Fitbit Sleep Score Model Feature Engineering

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.2     v purrr   1.0.2
## v tibble  3.2.1     v dplyr   1.1.2
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.2     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(dplyr)
library(ggplot2)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 1.1.0 --
## v broom        1.0.5     v rsample      1.1.1
## v dials        1.2.0     v tune         1.1.1
## v infer        1.0.4     v workflows    1.1.3
## v modeldata    1.2.0     v workflowsets 1.0.1
## v parsnip      1.1.0     v yardstick    1.2.0
## v recipes      1.0.7
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
## * Dig deeper into tidy modeling with R at https://www.tmwr.org
```

```r
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```r
library(yardstick)
```

Goal : How is sleep_score calculated? -> Find a model that best estimate sleep_score

After I built these models, I looked up fitbit website on their explanation of sleep score. They mentioned that sleep score is mainly based on heart beat and sleeping stages (deep sleep, awake, REM, etc.), which I realized that my dataset fitbit_df does not contain that specific data about the sleeping stages. Based on the variables I have, the number of minutes of deep sleep indeed is an important predictor. The most significant predictor is stress_score, while the correlation between sleep_score and stress_score is around 0.34, which is not that strong. This might due to the reason that stress score is calculated using similar predictors.

```r
fitbit_df <- read.csv('fitbit_data.csv')
fitbit_df$date <- as.Date(fitbit_df$date) # convert to date format
head(fitbit_df)
```

```
##         date AZM_minutes    rmssd    nremhr   entropy sleep_score
## 1 2023-06-29         157 67.89393 0.9697126 1106.6132          68
## 2 2023-06-30          34 63.09258 0.9740137  930.9208          65
## 3 2023-07-01           1 87.91776 0.9673021 1320.8890          85
## 4 2023-07-02          26 60.61797 0.9711250  950.8540          84
## 5 2023-07-03          44 96.20780 0.9771325 1310.1257          80
## 6 2023-07-04          44 89.09386 0.9704167 1309.5501          72
##   deep_sleep_min resting_heart_rate stress_score   o2_avg o2_lower_bound
## 1             96                 58           77 84.79727          70.70
## 2             65                 57           80 83.35863          93.05
## 3            106                 57           86 84.84333          86.35
## 4             90                 56           79 84.86729          86.75
## 5             78                 56           82 83.33722          90.85
## 6             63                 54           79 78.59688          71.75
##   o2_upper_bound calories
## 1           98.8  2345.97
## 2           98.4  1772.70
## 3           98.6  1669.63
## 4           98.2  1591.05
## 5           97.6  2095.86
## 6           96.8  1463.32
```

Split Data

```r
set.seed(123)
split <- initial_split(fitbit_df, prop=0.9)
train <- training(split)
test <- testing(split)
```

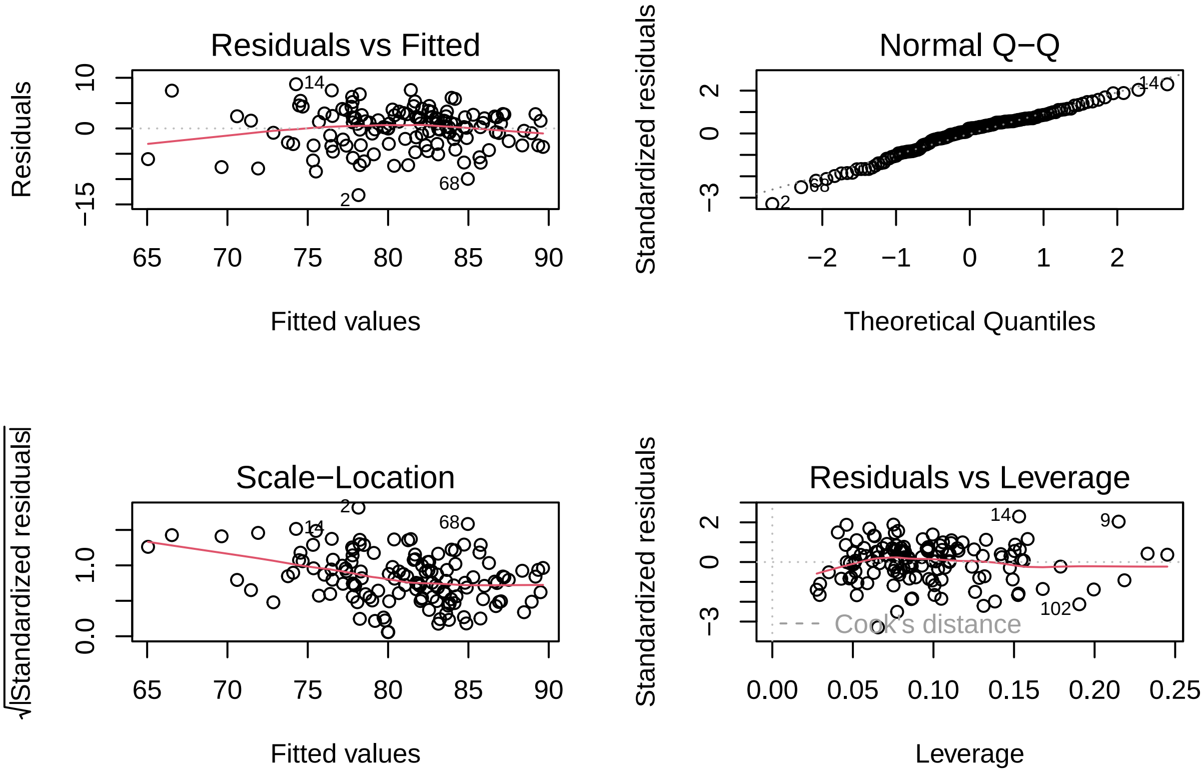
m0 : Initial Linear Model (without recipe, use original train data, all predictors)

rsq = 0.5789 rse = 4.135

