```
knitr::opts_chunk$set(echo = TRUE)
# import libraries
library(tidyverse)   # general
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)       # model
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(DescTools)   # descriptive statistics
```

```
##
## Attaching package: 'DescTools'

## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
library(ggcorrplot)  # correlation matrix plot
# import data set
data <- read.csv("2019.csv")
```

Introduction

Predictive machine learning algorithms are often at the forefront of modern computational science. Autonomously estimating results given troves of simulated data is one of the most powerful uses of machine learning. Examples vary from determining exoplanet existence based on light flux data to leveraging the flow of the US bond and stock markets. Depending on the type of data you are processing, prediction can be accomplished through classification models, random forest, k-nearest neighbors or many other machine learning algorithms. This project employs regression analysis in order to study the happiness of countries.

The World Happiness Report, uploaded to Kaggle by the Sustainable Development Solutions Network, is described as "a landmark survey of the state of global happiness." The most recent survey was conducted throughout 2019 and ranks all countries with a happiness score. Particular factors of a society are included with the score that help gauge the livelihood in that country. A glimpse of our data is shown below:

```
glimpse(data)
```

```
## Rows: 156
## Columns: 9
## $ Overall.rank                <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13~
## $ Country.or.region           <chr> "Finland", "Denmark", "Norway", "Iceland"~
## $ Score                       <dbl> 7.769, 7.600, 7.554, 7.494, 7.488, 7.480,~
## $ GDP.per.capita              <dbl> 1.340, 1.383, 1.488, 1.380, 1.396, 1.452,~
## $ Social.support              <dbl> 1.587, 1.573, 1.582, 1.624, 1.522, 1.526,~
## $ Healthy.life.expectancy     <dbl> 0.986, 0.996, 1.028, 1.026, 0.999, 1.052,~
## $ Freedom.to.make.life.choices <dbl> 0.596, 0.592, 0.603, 0.591, 0.557, 0.572,~
## $ Generosity                  <dbl> 0.153, 0.252, 0.271, 0.354, 0.322, 0.263,~
## $ Perceptions.of.corruption   <dbl> 0.393, 0.410, 0.341, 0.118, 0.298, 0.343,~
```

The data being used has 156 countries listed with 9 informative columns. These columns are happiness rank, country name, overall happiness score, and six factors that aid in defining the happiness score: GDP per capita, Social Support, Life Expectancy, Freedom, Generosity, and Truth.

The scores are based on answers from the Gallup World Poll to the main life evaluation question asked in the poll. This question, known as the *Cantril ladder*, asks respondents to think of a ladder with the best possible life for them being a 10 and the worst possible life being a 0 and to rate their own current lives on that scale. The emblematic Orwellian dystopia, a society in which the above aspects are destructive to the welfare of a free and open society, is most closely associated with the lowest scoring countries. Therefore, higher scores represent, essentially, a closer approximation to a Utopia.

Confusion exists as to how the numbers for each factor apply to the happiness score (discussions can be found here and here). These values represent the effect that each societial variable adds to a standard dystopian score. Similarly, having higher factor scores will lead to a higher happiness score. It is worth mentioning that previous versions of the happiness report have a dystopia residual that represent more general improvements to the standard dystopian. This is not available in the most recent (2018, 2019) data sets.

