10)
my_sleep$date = substr(my_sleep$'End Time', 1, 10)
```

```
sleep_score_data <- sleep_score_data[, c('date', 'resting_heart_rate', 'restlessness', 'overall_score')]
my_sleep <- my_sleep[, c('date', 'Minutes Asleep', 'Minutes Awake',
                         'Number of Awakenings', 'Time in Bed',
                         'Minutes REM Sleep', 'Minutes Light Sleep',
                         'Minutes Deep Sleep')]
```

The two data frames are merged by `date` and loaded into a new data frame, `sleep_data`.There are 158 observations in `sleep_data`, 27 of which are missing values.

```
# Merge dataframes by date and rename columns
sleep_data <- merge(my_sleep, sleep_score_data, by = 'date')

# Rename columns
sleep_data <- sleep_data %>%
  select(-date) %>%
  drop_na() %>%
  rename(min_asleep = 'Minutes Asleep',
         min_awake = 'Minutes Awake',
         awakenings = 'Number of Awakenings',
         time_bed = 'Time in Bed',
         rem = 'Minutes REM Sleep',
         light_sleep = 'Minutes Light Sleep',
         deep_sleep = 'Minutes Deep Sleep',
         resting_hr = resting_heart_rate,
         restless = restlessness,
         sleep_score = overall_score) %>%
  mutate(restless = restless*100)
head(sleep_data)
```

```
##   min_asleep min_awake awakenings time_bed rem light_sleep deep_sleep
## 1        495        83         23      578 106         328         61
## 2        332        52         18      384  36         240         56
## 3        169        27          3      196  20         138         11
## 4        322        43          8      365  87         171         64
## 5        381        48          8      429  71         275         35
## 6        411        43         10      454  84         269         58
##   resting_hr restless sleep_score
## 1         52 7.968476          81
## 2         54 4.755614          72
## 3         54 1.639344          65
## 4         55 3.419973          76
## 5         56 3.608847          77
## 6         52 2.714932          82
```

Exploratory Data Analysis

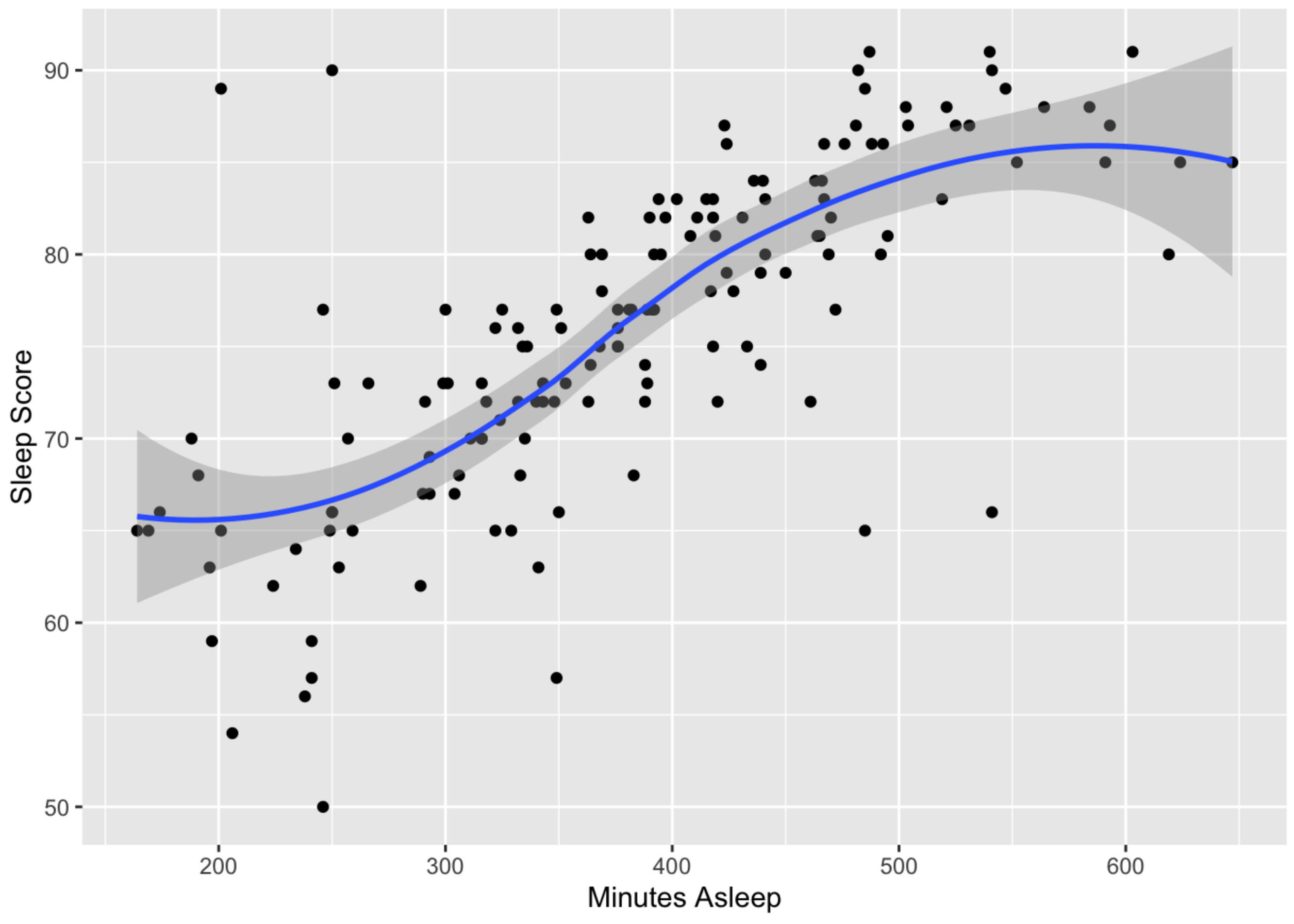
There are 149 total observations in `sleep_data`, where each row represents a different day.