```
m1 <- lm(data=train, sleep_score ~ .)
summary(m1)
```

```
##
## Call:
## lm(formula = sleep_score ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1577  -2.6379   0.9135   2.4324   8.7290
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.111e+03  1.941e+02  -5.722 7.67e-08 ***
## date                 5.010e-02  9.170e-03   5.463 2.51e-07 ***
## AZM_minutes          2.724e-02  1.204e-02   2.263 0.025426 *
## rmssd               -1.592e-01  5.304e-02  -3.002 0.003254 **
## nremhr               1.178e+02  4.508e+01   2.613 0.010099 *
## entropy              7.452e-03  2.980e-03   2.501 0.013724 *
## deep_sleep_min       7.349e-02  2.148e-02   3.421 0.000849 ***
## resting_heart_rate  2.859e-01  1.575e-01   1.815 0.071982 .
## stress_score         4.449e-01  7.203e-02   6.177 8.90e-09 ***
## o2_avg               1.761e-01  1.727e-01   1.020 0.309924
## o2_lower_bound       1.662e-02  4.072e-02   0.408 0.683947
## o2_upper_bound       4.237e-01  2.704e-01   1.567 0.119760
## calories            -9.700e-03  2.415e-03  -4.016 0.000103 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 4.135 on 122 degrees of freedom
## Multiple R-squared:  0.5789, Adjusted R-squared:  0.5374
## F-statistic: 13.97 on 12 and 122 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(m1)
```



m1 : Linear Regression (with recipe, use original train data, all predictors)

rsq = 0.6826 mrse = 3.432731 R-Squared increased by around 0.11, which means approximately 11% more of the variability in the dependent variable is explained by the predictors. In the recipe, for all numerical predictors, I normalized, Yeo-Johnson transformed, and removed correlations with threshold = 0.5. I tested several threshold for step_corr() and there is no significant difference. Since there is also a date predictor, I added step_date() and step_holiday() and it turned out to be very helpful on improving the new model.

```r
m1 <- linear_reg()

m1_recipe <- recipe(data=train, sleep_score ~ .) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_date(date, features = c("dow", "month", "year")) %>%
  step_holiday(date) %>%
  step_corr(all_numeric_predictors(), threshold = 0.5) %>%
  step_YeoJohnson(all_numeric_predictors())

m1_wkfl <- workflow() %>%
  add_model(m1) %>%
  add_recipe(m1_recipe)

m1_fit <- m1_wkfl %>%
  fit(data=train)
```

```
## Warning in stats::cor(x, use = use, method = method): the standard deviation is
## zero
## Warning: The correlation matrix has missing values. 4 columns were excluded from
## the filter.
```

```
m1_aug <- m1_fit %>%
  augment(test)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
m1_aug %>%
  metrics(truth = sleep_score, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       3.43
## 2 rsq     standard       0.683
## 3 mae     standard       2.80
```

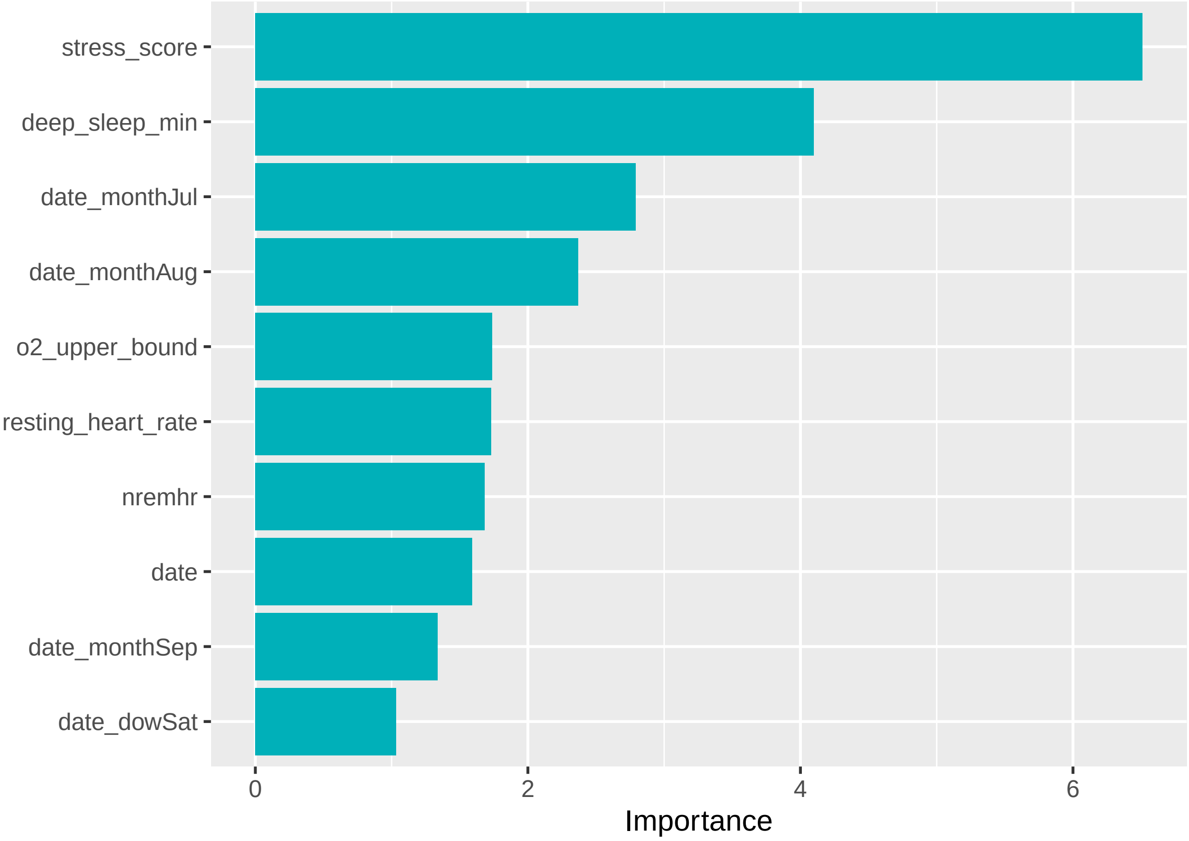
The most significant predictors are 'stress_score', 'deep_sleep_min', and 'date_monthJul'. One important thing need to be noticed is that there are several months seem to have unusual sleep_scores that are shown below, such as July, August, and September. These 3 consecutive months happen to be Summer break, which makes sense. Before these models listed, I was thinking of getting rid of 'date' column and leave all numeric columns since I was treating 'date' as ID od the dates. And the rsq was even lower at that time. After I tried adding 'date' back and also adding step_date() and step_holiday(), the model works a lot better!

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
m1_fit %>%
  extract_fit_parsnip() %>%
  vip(aesthetics = list(fill = "#00B0B9"))
```



m2 : Reduced Linear Regression (with recipe, use original train data, reduced predictors)

rsq = 0.7692907 mrse = 2.916 This time I only chose predictors that are ranked as 'highly important' by vip function. R-Squared increased by 0.19, and mrse keep reducing!

```
m2 <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

m2_recipe <- recipe(data=train, sleep_score ~ stress_score + deep_sleep_min + date) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_date(date, features = c("dow", "month", "year")) %>%
  step_holiday(date) %>%
  step_corr(all_numeric_predictors(), threshold = 0.5) %>%
  step_YeoJohnson(all_numeric_predictors())

m2_wkfl <- workflow() %>%
  add_model(m2) %>%
  add_recipe(m2_recipe)

m2_fit <- m2_wkfl %>%
  fit(data=train)
```

```
## Warning in stats::cor(x, use = use, method = method): the standard deviation is
## zero

## Warning: The correlation matrix has missing values. 4 columns were excluded from
## the filter.
```

```
m2_aug <- m2_fit %>%
  augment(test)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
m2_aug %>%
  metrics(truth = sleep_score, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       2.92
## 2 rsq     standard       0.769
## 3 mae     standard       2.40
```

Is YeoJohnson the best transformation for all numeric predictors? First of all, all (total of 2) numeric predictors I will be using, which are 'stress_score' and 'deep_sleep_min', are hard to tell if normally distributed and might need transformation.