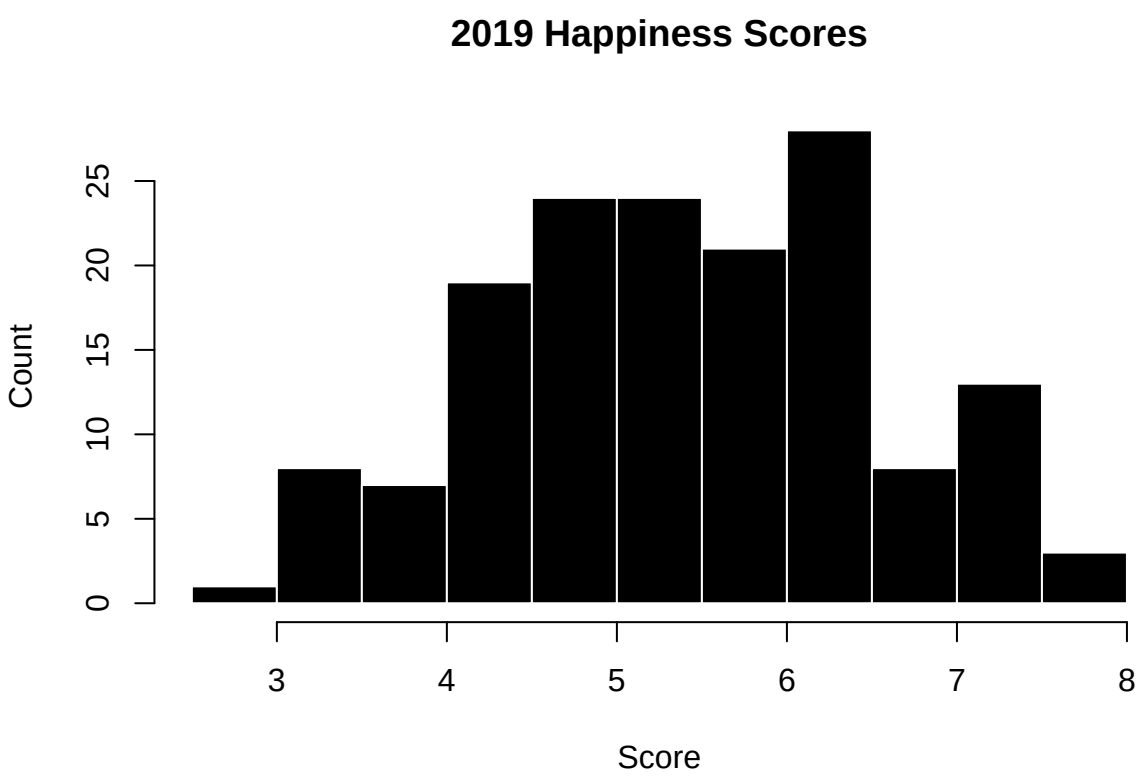
It is important to note that the factor values themselves are not *directly* correlated with the overall happiness score. The goal of this report is to determine the extent in which they are correlated. The previously mentioned discussions state the best model one can use to estimate happiness scores is the sum of the individual factors, that is `sum(data$[factors]) = data$Score`. This model is presented along with a generalized linear model equation based on 2019 data alone and 2018/2019 data together.

The rest of the report is organized as follows: second, the data set is dissected and visualized in the **Data** section. Third, the data is partitioned and the three modeling methods are shown in the **Methods** section. Fourth, the three models are compared in the **Results** section. Finally, the **Conclusion** section summarizes and compares the results.

Data

The qualitative structure and the general relationship of the data is described above. The data found here is quantitatively normal. A histogram of the happiness scores shows an approximately normal distribution and descriptive statistics support this claim.

```
hist(data$Score, freq=TRUE, col="black", border="white",
     main="2019 Happiness Scores", xlab="Score", ylab="Count")
```




```
summary(data$Score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.853   4.545   5.380   5.407   6.184   7.769
```

The histogram is right-leaning. This can be shown quantitively by comparing the mean score (5.4070962) and the median score (5.3795). A mean greater than the median signifies that there are more values larger that the middle. The range of scores is (2.853, 7.769). The three highest and lowest observations are displayed below:

```
data %>% filter(Overall.rank <=3) %>% select(Overall.rank, Country.or.region, Score)
```

```
##   Overall.rank Country.or.region Score
## 1            1           Finland 7.769
## 2            2           Denmark 7.600
## 3            3            Norway 7.554
```

```
data %>% filter(Overall.rank >= 154) %>% select(Overall.rank, Country.or.region, Score)
```

```
##   Overall.rank        Country.or.region Score
## 1          154              Afghanistan 3.203
## 2          155 Central African Republic 3.083
## 3          156              South Sudan 2.853
```

Remember that low happiness scores tend to be more like dystopian societies. This means **lower happiness scores tend to have lower factor scores**. Thus, through deduction, high factor scores have a positive effect and low factor scores have a negative effect on the happiness score. Ultimately, determining the correlation of each factor to each other is of utmost interest. The `ggcorrplot` library is used to probe this:

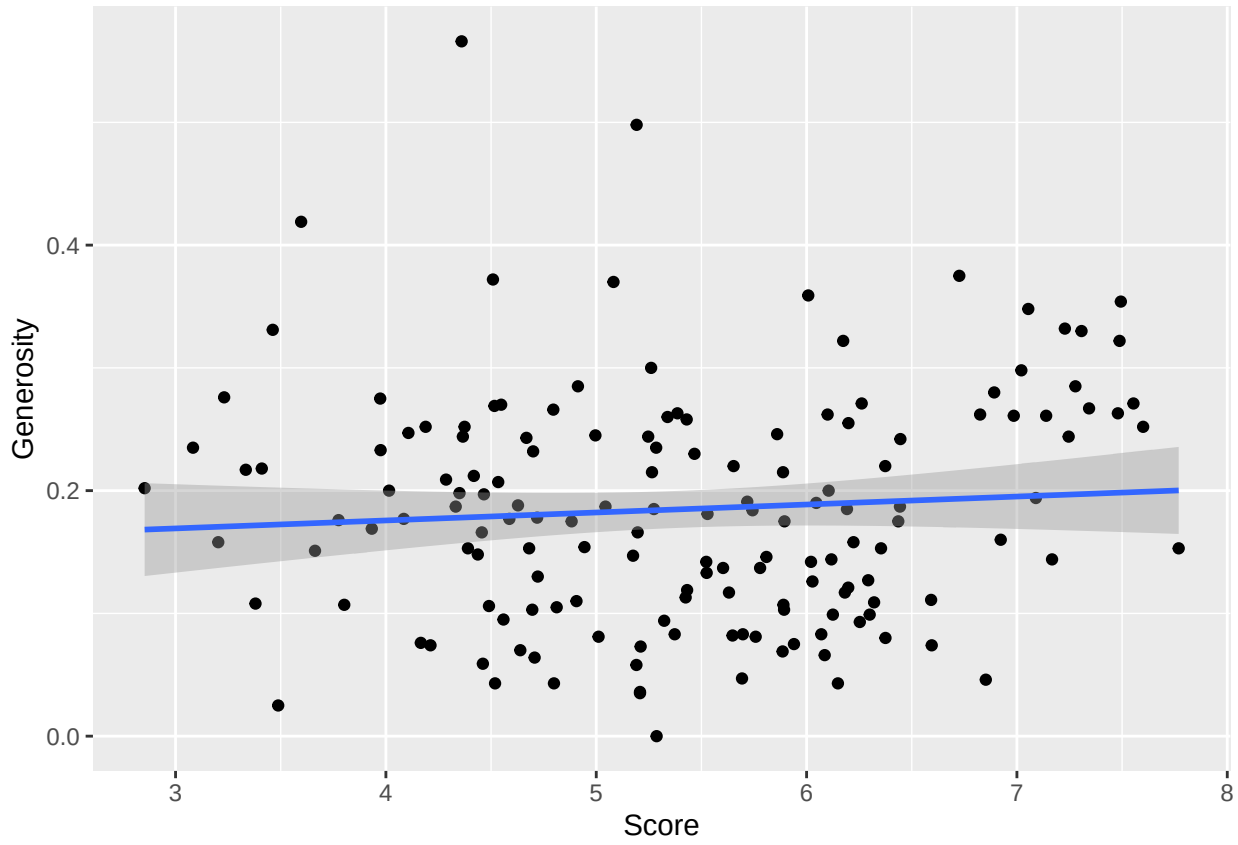
```
temp <- data[, c(3,4,5,6,7,8,9)]
cormat <- signif(cor(temp), 2)
ggcorrplot(cormat)
```


```
rm(temp, cormat)
```

All factors are at least somewhat related (above zero) with the partial exception of `Generosity`. The `Generosity` factor seems to have less of an effect on the happiness score with a correlation of around zero, thus, *removing the* ***`Generosity`*** *term in the model may improve the accuracy.* The relationships between specific variables can be portrayed more directly by plotting them and determining the line of best fit. For instance, compare `Score` versus `Generosity` and `Score` versus `GDP per capita`:

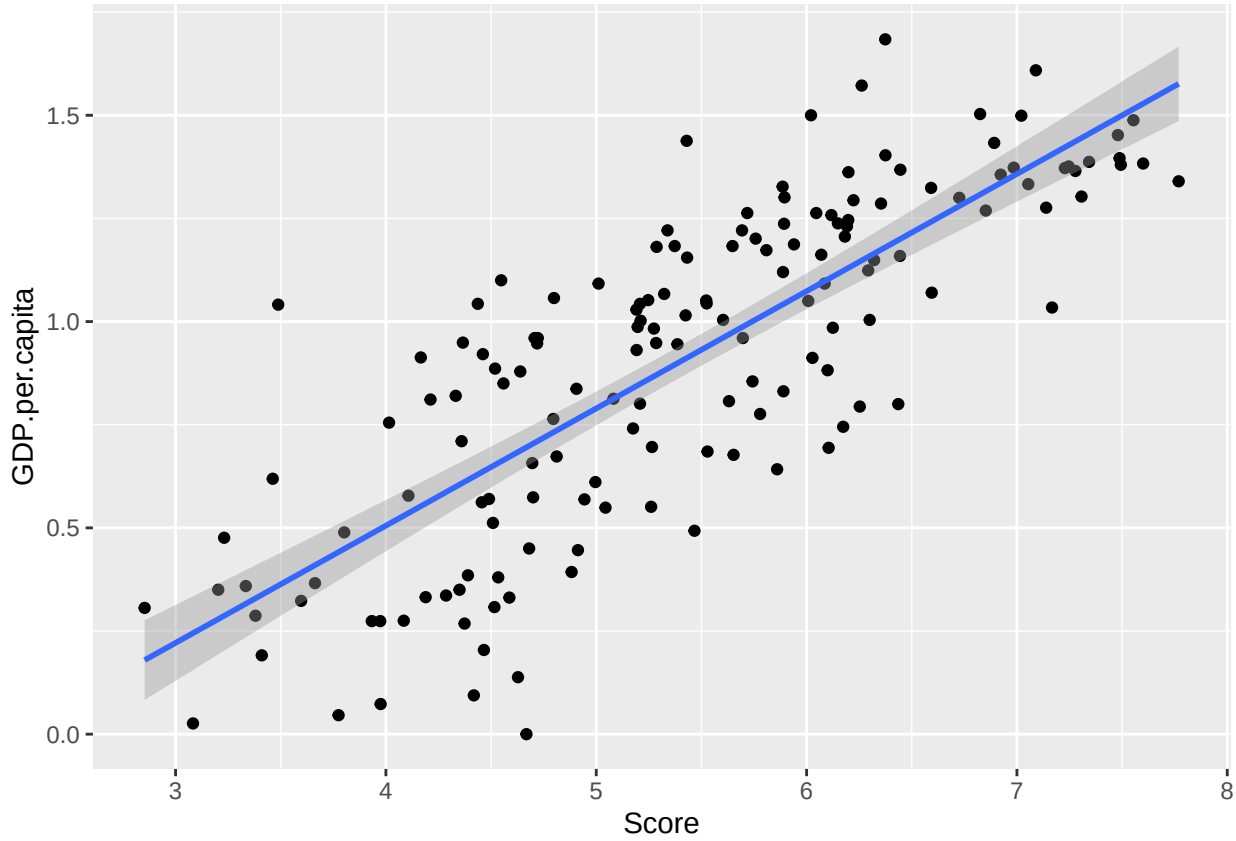
```
ggplot(data = data, aes(x = Score, Generosity)) +
  geom_point(color='black') +
  geom_smooth(method = "lm", se = TRUE)
```

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




```
ggplot(data = data, aes(x = Score, GDP.per.capita)) +
  geom_point(color='black') +
  geom_smooth(method = "lm", se = TRUE)
```

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



A horizontal line of best fit shows the lack of correlation between score and generosity. Recall that the correlation between score and generosity on the correlation map was close to zero. Conversely, the plot of Score and GDP per capita clearly shows a positive correlation between the two: When `GDP per capita` increases, `Score` also increases. Also note the line of best fit for generosity has relatively high confidence bands compared to the GDP per capita plot, realizing it is a rather unstable parameter.

Visualizing data patterns may offer clues of intriguing routes to choose when performing analysis but it doesn't provide any noteworthy prediction capabilities. This is where machine learning techniques begin to excel. In the next section, we build several models to determine the best way to predict a happiness score.