```
summary(sleep_data)
```

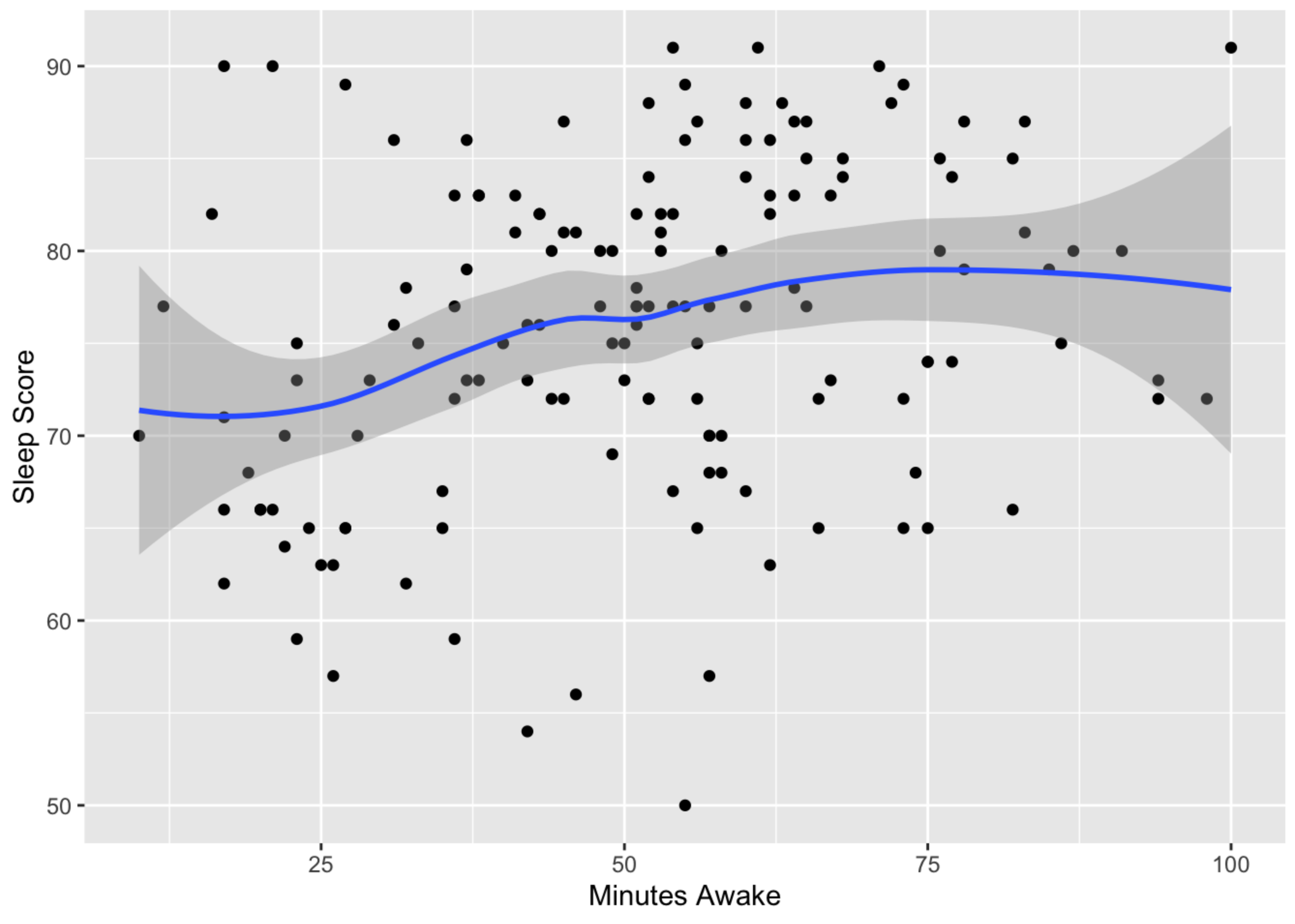
```
##    min_asleep      min_awake        awakenings       time_bed    
##  Min.   :164.0   Min.   : 10.00   Min.   : 3.00   Min.   :188.0  
##  1st Qu.:316.0   1st Qu.: 37.00   1st Qu.:12.00   1st Qu.:357.0  
##  Median :382.0   Median : 52.00   Median :18.00   Median :433.0  
##  Mean   :382.3   Mean   : 51.23   Mean   :17.77   Mean   :433.5  
##  3rd Qu.:463.0   3rd Qu.: 64.00   3rd Qu.:22.00   3rd Qu.:521.0  
##  Max.   :647.0   Max.   :100.00   Max.   :48.00   Max.   :729.0  
##       rem          light_sleep      deep_sleep       resting_hr   
##  Min.   : 12.0   Min.   : 83.0   Min.   :  8.00   Min.   :47.00  
##  1st Qu.: 62.0   1st Qu.:169.0   1st Qu.: 56.00   1st Qu.:51.00  
##  Median : 91.0   Median :213.0   Median : 72.00   Median :53.00  
##  Mean   : 89.7   Mean   :217.9   Mean   : 74.72   Mean   :53.12  
##  3rd Qu.:113.0   3rd Qu.:266.0   3rd Qu.: 91.00   3rd Qu.:56.00  
##  Max.   :209.0   Max.   :387.0   Max.   :191.00   Max.   :61.00  
##     restless        sleep_score   
##  Min.   : 1.299   Min.   :50.00  
##  1st Qu.: 4.589   1st Qu.:70.00  
##  Median : 5.756   Median :77.00  
##  Mean   : 6.101   Mean   :75.94  
##  3rd Qu.: 7.152   3rd Qu.:83.00  
##  Max.   :20.942   Max.   :91.00  
```

To better visualize the data, I graphed each of the predictors against `sleep_score`. From the plots it appears that for the most part all predictors have a positive relationship with `sleep_score`. `restless` is the only variable that has an obvious negative relationship with `sleep_score`. This makes sense because as the percentage of time asleep in restlessness increases, sleep quality should decrease. I would have expected `min_awake` to have a negatively affect sleep score, but as `min_awake` is larger, the total time asleep also increases.

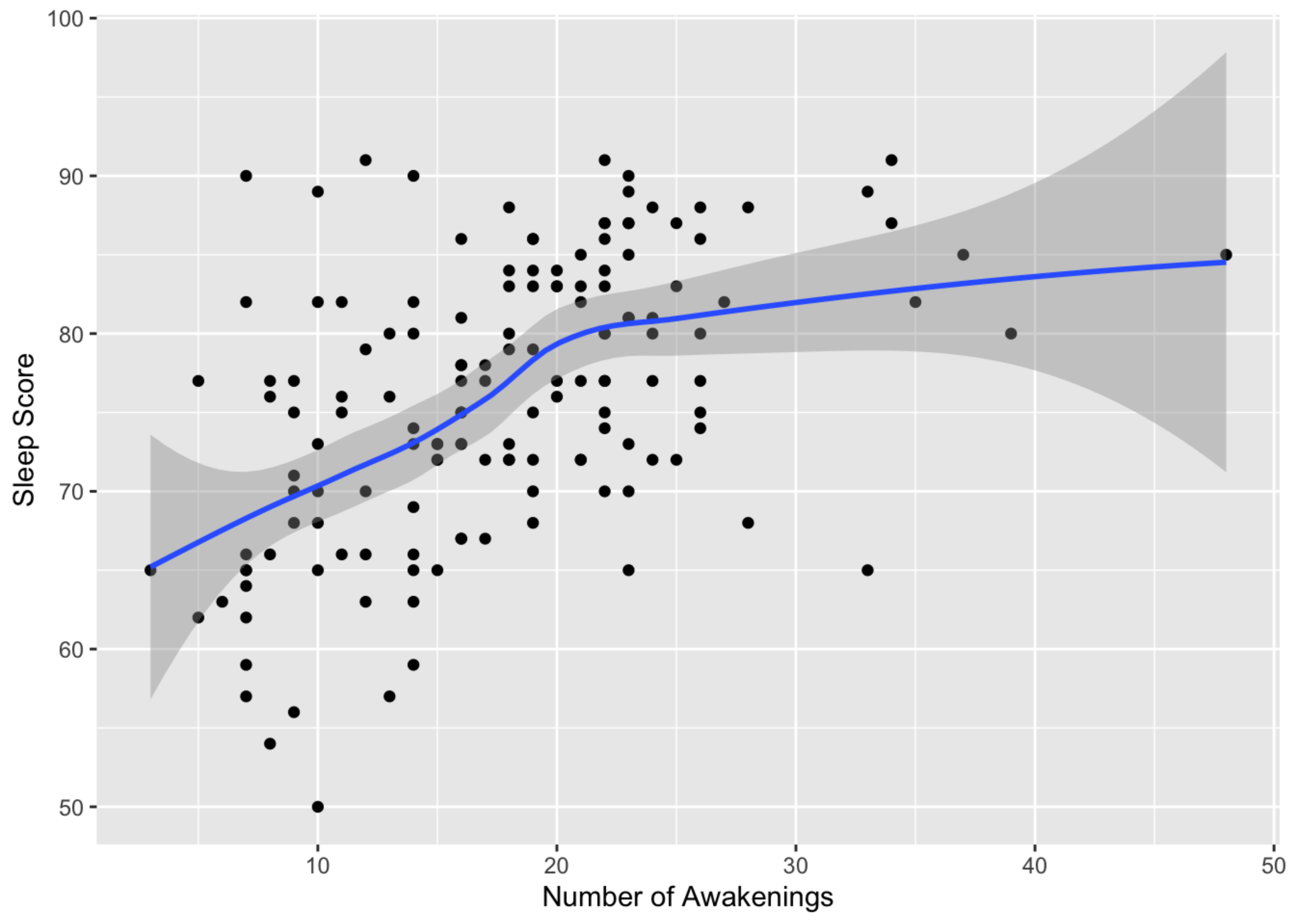
```
sleep_data %>% ggplot(aes(min_asleep, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Minutes Asleep", y="Sleep Score")
```



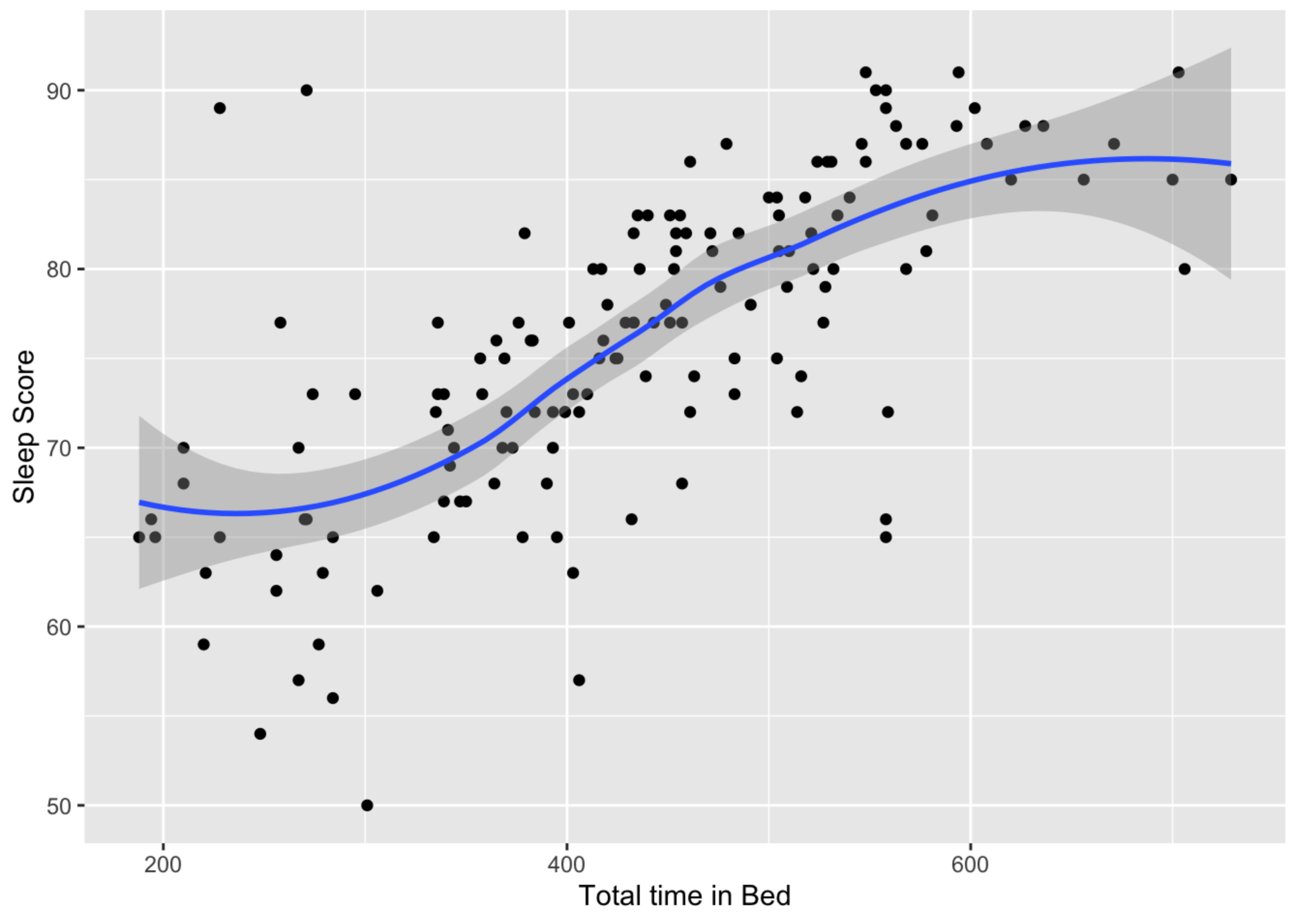
```
sleep_data %>% ggplot(aes(min_awake, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Minutes Awake", y="Sleep Score")
```



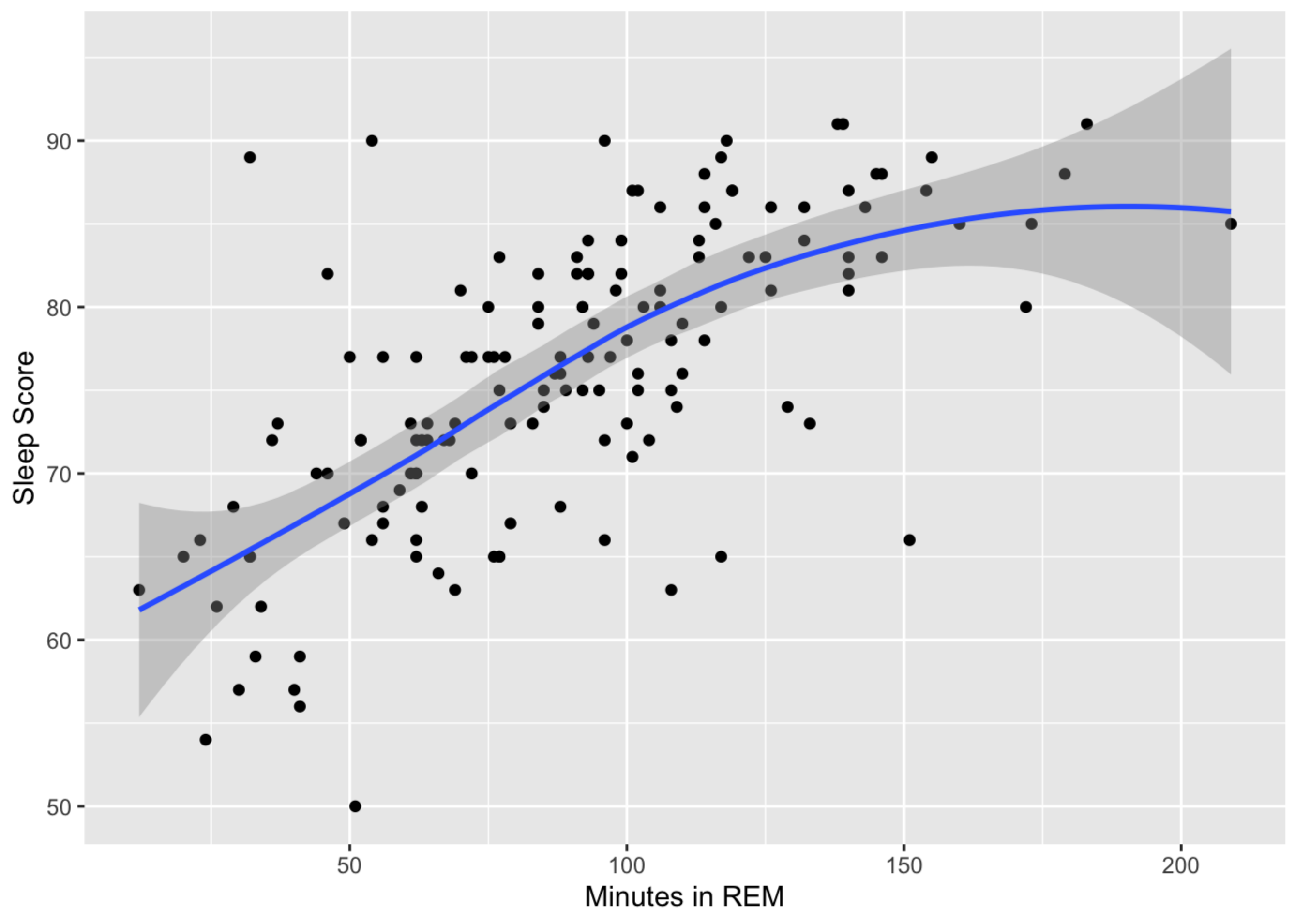
```r
sleep_data %>% ggplot(aes(awakenings, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Number of Awakenings", y="Sleep Score")
```



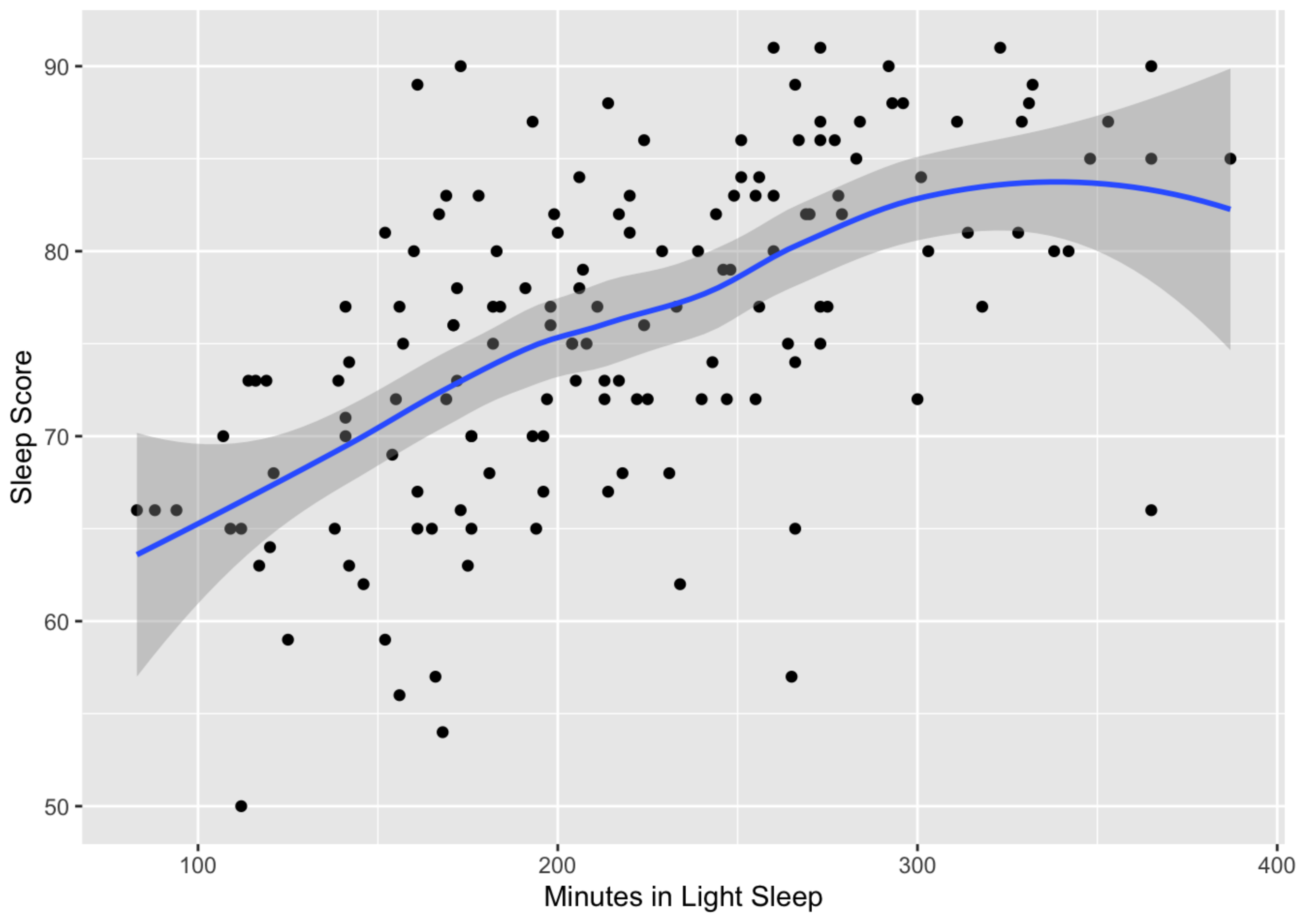
```r
sleep_data %>% ggplot(aes(time_bed, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Total time in Bed", y="Sleep Score")
```



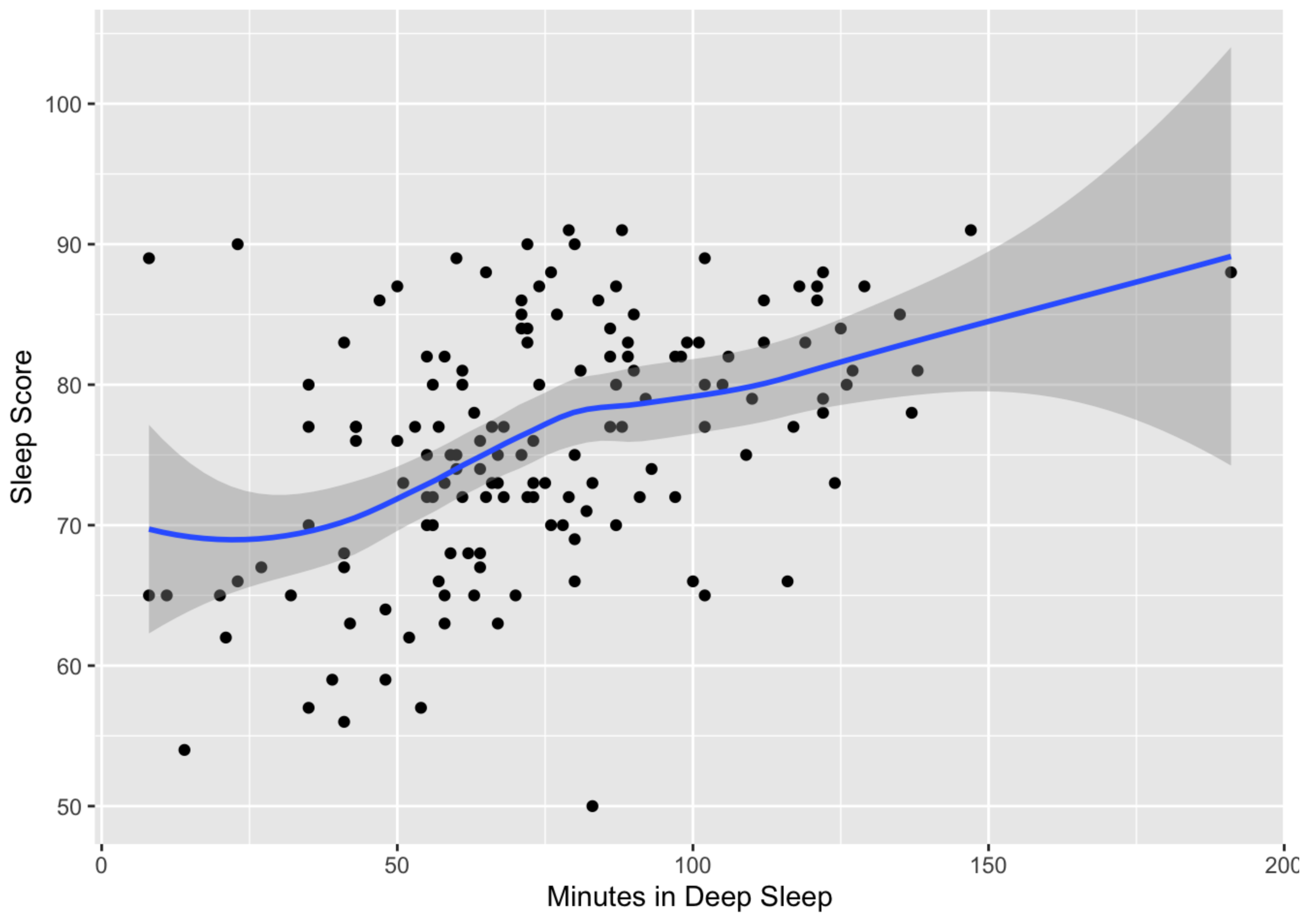
```
sleep_data %>% ggplot(aes(rem, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Minutes in REM", y="Sleep Score")
```



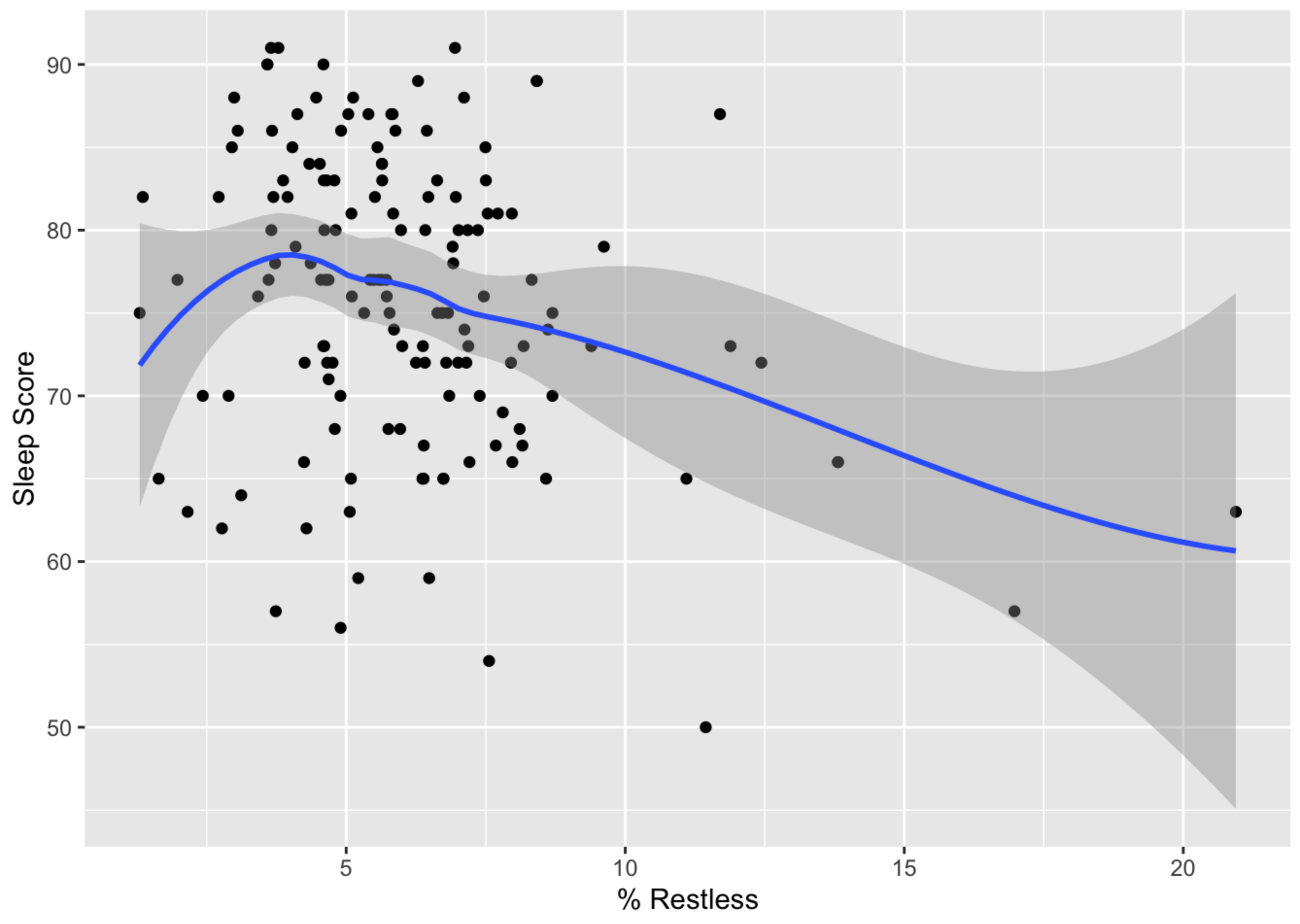
```
sleep_data %>% ggplot(aes(light_sleep, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Minutes in Light Sleep", y="Sleep Score")
```