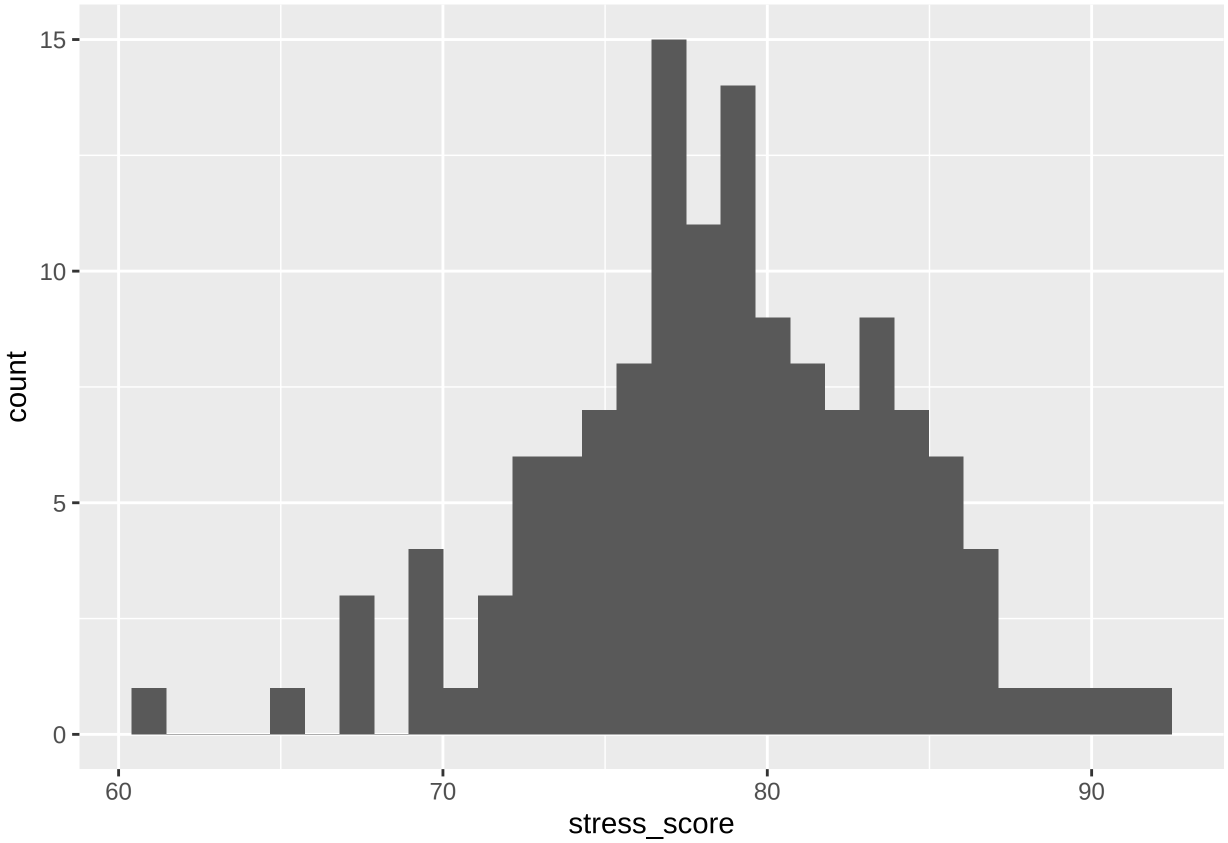
Conclusion : After binning, they show a normally distributed pattern with slight skewness. I tried log and box-cox transformation and they do not work. Without Yeo-Johnson, the fitting is slightly slower. So I keep the step_TeoJohnson() step in recipe.

Binning : stress_score

Figure 1 shows how it is distributed. Figure 2 is after log-transformation and it does not look well. So maybe it does not need a log transformation? I tried to fit them into 10 bins and now it indeed look like normally distributed.

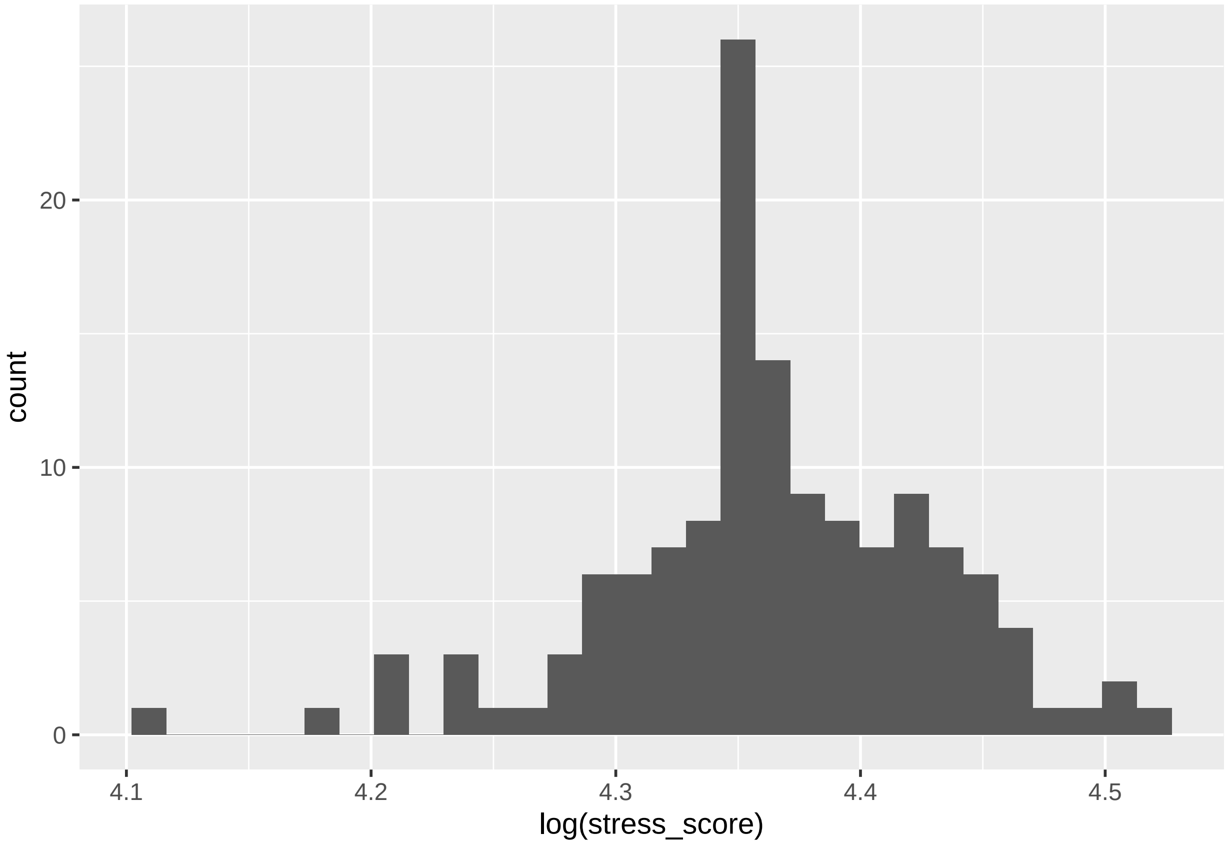
```
library(ggplot2)
ggplot(data=train, aes(x=stress_score)) + geom_histogram()
```