Methods

Model 1: The Sum of Factors

The first model is seemingly the most obvious and was proposed on a discussion forum from the Kaggle website. It states that the "perfect" prediction model is to simply take the sum of all factors as the happiness score. This model is attempted with one caveat: a "standard dystopia score" was discovered in earlier happiness reports and was given the value 1.85. This value is added to our predicted scores as well because each factor is a ranking of how much *better* the country is than the standard dystopia. Note the use Root Mean Square Error (RMSE) as a success indicator. This choice is further explained in the results section.

```
# find  predicted score by sum method and calculate the corresponding RMSE
sum_model <- data %>% mutate(pred_score = GDP.per.capita +
                               Social.support +
                               Healthy.life.expectancy +
                               Freedom.to.make.life.choices +
```

```
                              Generosity +
                              Perceptions.of.corruption +
                              1.85,
                            RMSE = RMSE(Score, pred_score))
# show top results of the summation model
sum_model %>%
  filter(Overall.rank <= 5) %>%
  select(Overall.rank, Country.or.region, Score, pred_score, RMSE)
```

```
##   Overall.rank Country.or.region Score pred_score      RMSE
## 1            1           Finland 7.769      6.905 0.5280065
## 2            2           Denmark 7.600      7.056 0.5280065
## 3            3            Norway 7.554      7.163 0.5280065
## 4            4           Iceland 7.494      6.943 0.5280065
## 5            5       Netherlands 7.488      6.944 0.5280065
```

```
# save RMSE for the first model
mod1_rmse <- RMSE(sum_model$Score, sum_model$pred_score)
```

All predicted scores shows the same RMSE! In addition to the dystopian standard score, previous Happiness Report data sets also have a "dystopian residual" that contributes to the happiness score. Since the residual is not shown or described in this data set, it is deemed inappropriate to introduce it ourselved into the model. Although, without this value, the data set does seem incomplete. In spite of its absence, the dystopian residual column is calculated and a snippet of these are shown below. This is calculated under the assumption the summation model is perfect. Simply subtracting all factors and the standard dystopia value from the happiness score will yield the residual. There is no mention of the missing residual in any recent discussions to the 2018 or 2019 dataset, therefore it is omitted in subsequent modeling techniques. We can conclude that the sum of factors model is not perfectly accurate as portrayed in the discussion boards; it has a RMSE of 0.5280065.

```
# calculate the missing dystopian residuals
sum_model <- sum_model %>% mutate(residual = Score - pred_score)
# show top results of the summation model
sum_model %>%
  filter(Overall.rank <= 5) %>%
  select(Overall.rank, Country.or.region, Score, pred_score, RMSE, residual)
```

```
##   Overall.rank Country.or.region Score pred_score      RMSE residual
## 1            1           Finland 7.769      6.905 0.5280065    0.864
## 2            2           Denmark 7.600      7.056 0.5280065    0.544
## 3            3            Norway 7.554      7.163 0.5280065    0.391
## 4            4           Iceland 7.494      6.943 0.5280065    0.551
## 5            5       Netherlands 7.488      6.944 0.5280065    0.544
```

Model 2: The 2019 GLM Model

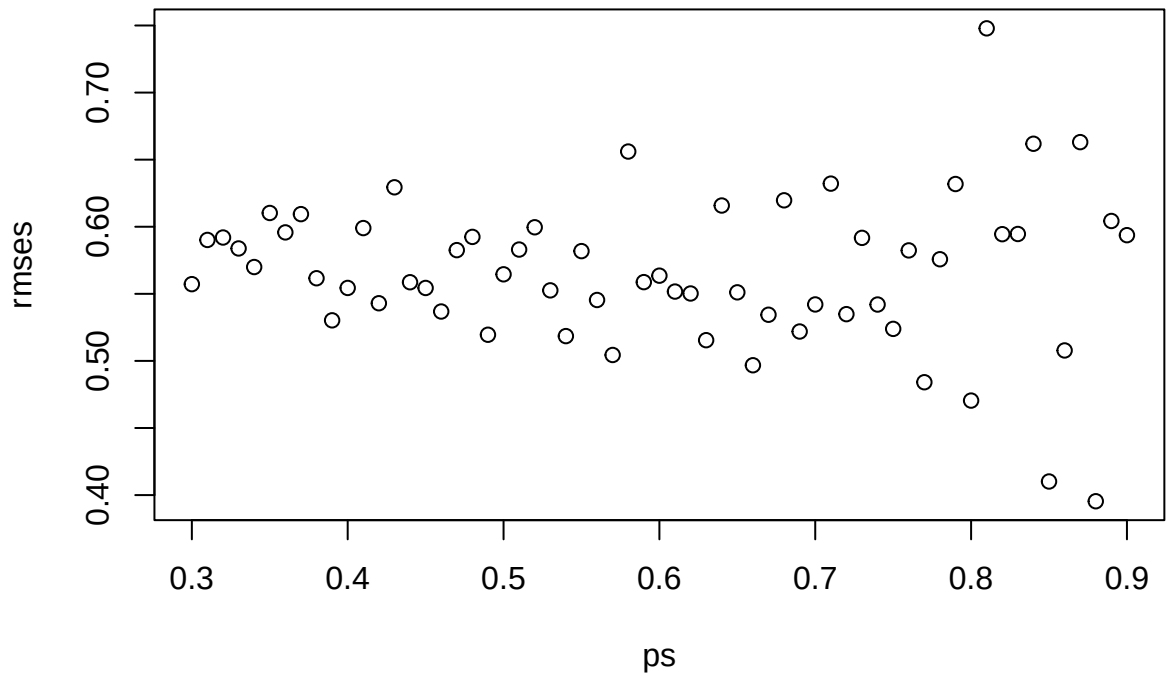
Before our first linear regression model is applied, the data must be partitioned into a training and test set. This step is common when employing machine learning algorithms that require a check on the goodness of fit. It reduces the probability of overfitting to our training data at the expense of our prediction model. This was not completed for our sum of factors model because a model did not need to be trained, the equation was simply `sum(data$[factors])`.