```
sleep_data %>% ggplot(aes(deep_sleep, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="Minutes in Deep Sleep", y="Sleep Score")
```



```
sleep_data %>% ggplot(aes(restless, sleep_score)) +
  geom_point() +
  geom_smooth() +
  labs(x="% Restless", y="Sleep Score")
```

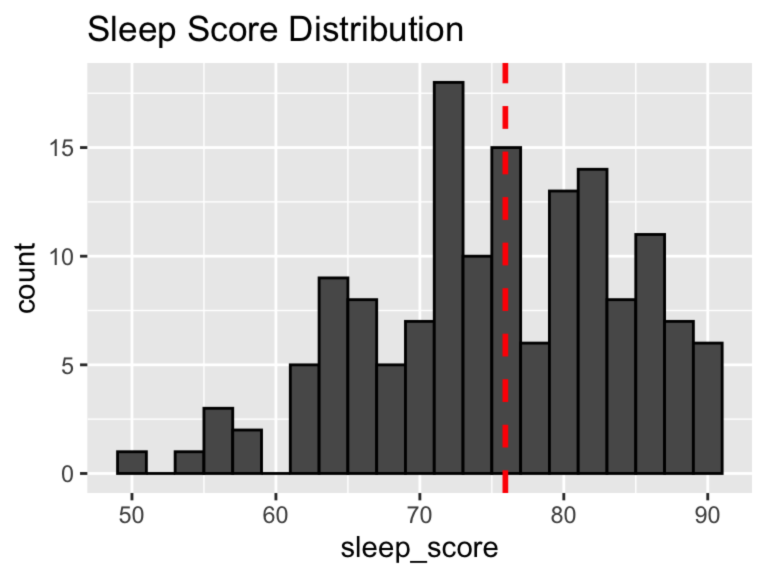


A histogram of sleep score displays the distribution, and we see that the distribution is left-skewed and my average sleep score is 75.94. This makes reasonable sense because having a bad nights rest is more likely to occur than a perfect nights sleep. Things like having to wake up early for class or going to bed late due to school work make a bad sleep likelier to occur.

```
mean(sleep_data$sleep_score)
```

```
## [1] 75.9396
```

```
sleep_data %>% ggplot(aes(sleep_score)) +
  geom_histogram(binwidth = 2, color = 'black') +
  geom_vline(xintercept = mean(sleep_data$sleep_score), col = 'red', lty = 2, lwd = 1) +
  labs(title = 'Sleep Score Distribution')
```



Cross Validation: Data Splitting

The validation set approach for cross-validation will be employed in order to estimate the test error rates that result from fitting various linear models on sleep_data. Cross-validation contains an element of randomness so I used set.seed() to ensure that my results are reproducible down the line.

sleep_data will be split into 2 sets: the training set and the test set. A random sample of 70% of observations (104 obs) will serve as the training set, and the remaining 30% (45 obs) of observations will serve as the validation set. The model is fit on the training set, and the fitted model is used to predict the sleep scores for the observations in the validation set.

```
# set.seed() for reproducible results
set.seed(123)
train <- sample(1:nrow(sleep_data), 0.70*nrow(sleep_data))

# Sample 75% of observations as training data
sleep_train <- sleep_data[train,]
# remaining 25% as test set
sleep_test <- sleep_data[-train,]

# sleep scores for training and test set
y.train <- sleep_train$sleep_score
y.test <- sleep_test$sleep_score
```

Model Building

First I fit a multiple linear regression model to the training set. `min_awake` and `restless` have a p-value < 0.05, meaning they are statistically significant. `time_bed` and `deep_sleep` were not "not defined because of singularities". This is indicative that there is multicollinearity in the predictors. 59.83% of the variability can be explained by this model.