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

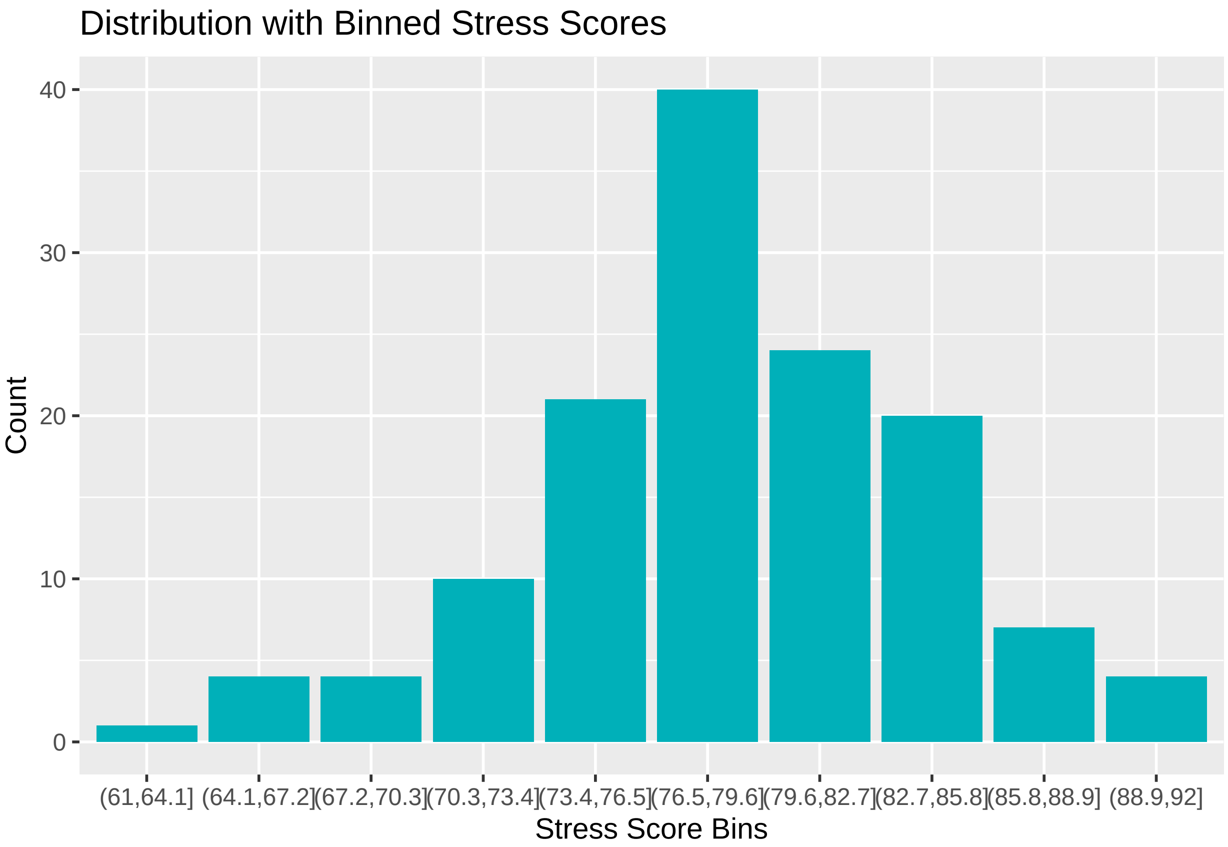


```
# log trans <- NOT working well
ggplot(data=train, aes(x=log(stress_score))) + geom_histogram()
```

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



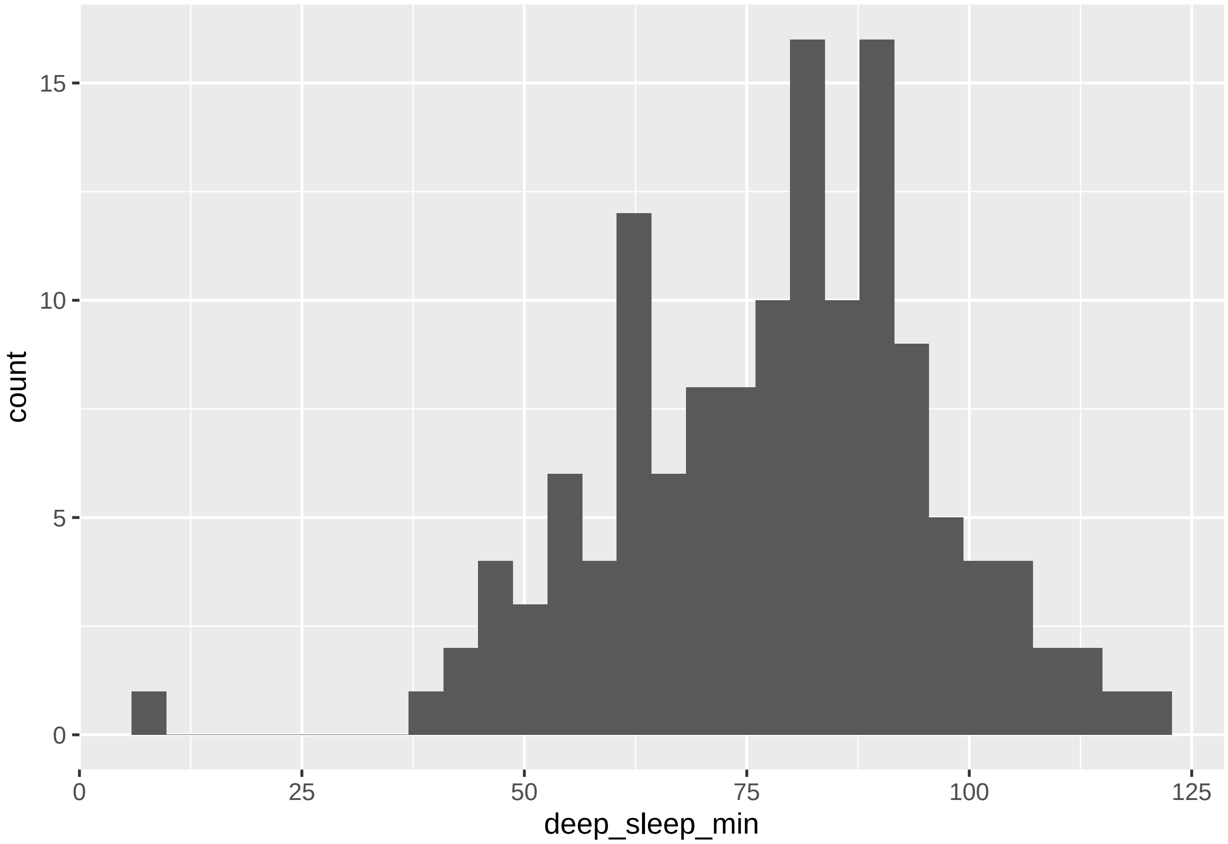
```
# binning <- work well!!
stress_bins <- cut(train$stress_score, breaks = 10)  # 10 bins
stress_binned <- data.frame(stress_score = train$stress_score, stress_bins)
ggplot(stress_binned, aes(x = stress_bins)) +
  geom_bar(fill = "#00B0B9") +
  labs(title = "Distribution with Binned Stress Scores", x = "Stress Score Bins", y = "Count")
```