```
sleep.lm <- lm(sleep_score ~ ., data = sleep_train)
summary(sleep.lm)
```

```
##
## Call:
## lm(formula = sleep_score ~ ., data = sleep_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.1993  -2.1699   0.6206   2.5796  23.0389
##
## Coefficients: (2 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 57.19973   10.66632   5.363 5.65e-07 ***
## min_asleep   0.02246    0.02305   0.974 0.332289
## min_awake   -0.07919    0.03567  -2.220 0.028780 *
## awakenings   0.11736    0.11722   1.001 0.319250
## time_bed          NA         NA      NA       NA
## rem          0.07042    0.03813   1.847 0.067835 .
## light_sleep  0.02260    0.02240   1.009 0.315425
## deep_sleep        NA         NA      NA       NA
## resting_hr   0.11741    0.18590   0.632 0.529161
## restless    -0.86226    0.23271  -3.705 0.000353 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.668 on 96 degrees of freedom
## Multiple R-squared:  0.5972, Adjusted R-squared:  0.5678
## F-statistic: 20.33 on 7 and 96 DF,  p-value: < 2.2e-16
```

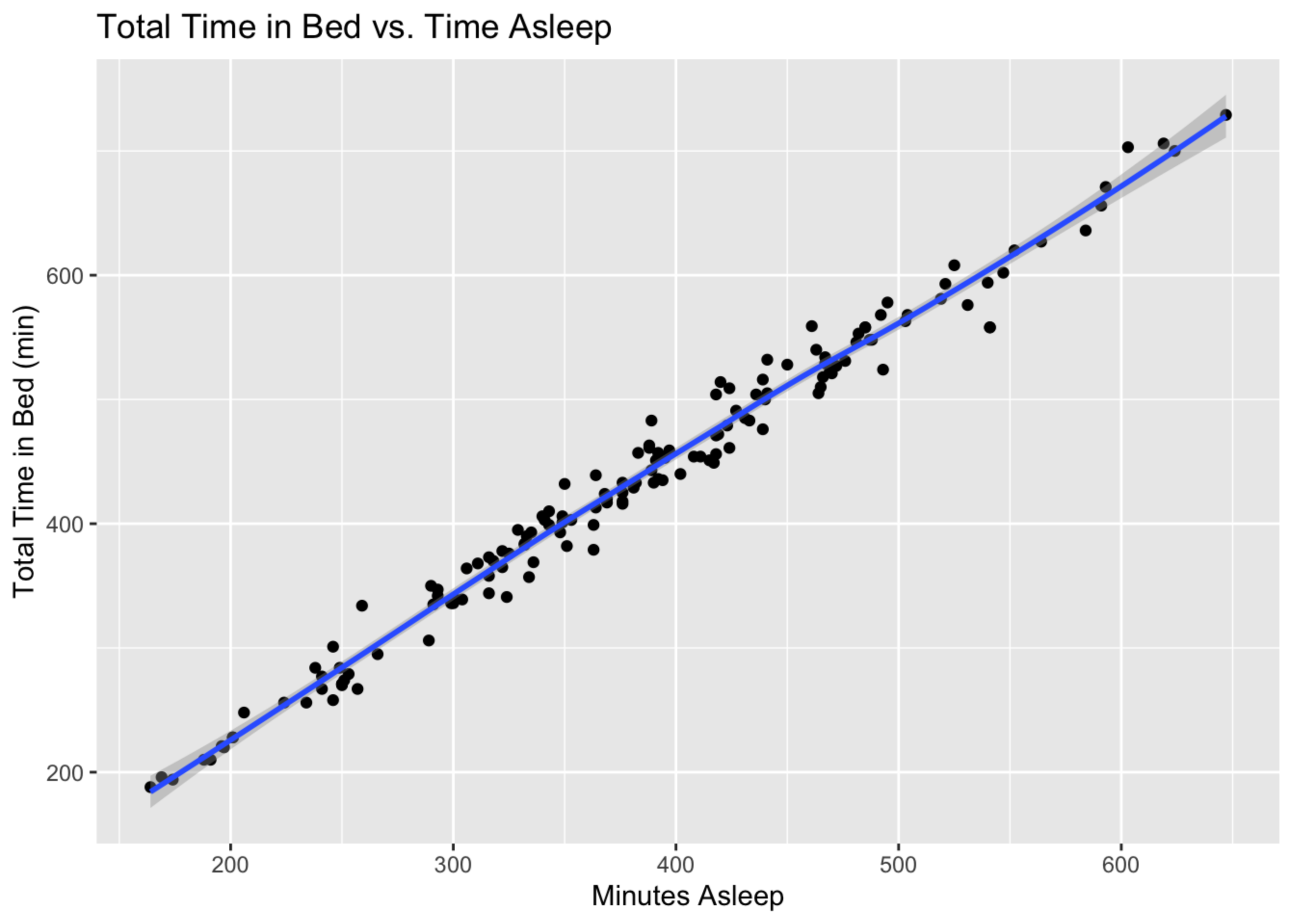
From the correlation matrix, we see that many of the predictors are correlated since more time asleep also means more time spent in the other stages of sleep. `min_asleep` and `time_bed` are perfectly correlated, and this makes sense because the total time spent in bed is typically the same amount of time spent asleep.

```
# Correlation matrix
cor(sleep_data)
```

```
##              min_asleep   min_awake  awakenings    time_bed         rem
## min_asleep   1.00000000  0.58485289  0.72818650  0.99071291  0.84021946
## min_awake    0.58485289  1.00000000  0.61603376  0.68971221  0.50868425
## awakenings   0.72818650  0.61603376  1.00000000  0.75329868  0.55478614
## time_bed     0.99071291  0.68971221  0.75329868  1.00000000  0.83531280
## rem          0.84021946  0.50868425  0.55478614  0.83531280  1.00000000
## light_sleep  0.84373985  0.49443852  0.65324800  0.83606737  0.49138275
## deep_sleep   0.59192342  0.32294704  0.41603409  0.58253030  0.61241558
## resting_hr  -0.09960565 -0.07964093 -0.07815007 -0.10226556 -0.09508876
## restless    -0.01958047  0.26642720  0.16623581  0.02718189 -0.02493736
## sleep_score  0.74061039  0.28682045  0.49760334  0.70920354  0.67578555
##              light_sleep  deep_sleep  resting_hr     restless sleep_score
## min_asleep   0.843739850  0.59192342 -0.09960565 -0.019580466  0.74061039
## min_awake    0.494438518  0.32294704 -0.07964093  0.266427195  0.28682045
## awakenings   0.653247997  0.41603409 -0.07815007  0.166235814  0.49760334
## time_bed     0.836067369  0.58253030 -0.10226556  0.027181890  0.70920354
## rem          0.491382753  0.61241558 -0.09508876 -0.024937361  0.67578555
## light_sleep  1.000000000  0.13703157 -0.04898378 -0.004480269  0.59066888
## deep_sleep   0.137031572  1.00000000 -0.12147647 -0.027476464  0.44760133
## resting_hr  -0.048983783 -0.12147647  1.00000000 -0.019602398  0.01453596
## restless    -0.004480269 -0.02747646 -0.01960240  1.000000000 -0.28032650
## sleep_score  0.590668883  0.44760133  0.01453596 -0.280326497  1.00000000
```

This multicollinearity can be visualized in the plot below.

```
sleep_data %>% ggplot(aes(min_asleep, time_bed)) +
  geom_point() +
  geom_smooth() +
  labs(x = "Minutes Asleep", y = "Total Time in Bed (min)", title = "Total Time in Bed vs. Time Asleep")
```



To resolve this issue of multicollinearity, we simply remove `time_bed` and `deep_sleep` from the model.

```
sleep.lm <- lm(sleep_score ~. - time_bed -deep_sleep, data = sleep_train)
summary(sleep.lm)
```

```
##
## Call:
## lm(formula = sleep_score ~ . - time_bed - deep_sleep, data = sleep_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.1993  -2.1699   0.6206   2.5796  23.0389
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 57.19973   10.66632   5.363 5.65e-07 ***
## min_asleep   0.02246    0.02305   0.974 0.332289
## min_awake   -0.07919    0.03567  -2.220 0.028780 *
## awakenings   0.11736    0.11722   1.001 0.319250
## rem          0.07042    0.03813   1.847 0.067835 .
## light_sleep  0.02260    0.02240   1.009 0.315425
## resting_hr   0.11741    0.18590   0.632 0.529161
## restless    -0.86226    0.23271  -3.705 0.000353 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.668 on 96 degrees of freedom
## Multiple R-squared:  0.5972, Adjusted R-squared:  0.5678
## F-statistic: 20.33 on 7 and 96 DF,  p-value: < 2.2e-16
```

The training MSE is 29.652 on the training set (75% of the observations) and the test MSE is 24.911.

```
pred.train.lm <- predict(sleep.lm, newdata = sleep_train, type = 'response')
pred.test.lm <- predict(sleep.lm, newdata = sleep_test, type = 'response')

# Training MSE
mean((pred.train.lm - y.train)^2)
```

```
## [1] 29.65247
```

```
# Test MSE
lm.mse = mean((pred.test.lm - y.test)^2)
lm.mse
```

```
## [1] 24.91087
```

Shrinkage Using the Lasso and Ridge Regression

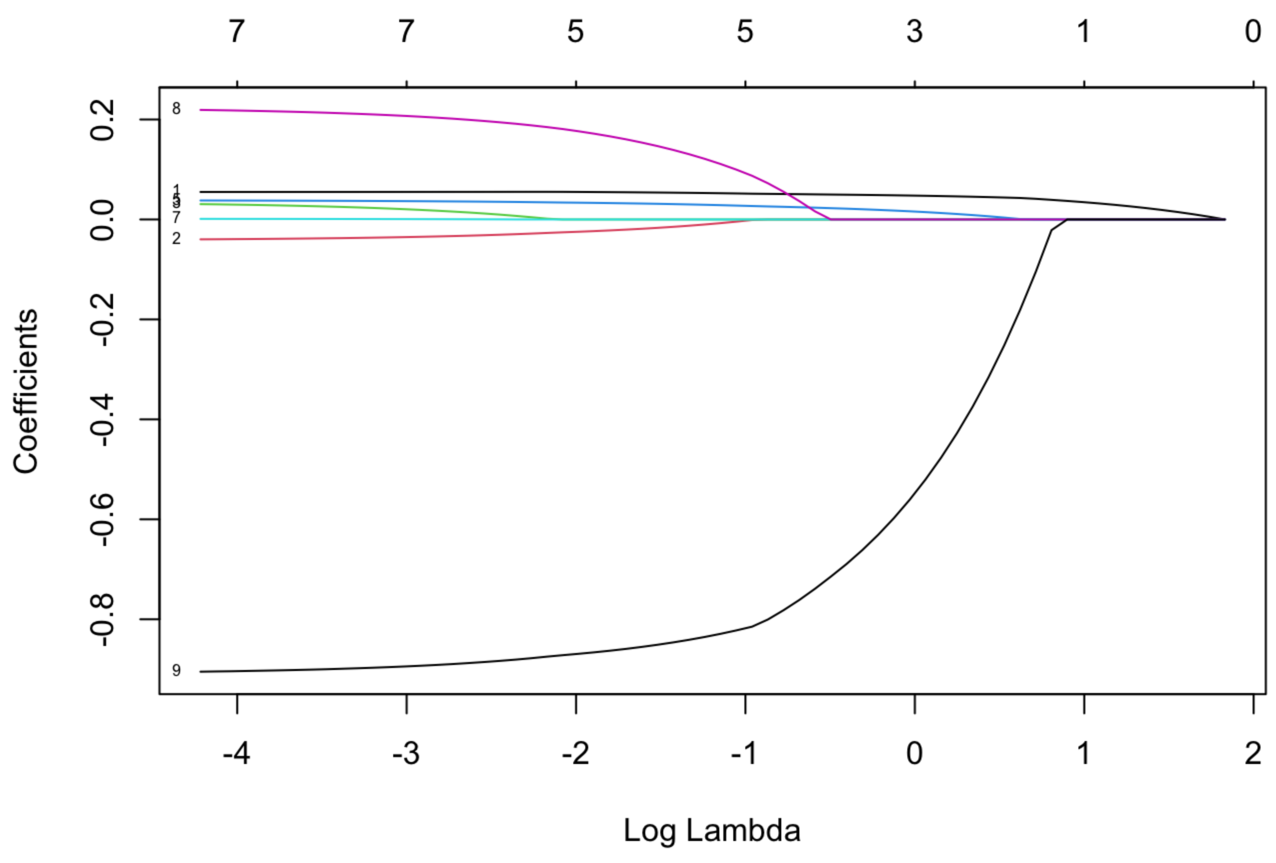
Setting up the training and test data sets for the shrinkage methods.