## Binning : deep_sleep_min Same for deep_sleep_min, after binning, despite it looks left-skewed, it still shows a normally distributed pattern.

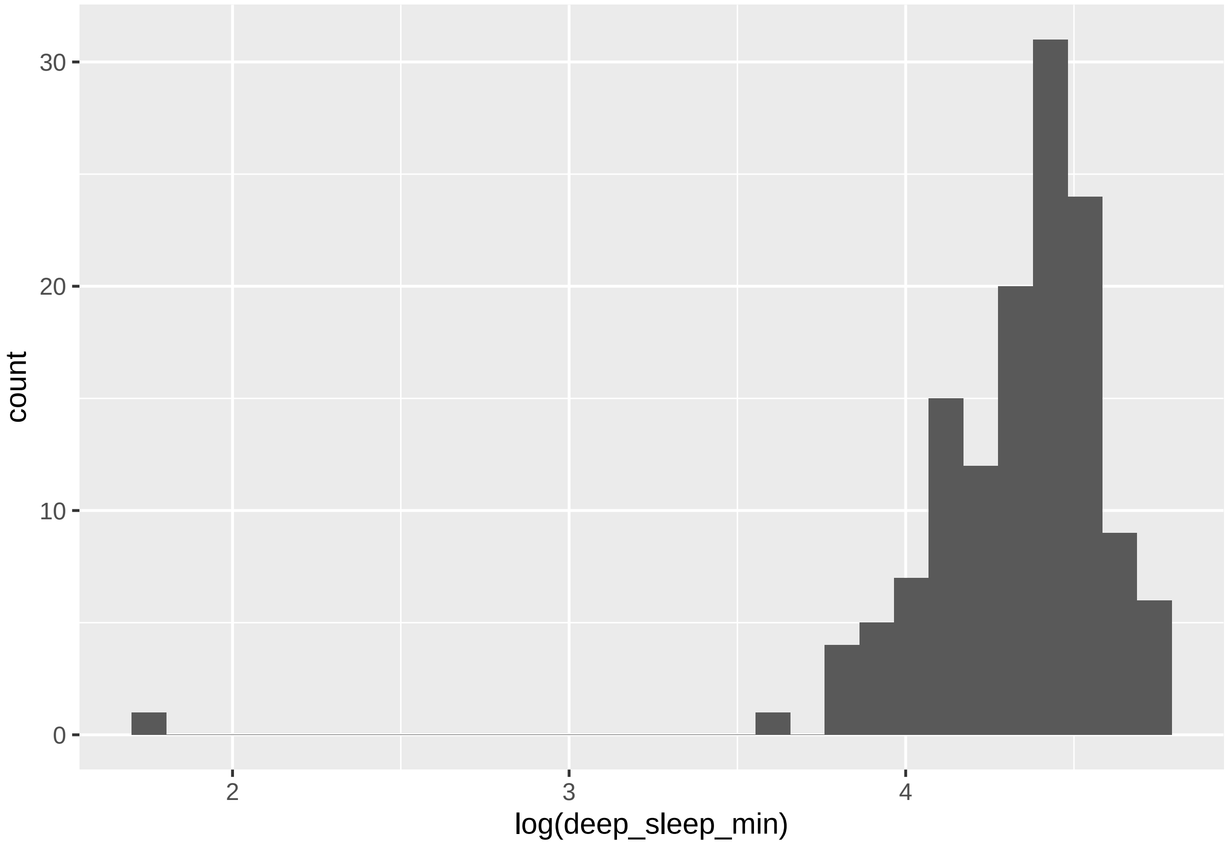
```
ggplot(data=train, aes(x=deep_sleep_min)) + geom_histogram()
```