```
set.seed(123)
# Building X matrix from data
xmod = model.matrix(sleep_score~., sleep_data)[,-1]
y = sleep_data$sleep_score

set.seed(1)
train=sample(1:nrow(xmod), nrow(xmod)*0.7)

xtrain = xmod[train, ]
ytrain = y[train]
# The rest as test data
xtest = xmod[-train, ]
ytest = y[-train]
```

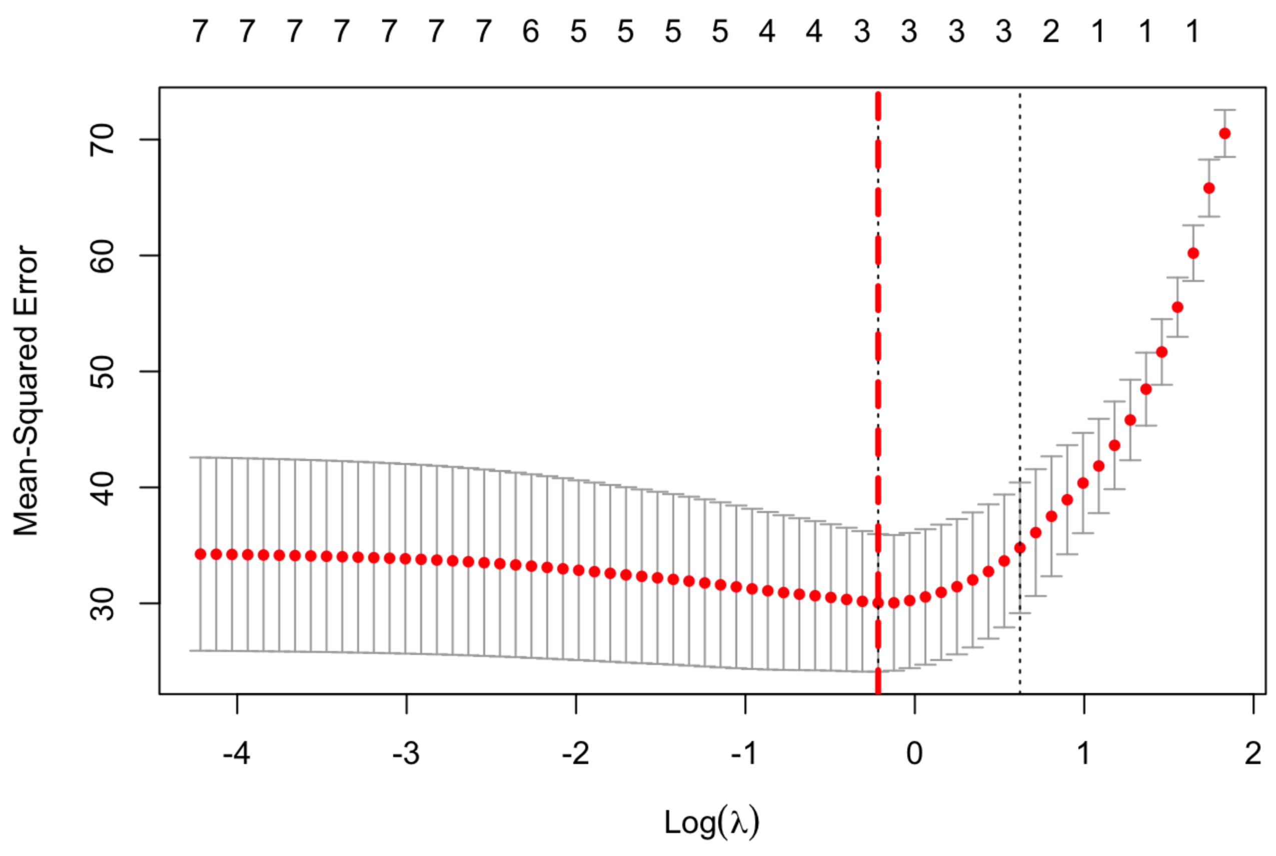
The Lasso

Using 5-fold cross-validation, the optimal tuning parameter for the lasso model is estimated to be $\lambda = 0.805$. The corresponding training error for the lasso regression is 26.899, and the test error is 36.11 In comparison to the linear regression model, the training MSE is smaller, however, the test MSE using the lasso model is larger than the test error using the regression model.

```
set.seed(123)
lasso.mod <- glmnet(xtrain, ytrain, alpha = 1)
plot(lasso.mod, xvar="lambda", label = TRUE)
```



```
cv.lasso <- cv.glmnet(xtrain, ytrain, alpha = 1, nfolds = 5)
plot(cv.lasso)
abline(v = log(cv.lasso$lambda.min), col="red", lwd=3, lty=2)
```



```
bestlam = cv.lasso$lambda.min
bestlam
```

```
## [1] 0.805222
```

```
lasso.pred.train <- predict(lasso.mod, s = bestlam, newx = xtrain)
lasso.pred.test <- predict(lasso.mod, s = bestlam, newx = xtest)

# MSEs
mean((lasso.pred.train-ytrain)^2)
```

```
## [1] 26.89904
```

```
lasso.mse = mean((lasso.pred.test-ytest)^2)
lasso.mse
```

```
## [1] 36.11032
```

Below, we see that the coefficient estimates of `awakenings` , `time_bed` and `light_sleep` , and `deep_sleep` are exactly zero. Consequently, the lasso model with λ chosen by cross-validation only contains 4 out of the 8 predictor variables.

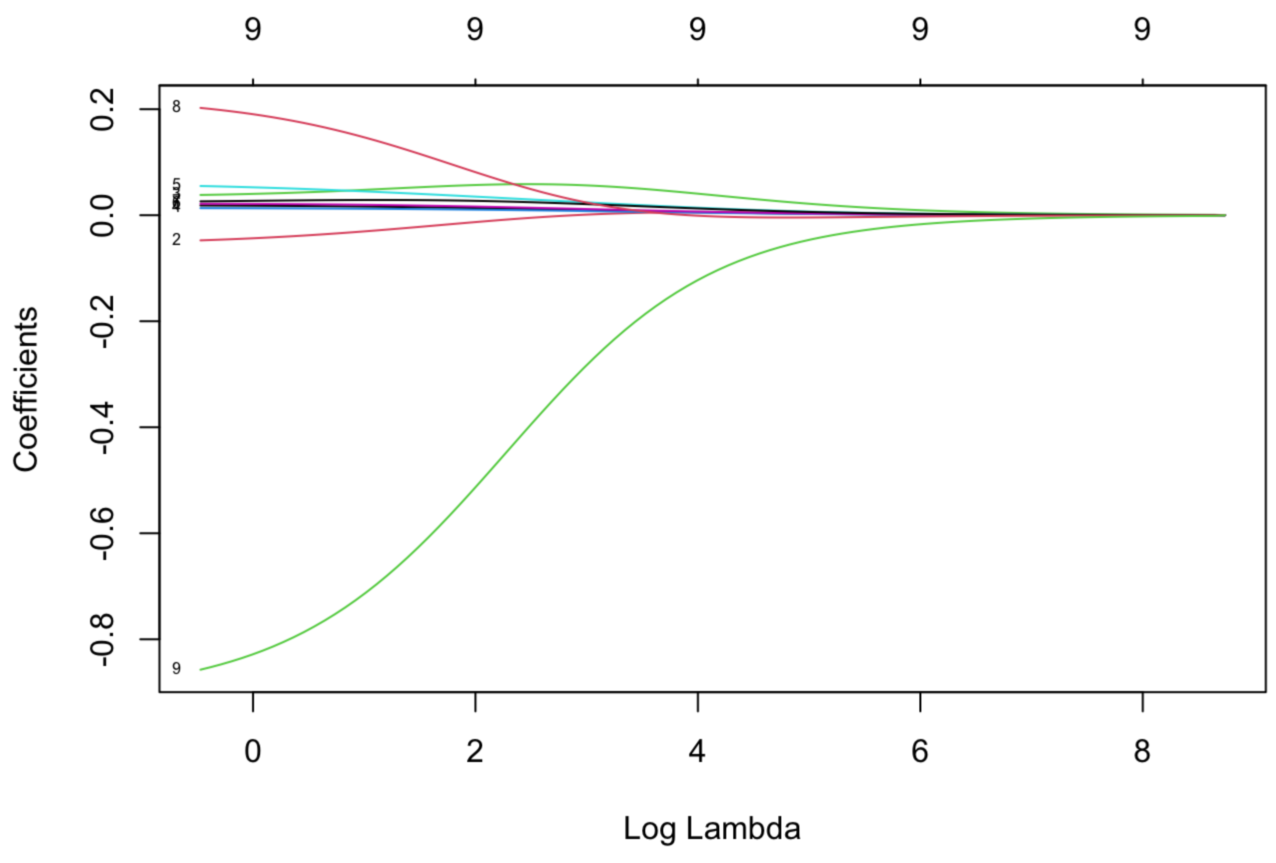
```
out = glmnet(xmod, y, alpha=1)
lasso.coef = predict(out, type="coefficients", s=bestlam)[1:10,]
lasso.coef
```

```
## (Intercept)  min_asleep   min_awake  awakenings    time_bed         rem
## 59.71298890  0.04458456  0.00000000  0.00000000  0.00000000  0.02963632
## light_sleep  deep_sleep  resting_hr    restless
##  0.00000000  0.00000000  0.00000000 -0.56973592
```

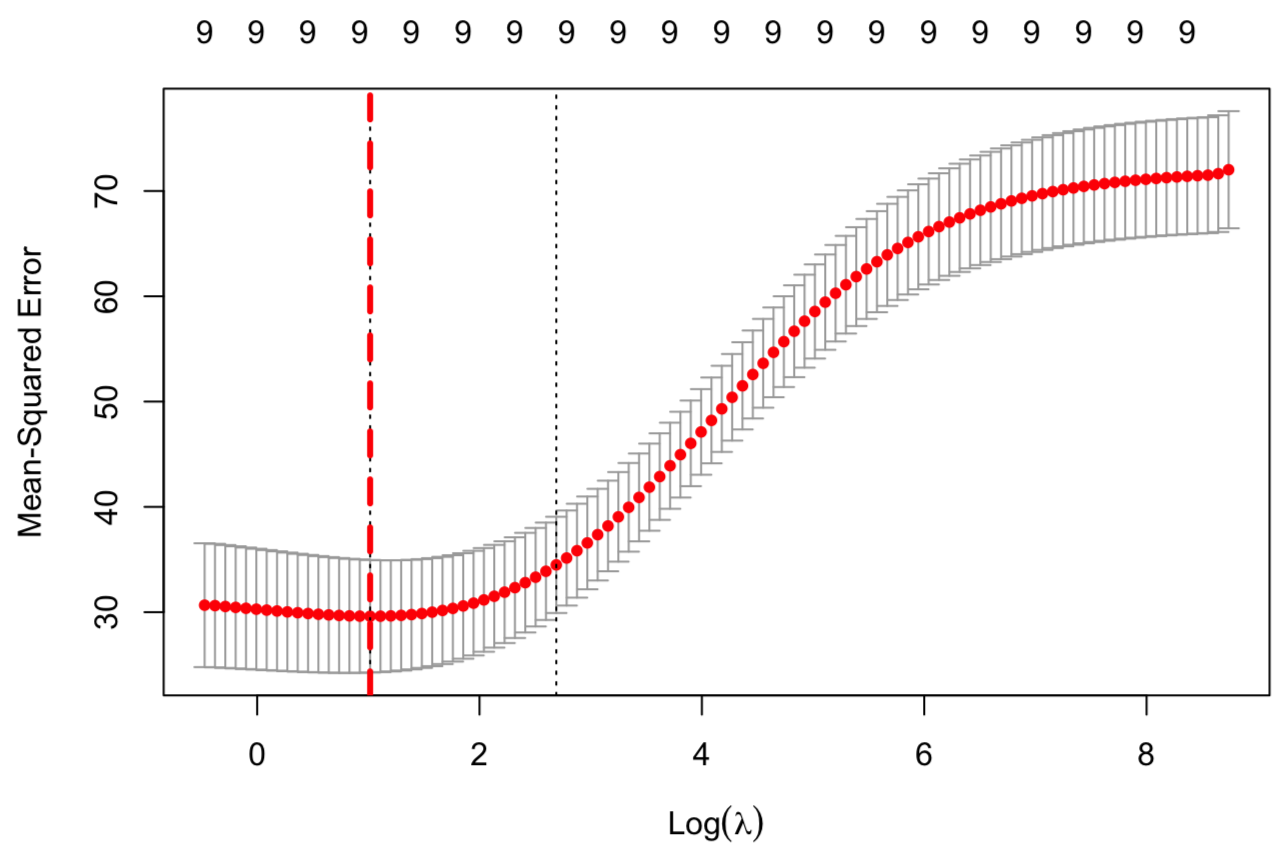
Ridge Regression

Now performing a 5-fold cross-validation to choose the optimal tuning parameter to fit the ridge regression model, we find that $\lambda = 2.762$. The associated training and test errors for the ridge regression model are 25.487 and 34.34, respectively. Comparing these MSEs to the lasso, the differences are so slight that the MSEs are nearly the same.

```
set.seed(123)
ridge.mod <- glmnet(xtrain, ytrain, alpha = 0)
plot(ridge.mod, xvar="lambda", label = TRUE)
```



```
cv.ridge <- cv.glmnet(xtrain, ytrain, alpha = 0, folds = 5)
plot(cv.ridge)
abline(v = log(cv.ridge$lambda.min), col="red", lwd=3, lty=2)
```



```
bestlam2 = cv.ridge$lambda.min
bestlam2
```

```
## [1] 2.762289
```

```
# Make predictions
ridge.pred.train <- predict(ridge.mod, s = bestlam2, newx = xtrain)
ridge.pred.test <- predict(ridge.mod, s = bestlam2, newx = xtest)
# MSEs
mean((ridge.pred.train-ytrain)^2)
```

```
## [1] 25.48725
```

```
ridge.mse = mean((ridge.pred.test-ytest)^2)
ridge.mse
```

```
## [1] 34.33954
```

However the advantage with the lasso over the ridge regression model is that the resulting coefficient estimates from the lasso model are sparse, if not zero. The lasso effectively shrinks the coefficient estimates toward zero, meanwhile, none of the coefficient estimates for the ridge regression model are exactly zero.

```
# Coefficient estimates for ridge regression model
out2 = glmnet(xmod, y, alpha=0)
ridge.coef = predict(out2, type="coefficients", s=bestlam2)[1:10,]
ridge.coef
```

```
## (Intercept)  min_asleep   min_awake  awakenings    time_bed          rem
## 49.98155924  0.01549536 -0.04012160  0.08583411  0.01127958  0.05007009
## light_sleep  deep_sleep  resting_hr    restless
##  0.01883569  0.02129431  0.17721078 -0.64326239
```

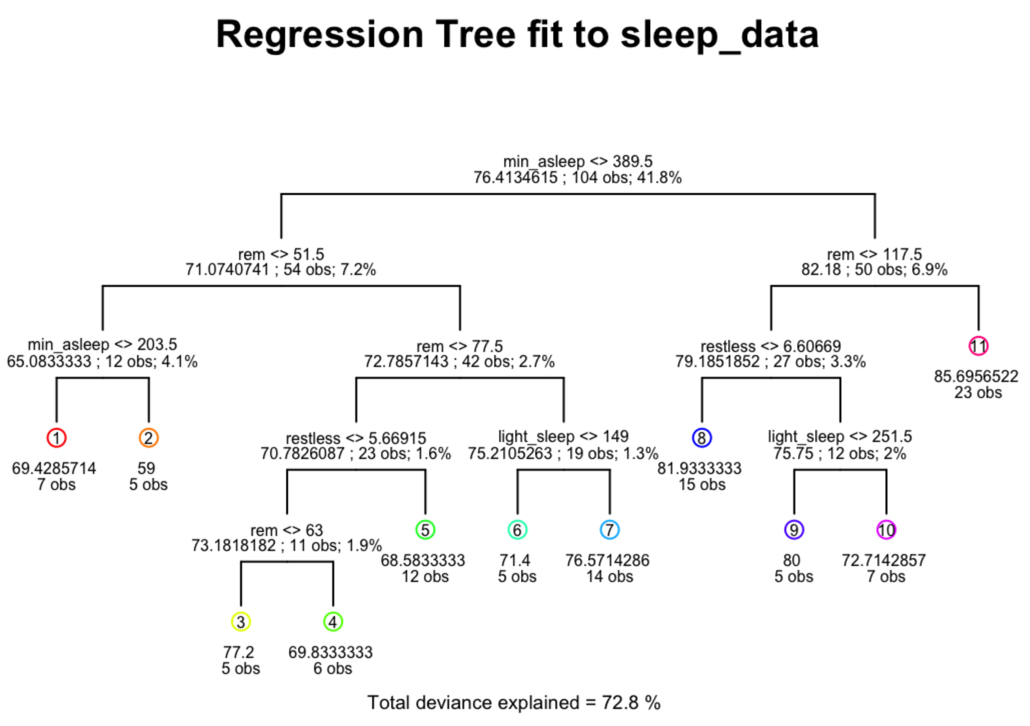
Regression Trees

The regression tree fit to my sleep dataset uses 5 out of 8 predictors: `min_asleep` , `min_awake` , `rem` , `deep_sleep` , and `awakenings` . From the figure we see that `min_sleep` is the most indicative predictor of `sleep_score` and is partitioned at 389.5 minutes or about 6.5 hours of sleep. This is intuitive since longer time asleep would seem to improve improve sleep quality, and hence Fitbit's sleep score. After predicting sleep scores on the test set, the test MSE was calculated to be 39.416. This regression tree model performed poorly in comparison to the former models, as indicated by the large test MSE.

```
set.seed(123)
tree.sleep <- tree(sleep_score ~ ., sleep_train)
summary(tree.sleep)
```

```
##
## Regression tree:
## tree(formula = sleep_score ~ ., data = sleep_train)
## Variables actually used in tree construction:
## [1] "min_asleep"   "rem"          "restless"     "light_sleep"
## Number of terminal nodes:  11
## Residual mean deviance:  22.3 = 2074 / 93
## Distribution of residuals:
##     Min.  1st Qu.  Median     Mean  3rd Qu.     Max.
## -9.0000 -2.6960 -0.5714  0.0000  2.0670 19.5700
```

```
# Visualize regression tree
draw.tree(tree.sleep, nodeinfo=TRUE, cex = .5)
title("Regression Tree fit to sleep_data")
```



After predicting sleep scores on the training and test set, the training MSE is 19.944 and the test MSE is 44.541. The regression tree model received the lowest training error out of all the former methods, but also received the highest test MSE of all the models thus far. This is an indication that the regression tree model overfitted the data due to high tree complexity.

```
# Predict on train/test set
tree.pred.train = predict(tree.sleep, sleep_train)
tree.pred.test = predict(tree.sleep, sleep_test)
mean((tree.pred.train - y.train)^2)
```

```
## [1] 19.9435
```

```
regtree.mse = mean((tree.pred.test - y.test)^2)
regtree.mse
```

```
## [1] 44.54071
```

Pruning the tree with K-fold CV

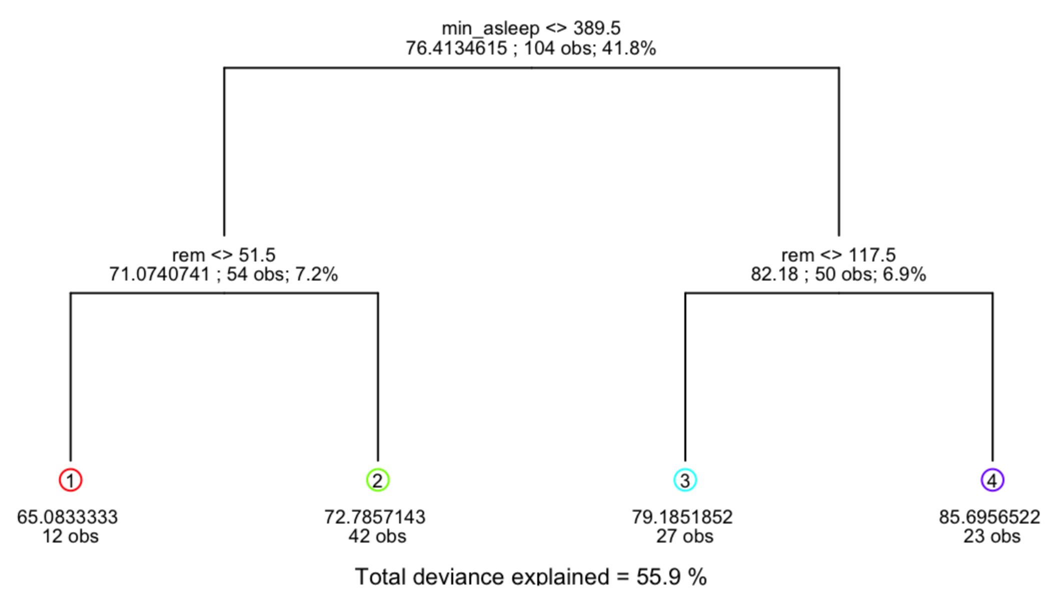
The regression tree model performed poorly on the test set, likely because the resulting tree model was too complex. Pruning the tree may be able to reduce variance with little bias. A 10-fold cross-validation was performed to determine the optimal level of tree complexity. Cross-validation estimates that a tree with 4 terminal nodes is the best size of a tree which minimizes the cross-validation estimate of the test error rate. After pruning the tree, the test MSE was calculated to be 31.093 which is a big improvement from the unpruned regression tree model.

```
set.seed(123)
# K-fold cross-validation
cv.sleep <- cv.tree(tree.sleep, K=10)

# CV determines best size
bestcv = min(cv.sleep$size[cv.sleep$dev == min(cv.sleep$dev)])
bestcv
```

```
## [1] 4
```

```
# prune the tree
prune.sleep <- prune.tree(tree.sleep, best = bestcv)
draw.tree(prune.sleep, nodeinfo = TRUE, cex = 0.6)
title("Pruned Tree of Size 4")
```

Pruned Tree of Size 4



After pruning the tree and predicting sleep score on the training and validation sets, train MSE = 19.944 and test MSE = 27.588 which is a big improvement from the unpruned regression tree model.

```
set.seed(123)
# Predict on train/test set
pred.prune.train = predict(tree.sleep, sleep_train)
pred.prune.test = predict(prune.sleep, sleep_test)
# MSEs
mean((pred.prune.train - y.train)^2)
```

```
## [1] 19.9435
```

```
prune.mse = mean((pred.prune.test - y.test)^2)
prune.mse
```

```
## [1] 27.58846
```

Random Forest

Next, I fit a random forest model on the training and test sets and yielded a train MSE of 9.736 and a test MSE of 23.316. The random forest model has yielded the lowest training error out of all the models, however its test MSE is over double its training error meaning the model was overfitted.

```
set.seed(123)
sleep.rf <- randomForest(sleep_score ~ ., data = sleep_train, importance=TRUE)
sleep.rf
```

```
##
## Call:
##  randomForest(formula = sleep_score ~ ., data = sleep_train, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 47.07563
##                     % Var explained: 36.05
```

```
# Predictions on train/test set
pred.rf.train <- predict(sleep.rf, newdata = sleep_train)
pred.rf.test <- predict(sleep.rf, newdata = sleep_test)

# MSE
mean((pred.rf.train - y.train)^2)
```

```
## [1] 9.735865
```

```
rf.mse = mean((pred.rf.test - y.test)^2)
rf.mse
```

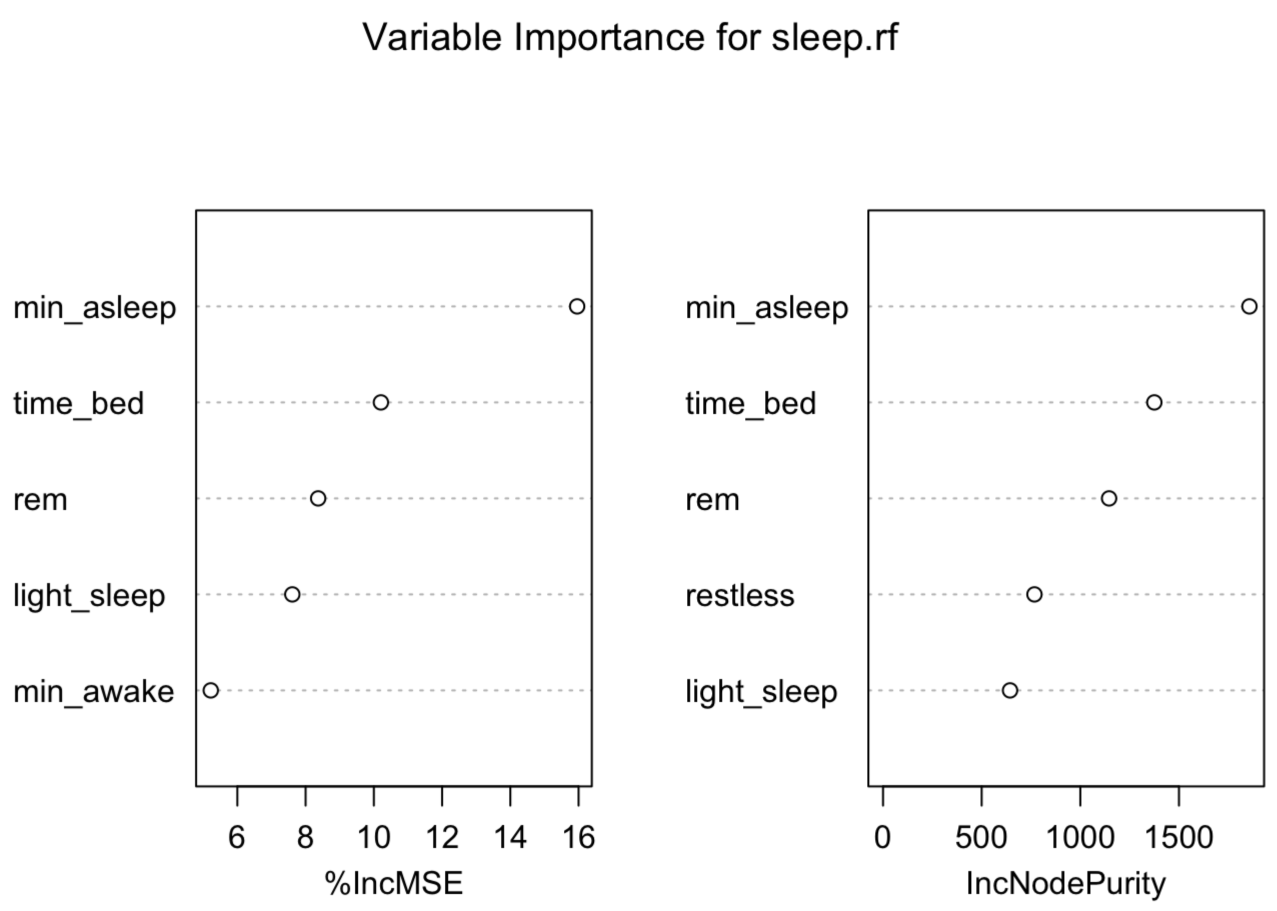
```
## [1] 23.31609
```

The importancce() function indicates which variables are most important in the random forest model, and the plot below allows us to visualize the importance of the variables. The results indicate that across all trees in the random forests, the total time asleep (`min_asleep`) and minutes spent in REM sleep stage (`rem`) are the two most important variables.