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

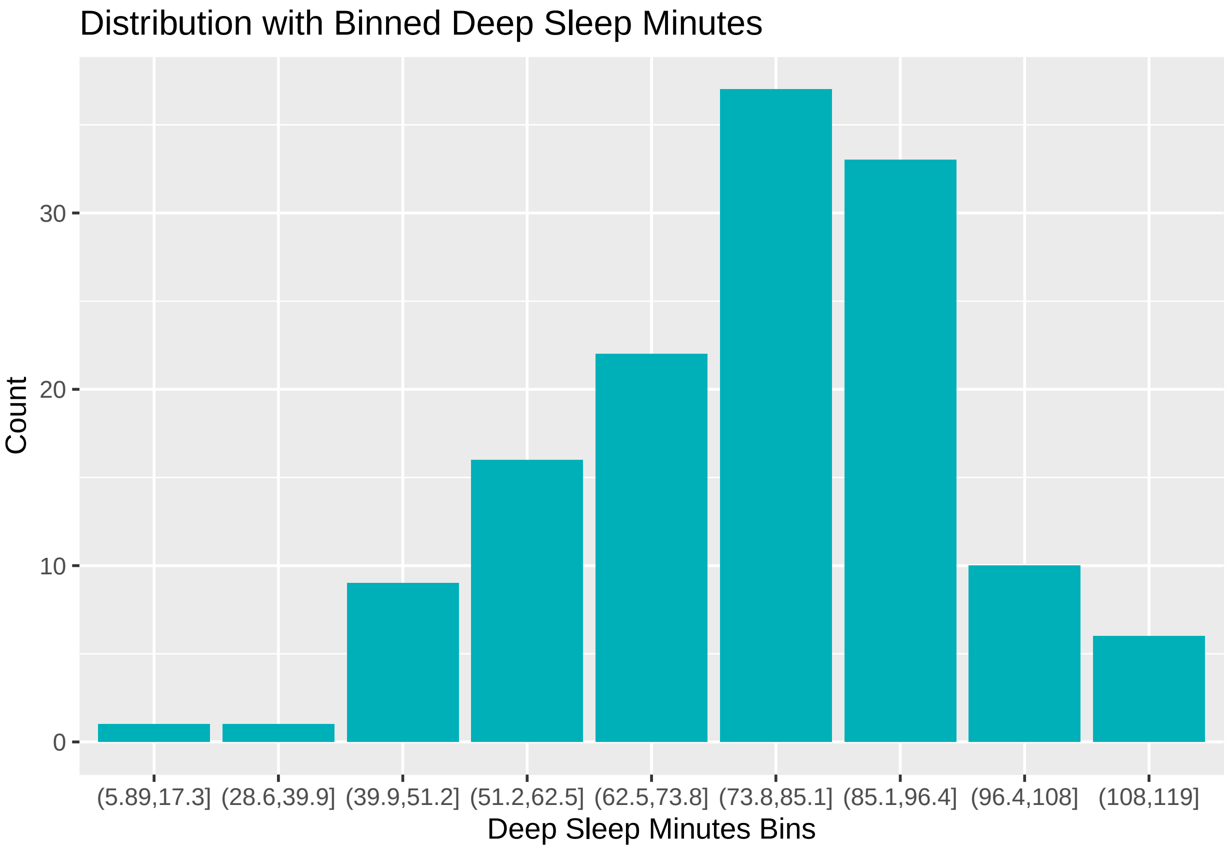


```r
# log transformation <- worse
ggplot(data=train, aes(x=log(deep_sleep_min))) + geom_histogram()
```

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



```
# binning
deep_bins <- cut(train$deep_sleep_min, breaks = 10)
deep_binned <- data.frame(deep_sleep_min = train$deep_sleep_min, deep_bins)
ggplot(deep_binned, aes(x = deep_bins)) +
  geom_bar(fill="#00B0B9")+
  labs(title = "Distribution with Binned Deep Sleep Minutes", x = "Deep Sleep Minutes Bins", y = "Count
```



Influential Points

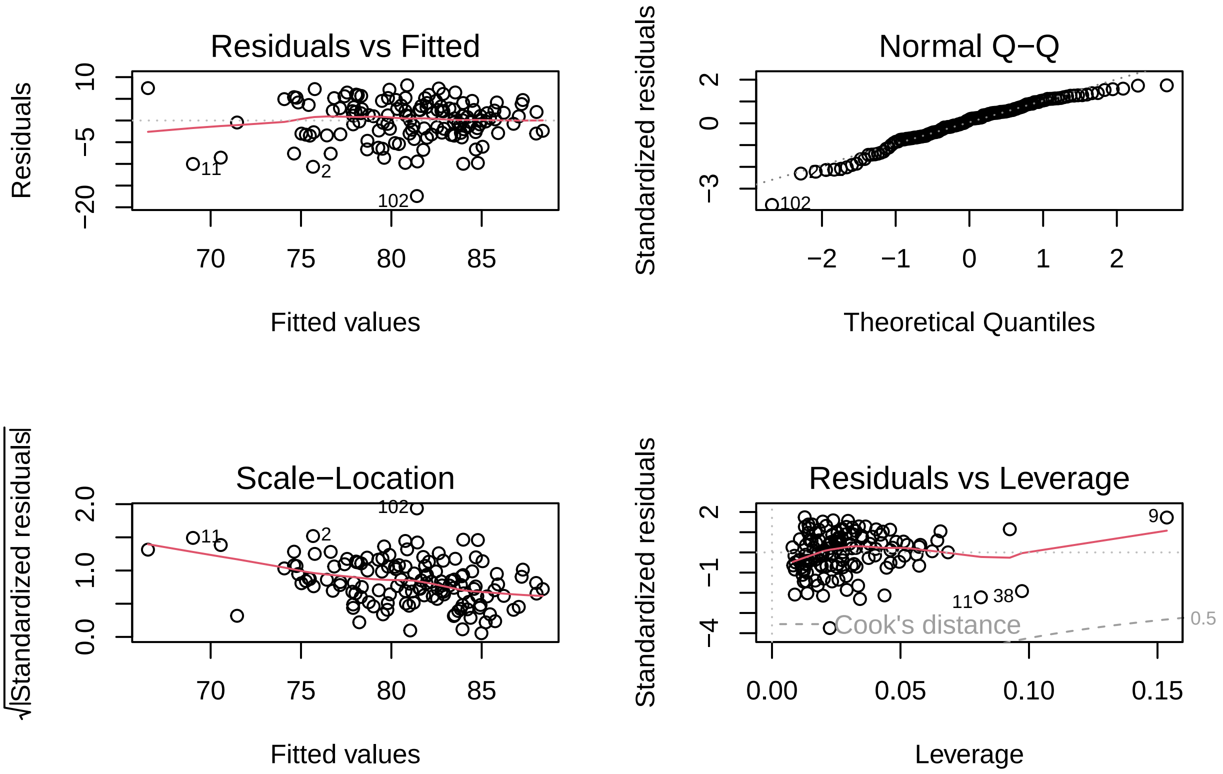
The last step I did is to check influential points such as Leverage Points and Outliers. Both are not working with workflow so I have to create a lm model with the same predictors I am using for reduced (better) model.

```
lm <- lm(data=train, sleep_score ~ stress_score + deep_sleep_min + date)
summary(lm)
```

```
##
## Call:
## lm(formula = sleep_score ~ stress_score + deep_sleep_min + date,
##     data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.4118  -2.8795   0.7954   3.2943   8.1244
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -968.33545  155.82130  -6.214 6.37e-09 ***
## stress_score      0.38039    0.07798   4.878 3.05e-06 ***
## deep_sleep_min    0.11025    0.02258   4.883 2.99e-06 ***
## date              0.05151    0.00794   6.487 1.64e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 4.701 on 131 degrees of freedom
## Multiple R-squared:  0.4155, Adjusted R-squared:  0.4021
## F-statistic: 31.04 on 3 and 131 DF,  p-value: 3.183e-15
```

```
par(mfrow=c(2,2))
plot(lm)
```