```
importance(sleep.rf)
```

```
##               %IncMSE IncNodePurity
## min_asleep  15.957894     1857.1104
## min_awake    5.220972      383.6822
## awakenings   1.372713      455.0532
## time_bed    10.206894     1375.1076
## rem          8.367828     1145.7462
## light_sleep  7.610470      643.7939
## deep_sleep  -7.220538      406.3672
## resting_hr   1.296350      148.5359
## restless     2.286603      767.5247
```

```
varImpPlot(sleep.rf, sort=T, main="Variable Importance for sleep.rf", n.var=5)
```



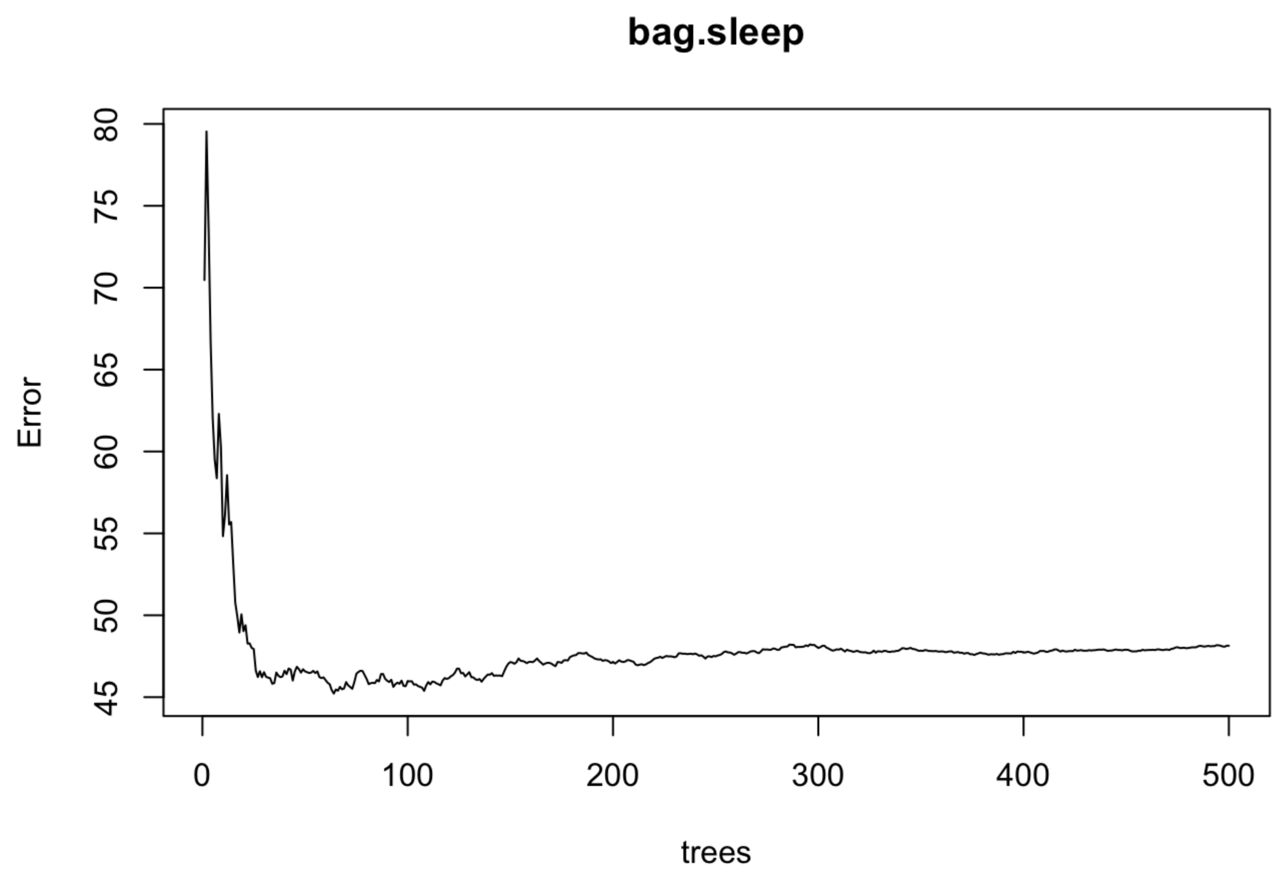
Bagging

Using all 9 predictors for each split in the tree, the test MSE associated with the bagged regression tree is 22.857, which is very similar but ever so slightly smaller than that of the random forests test MSE. However, bagging performed better than a optimally-pruned single tree.

```
set.seed(123)
bag.sleep <- randomForest(sleep_score ~ ., data = sleep_train, mtry = 9, importance=TRUE)
bag.sleep
```

```
##
## Call:
##  randomForest(formula = sleep_score ~ ., data = sleep_train, mtry = 9,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 9
##
##           Mean of squared residuals: 48.13822
##                     % Var explained: 34.6
```

```
plot(bag.sleep)
```



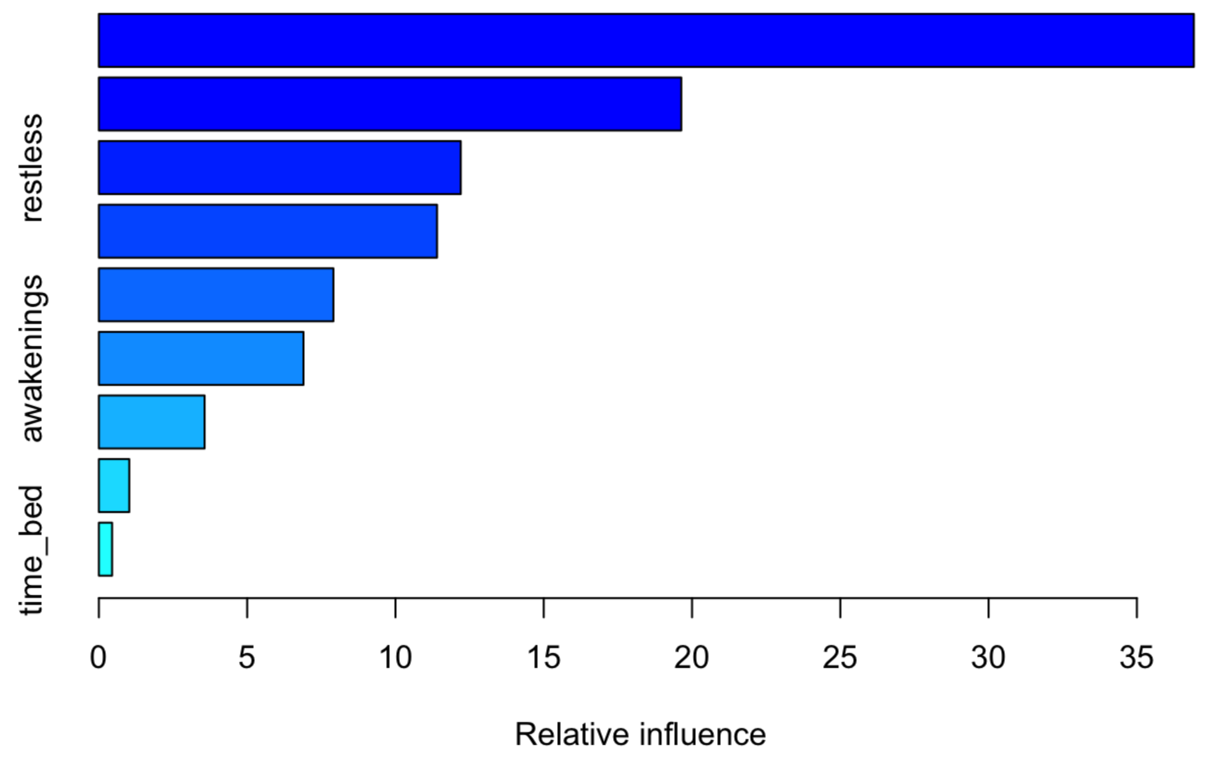
```
pred.bag = predict(bag.sleep, newdata=sleep_test)
bag.mse = mean((pred.bag - y.test)^2)
bag.mse
```