## Leverage Points : 9 38 147 15 11 25 36 83 105 124

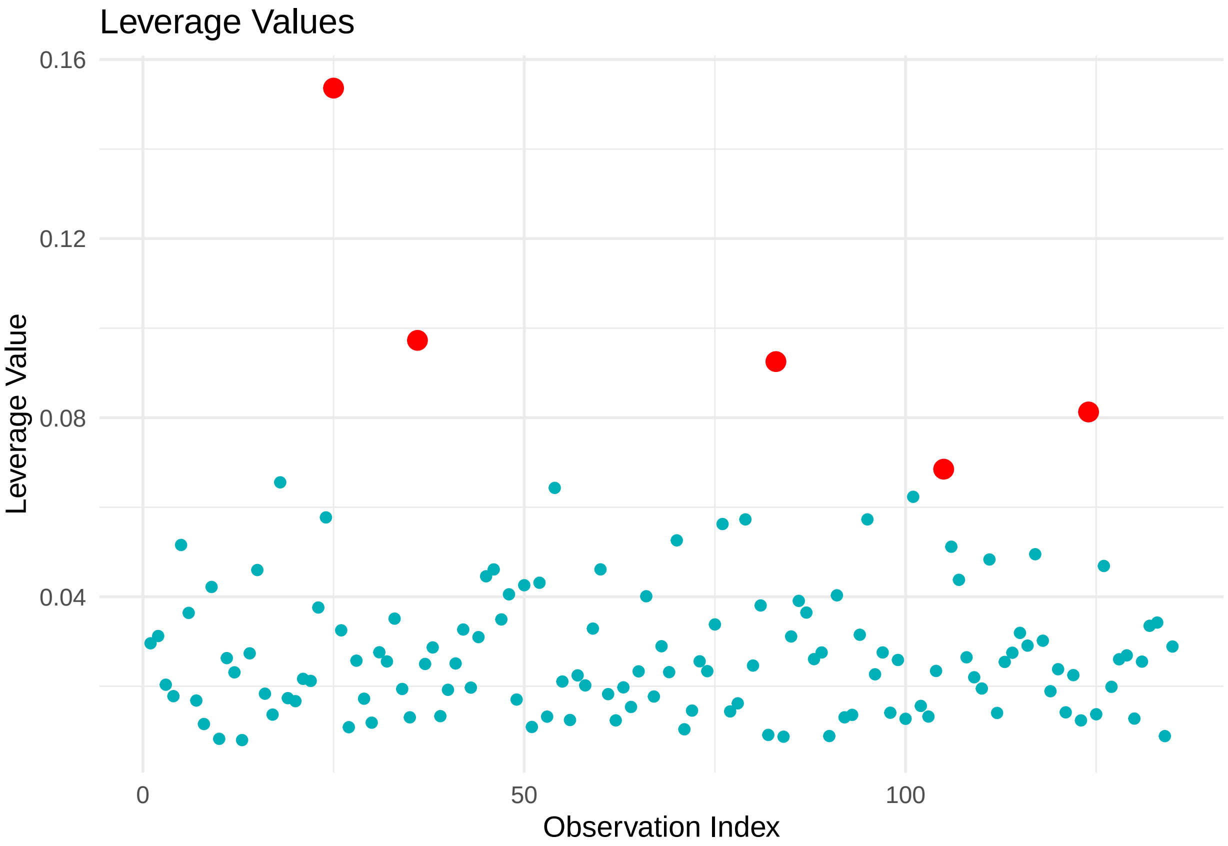
```
leverage_values <- hatvalues(lm)
n <- nrow(train)
k <- 3  # Number of predictors
leverage_threshold <- 3 * k / n
high_leverage_points <- which(leverage_values > leverage_threshold)

print(high_leverage_points)
```

```
##   9  38 147  15  11
##  25  36  83 105 124
```

```
leverage_df <- data.frame(
  Observation = 1:nrow(train),
  Leverage = leverage_values
)

ggplot(leverage_df, aes(x = Observation, y = Leverage)) +
  geom_point(color = "#00B0B9") +
  geom_point(data = leverage_df[high_leverage_points, ], color = "red", size = 3) +
  ggtitle("Leverage Values") +
  xlab("Observation Index") +
  ylab("Leverage Value") +
  theme_minimal()
```

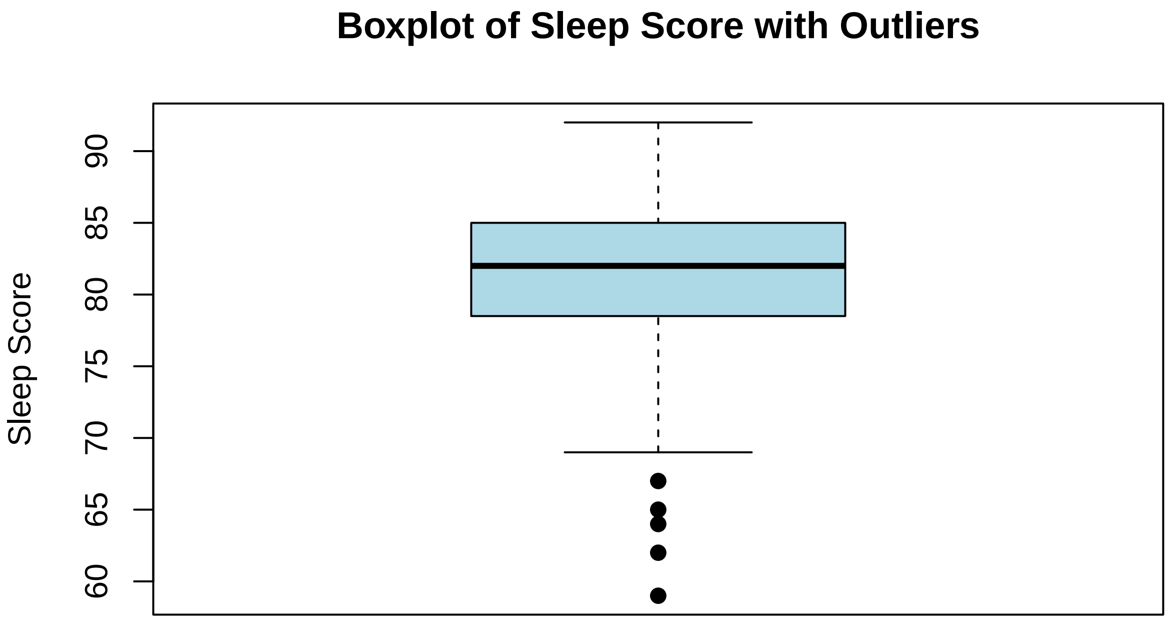


Outliers : 36 122 124 132 133

```
q <- quantile(train$sleep_score, c(0.25, 0.75))
iqr <- IQR(train$sleep_score)
threshold <- 1.5 * iqr
outliers <- which(train$sleep_score < (q[1] - threshold) | train$sleep_score > (q[2] + threshold))
print(outliers)
```

```
## [1]  36 122 124 132 133
```

```
boxplot(train$sleep_score, main = "Boxplot of Sleep Score with Outliers",
        ylab = "Sleep Score", col = "lightblue", pch = 19)
points(outliers, train$sleep_score[outliers], col = "red", pch = 19)
```