```
## [1] 22.85702
```

Boosting

Boosted regression trees were fit to sleep_data using the gbm() package. `min_asleep` and `rem` have a relative influence of 36.920 and 19.637, respectively, and have the most influence on `sleep_score` .

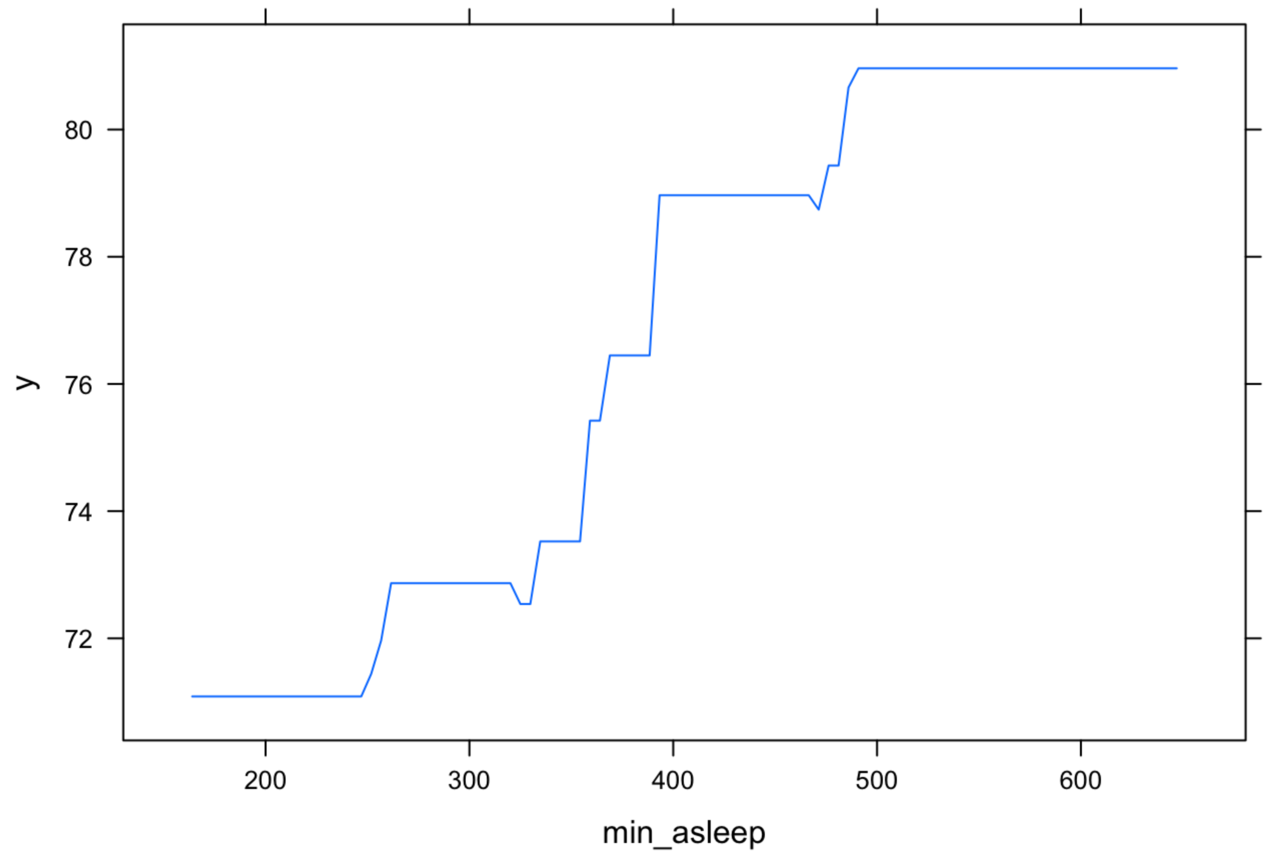
```
set.seed(123)
boost.sleep <- gbm(sleep_score ~ ., data = sleep_train, distribution = "gaussian")
summary(boost.sleep)
```



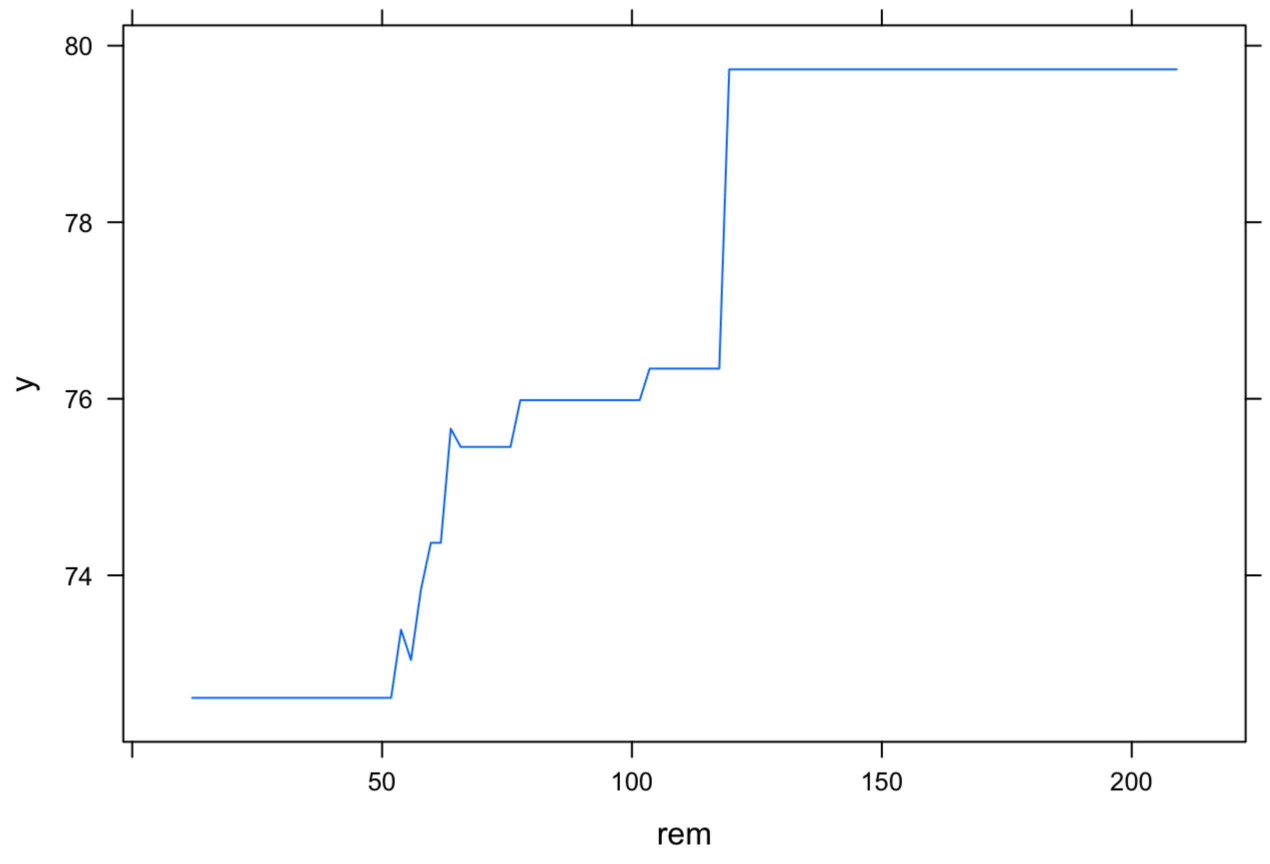
```
##                     var    rel.inf
## min_asleep   min_asleep 36.9204393
## rem                 rem 19.6369153
## restless       restless 12.2016348
## light_sleep light_sleep 11.4014849
## min_awake     min_awake  7.9072751
## awakenings   awakenings  6.8980857
## deep_sleep   deep_sleep  3.5646670
## resting_hr   resting_hr  1.0242637
## time_bed       time_bed  0.4452344
```

Producing partial dependence plots illustrate the effect `min_asleep` and `rem` have on `sleep_score` after integrating out all other predictors.

```
par(mfrow =c(1,2))
plot(boost.sleep, i="min_asleep")
```



```
plot(boost.sleep, i="rem")
```



The boosted model can now be used to predict `sleep_score` on the test set. The test MSE for the boosted regression tree is calculated to be 28.08, and the training MSE is 23.0678 The boosted model did not perform better than bagging, but did perform better than the optimally-pruned single tree.

```
pred.boost.train <- predict(boost.sleep, newdata = sleep_train)
pred.boost.test <- predict(boost.sleep, newdata = sleep_test)

mean((pred.boost.train - y.train)^2)
```

```
## [1] 23.06777
```

```
boost.mse = mean((pred.boost.test - y.test)^2)
boost.mse
```

```
## [1] 28.08043
```

Model Selection and Performance

I created a data frame of all the models and their associated test MSEs to take a better look side by side. The bagged tree model received the lowest test MSE of 22.522, but is incredibly similar to the random forest model's test MSE.

```r
all.mse = c(lm.mse, lasso.mse, ridge.mse, regtree.mse, prune.mse, rf.mse, bag.mse, boost.mse)

Model=c("Multiple Linear Regression",
                    "Lasso",
                    "Ridge Regression",
                    "Regression Tree",
                    "Optimal Pruned Tree",
                    "Random Forest",
                    "Bagged Tree",
                    "Boosted Tree")

df = data.frame(Model, Test_MSE=all.mse)
df[order(df$Test_MSE),]
```

```
##                          Model Test_MSE
## 7                  Bagged Tree 22.85702
## 6                Random Forest 23.31609
## 1 Multiple Linear Regression 24.91087
## 5          Optimal Pruned Tree 27.58846
## 8                 Boosted Tree 28.08043
## 3             Ridge Regression 34.33954
## 2                        Lasso 36.11032
## 4              Regression Tree 44.54071
```

Because the bagged tree model and the random forest model have very similar test MSE, calculating their respective R^2 values will provide more information on which model is a better fit.

```r
# Computing R^2 for bagged tree model
rss <- sum((pred.bag - y.test) ^ 2)
tss <- sum((y.test - mean(y.test)) ^ 2)
rsq <- 1 - rss/tss
rsq
```

```
## [1] 0.7240906
```

```r
# R^2 for random forest
rss <- sum((pred.rf.test - y.test) ^ 2)
tss <- sum((y.test - mean(y.test)) ^ 2)
rsq <- 1 - rss/tss
rsq
```

```
## [1] 0.718549
```

```r
# rmse for bagged
rmse(y.test, pred.bag)
```

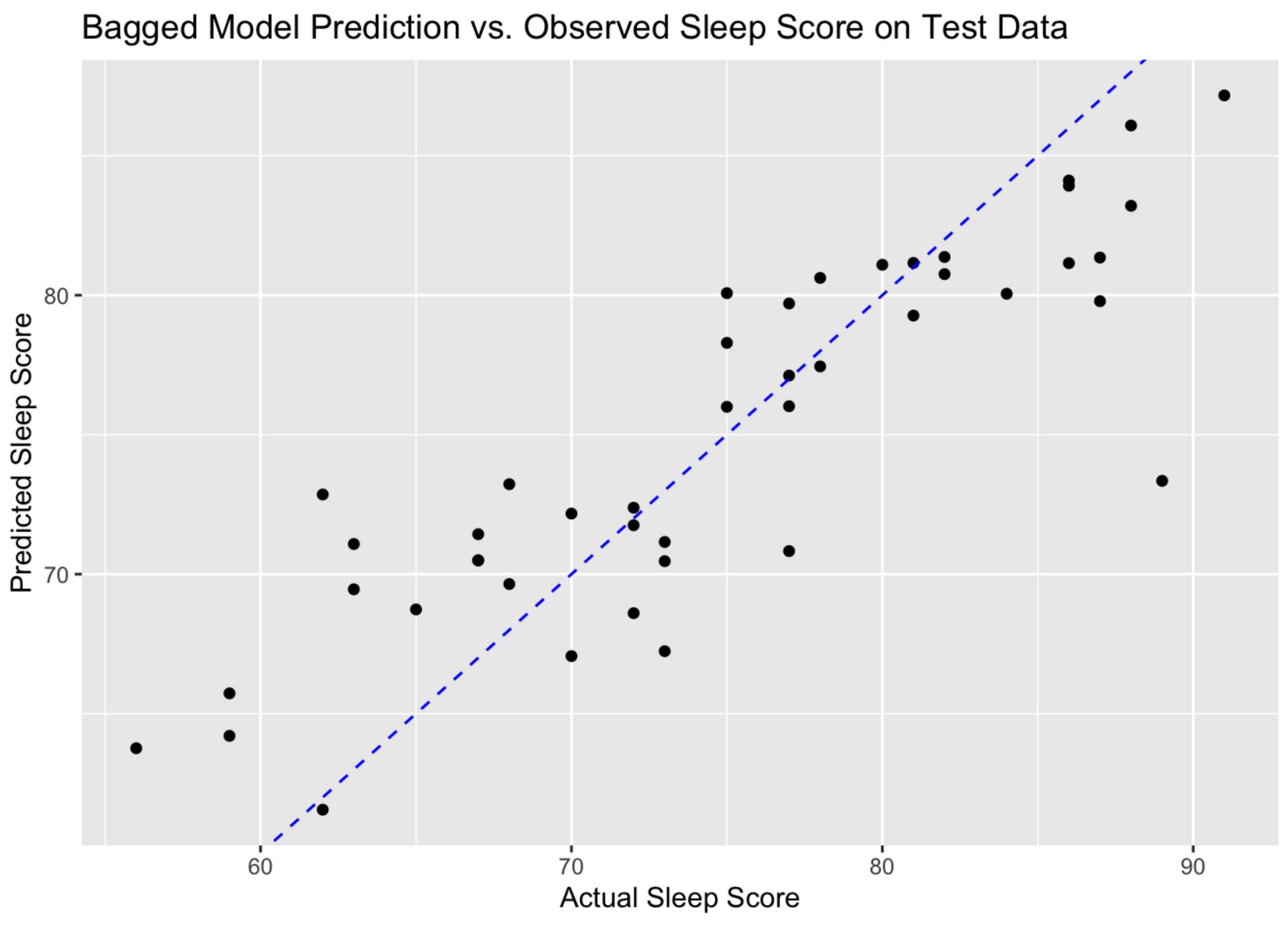
```
## [1] 4.780901
```

```r
# rmse for random forest
rmse(y.test, pred.rf.test)
```

```
## [1] 4.828674
```

72.41% of total variability can be explained by the bagged regression tree, whereas 71.85% of total variability can be explained by the random forest of regression tree. Although these differences are minuscule, I think the bagged model is the best model because it had a smaller test MSE, R^2, and `rmse` of 4.78 versus the random forest's rmse of 4.83.

```r
final_model = bag.sleep
model_pred <- predict(final_model, newdata = sleep_test)

model_df <- data.frame(actual=y.test, predicted=model_pred)
model_df %>% ggplot(aes(x=y.test, y=model_pred)) +
  geom_point() +
  geom_abline(col='blue', lty = 2) +
  labs(title = 'Bagged Model Prediction vs. Observed Sleep Score on Test Data',
       x = 'Actual Sleep Score',
       y = 'Predicted Sleep Score')
```



```
head(model_df)
```

```
##    actual predicted
## 1      81  79.27030
## 2      72  68.60260
## 3      65  68.73670
## 5      77  77.11863
## 11     82  81.37067
## 18     87  79.78600
```

Conclusion

This model may be useful to both Fitbit as a company and their consumers. By training a model to correctly predict sleep scores, Fitbit can provide ways to increase sleep score personalized to the user. The boosted model showed that the top three most important predictors of sleep score are time asleep, time in REM stage, and restlessness. A user may be getting 7-8 hours of sleep a night but still receiving low sleep scores which could be explained by a large proportion of unconscious restlessness which might suggest getting more exercise throughout the day to minimize restlessness.

The models I fit to my dataset did not perform exceptionally well, but also did not perform terribly either. I think if I were to revise this project, I would collect more observations. Not only that, but I would also like to see how Fitbit's other metrics affect sleep score such as daily steps, calories burned, and activity level, etc. Expanding the dataset would have made the model not only more interesting, but also provide Fitbit users significant pieces of information about the interactions between sleep, activity, and overall wellbeing.