m3 : Final Model (with recipe, reduced predictors, reduced train data)

rsq = 0.79 rmse = 2.6244766

R-Squared once again increased by around 0.02, which is approxmately 2% of more data points could be explained by predictors. rmse once again reduced by around 0.3. This model is trained by the reduced training data, cleaned_train, which removed data points that are both determined as high leverage points and outliers from the previous steps. Others stay the same as m2 model.

```r
# new reduced train data
combined_outliers <- intersect(high_leverage_points, outliers)
cleaned_train <- train[-combined_outliers, ]

m3 <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

m3_recipe <- recipe(data=cleaned_train, sleep_score ~ stress_score + deep_sleep_min + date) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_date(date, features = c("dow", "month", "year")) %>%
  step_holiday(date) %>%
  step_corr(all_numeric_predictors(), threshold = 0.5) %>%
  step_YeoJohnson(all_numeric_predictors())

m3_wkfl <- workflow() %>%
  add_model(m3) %>%
  add_recipe(m3_recipe)

m3_fit <- m3_wkfl %>%
  fit(data=cleaned_train)
```

```
## Warning in stats::cor(x, use = use, method = method): the standard deviation is
## zero

## Warning: The correlation matrix has missing values. 4 columns were excluded from
## the filter.
```

```
m3_aug <- m3_fit %>%
  augment(test)
```

```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
m3_aug %>%
  metrics(truth = sleep_score, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       2.62
## 2 rsq     standard       0.792
## 3 mae     standard       2.15
```

Evaluation of Final Model m3

```
m3_pred <- test %>%
  bind_cols(m3_fit %>%
    predict(new_data = test) %>% # inside of bind_cols()
    rename(predictions = .pred))
```

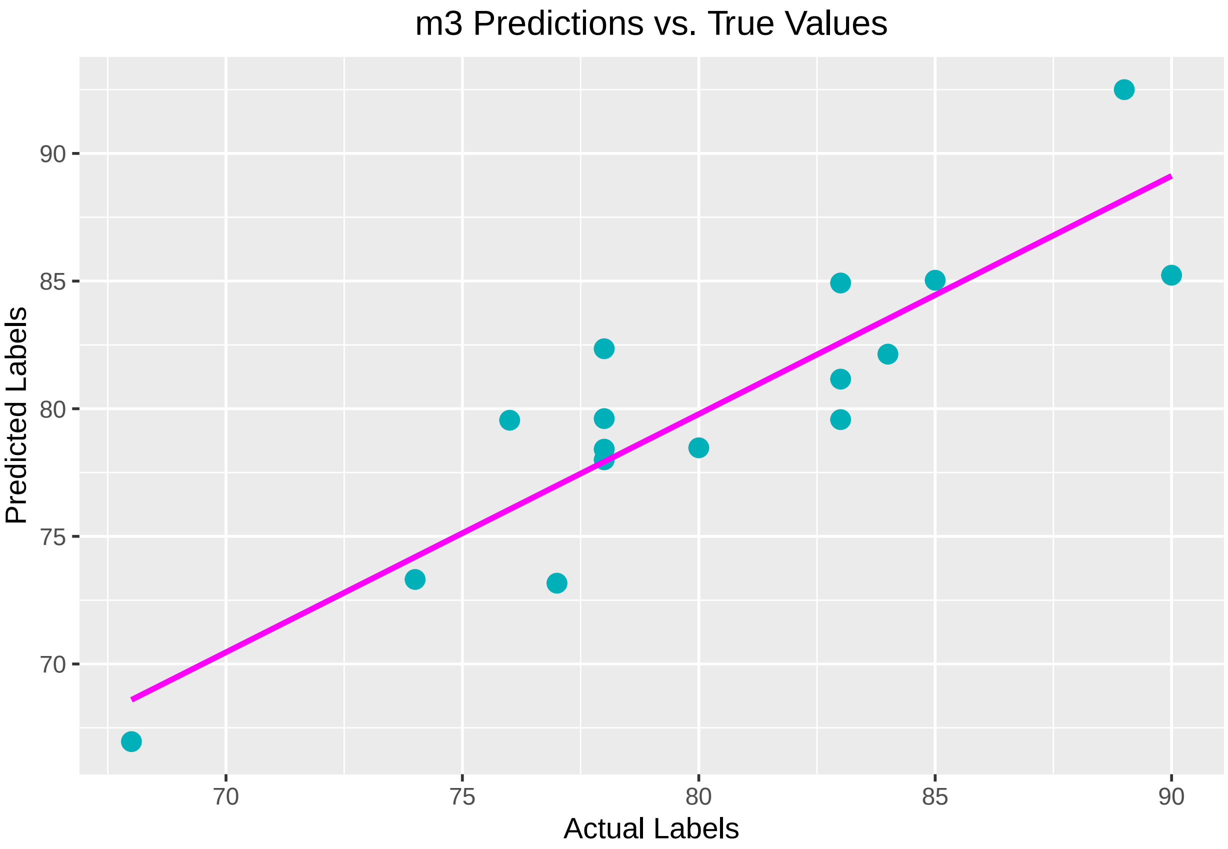
```
## Warning in predict.lm(object = object$fit, newdata = new_data, type =
## "response"): prediction from a rank-deficient fit may be misleading
```

```
m3_pred %>%
  select(c(sleep_score, predictions)) %>%
  slice_head(n = 5)
```

```
##   sleep_score predictions
## 1          68    66.95888
## 2          85    85.03298
## 3          78    79.61222
## 4          84    82.13733
## 5          77    73.16515
```

```
m3_pred %>%
  ggplot(mapping = aes(x = sleep_score, y = predictions)) +
  geom_point(size = 3, color = "#00B0B9") +
  geom_smooth(method = "lm", se = FALSE, color = 'magenta') +
  ggtitle("m3 Predictions vs. True Values") +
  xlab("Actual Labels") +
  ylab("Predicted Labels") +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



Besides of models shown above, I have also tried Random Forest, Gradient Boosting, LASSO, Ridge. Some are not working (LASSO, Ridge), others (Random Forest, Gradient Boosting) are not working as good as Linear Regression model m3.