The Happiness Report has a little over 150 country observations and six factors in which we will condition our model. Given the relatively low number of observations compared to the amount of factors, the model may have a tendency to overfit to the training data by overweighting unimportant variables. This is almost unavoidable when working with low volumes of data. The reality of regression is that you can *always* find a model that fits your training data exactly but which is typically useless for prediction. Keeping this in mind, an optimal training-test data split ratio, that is 70 training:30 test, 80 training:20 test, . . . , etc., is first determined.

```
# --- test for an appropriate ratio in data partitioning
# a sequence of p's we want to test
ps <- seq(from=.30, to=.90, by=.01)
# calculate RMSEs for each p value
rmses <- sapply(ps, function(p){
  train_index <- createDataPartition(data$Score, times=1, p=p, list=FALSE)
  train <- data[train_index,]
  test <- data[-train_index,]
  fit <- glm(Score ~ GDP.per.capita +
               Social.support +
               Healthy.life.expectancy +
               Freedom.to.make.life.choices +
               Generosity +
               Perceptions.of.corruption,
             data = train)
  test <- test %>% mutate(pred_score = predict.glm(fit, newdata=test))
  RMSE(test$Score, test$pred_score)
})
```

The tested model in the partitioning is explored after this section. Plotting `rmses` versus `p` shows a slight pattern: when you increase the training data size, our RMSE decreases. Intuitively, this makes sense. The data is quite correlated as it has already been shown and the model is being allowed to work with more training data to make better predictions in the test set. From the plot below, the lowest RMSE is 0.3954055 with a ratio of 0.88:0.12.

```
# no real clear winner in terms of best accuracy in probabilities
plot(ps, rmses)
```



While useful in achieving a low RMSE, employing only 0.12 percent of our data to test does not leave much in terms of prediction. Ultimately, an arbitrary value of 0.70 is chosen because RMSE seems to become more sporadic after this value. Future models in the *methods* section use 0.70 as well. This ratio is also kept when the current data is supplemented with more data.

```
# set seed to keep partitioning consistent
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# ----- Data partitioning -----
train_index <- createDataPartition(data$Score, times=1, p=0.70, list=FALSE)
train <- data[train_index,]
test <- data[-train_index,]
```

With our data partitioned 0.70:0.30, a generalized linear model is fitted using the `caret` package. `Score` is predicted using all six factors. The first five predicted scores are shown.

```
# --- fit our glm model, caret::glm
fit <- glm(Score ~ GDP.per.capita +
             Social.support +
             Healthy.life.expectancy +
             Freedom.to.make.life.choices +
             Generosity +
```

```
            Perceptions.of.corruption,
          data = train)
# add predicted scores to a 'results' data frame
results <- test %>%
  mutate(pred_score = predict.glm(fit, newdata=test))
```

```
# show top five observations
results %>%
  select(Overall.rank, Country.or.region, Score, pred_score) %>%
  head()
```

```
##    Overall.rank Country.or.region Score pred_score
## 1             1           Finland 7.769   6.958656
## 7             7            Sweden 7.343   6.986618
## 10           10           Austria 7.246   6.793396
## 12           12        Costa Rica 7.167   6.280958
## 17           17           Germany 6.985   6.720572
## 18           18           Belgium 6.923   6.587322
```

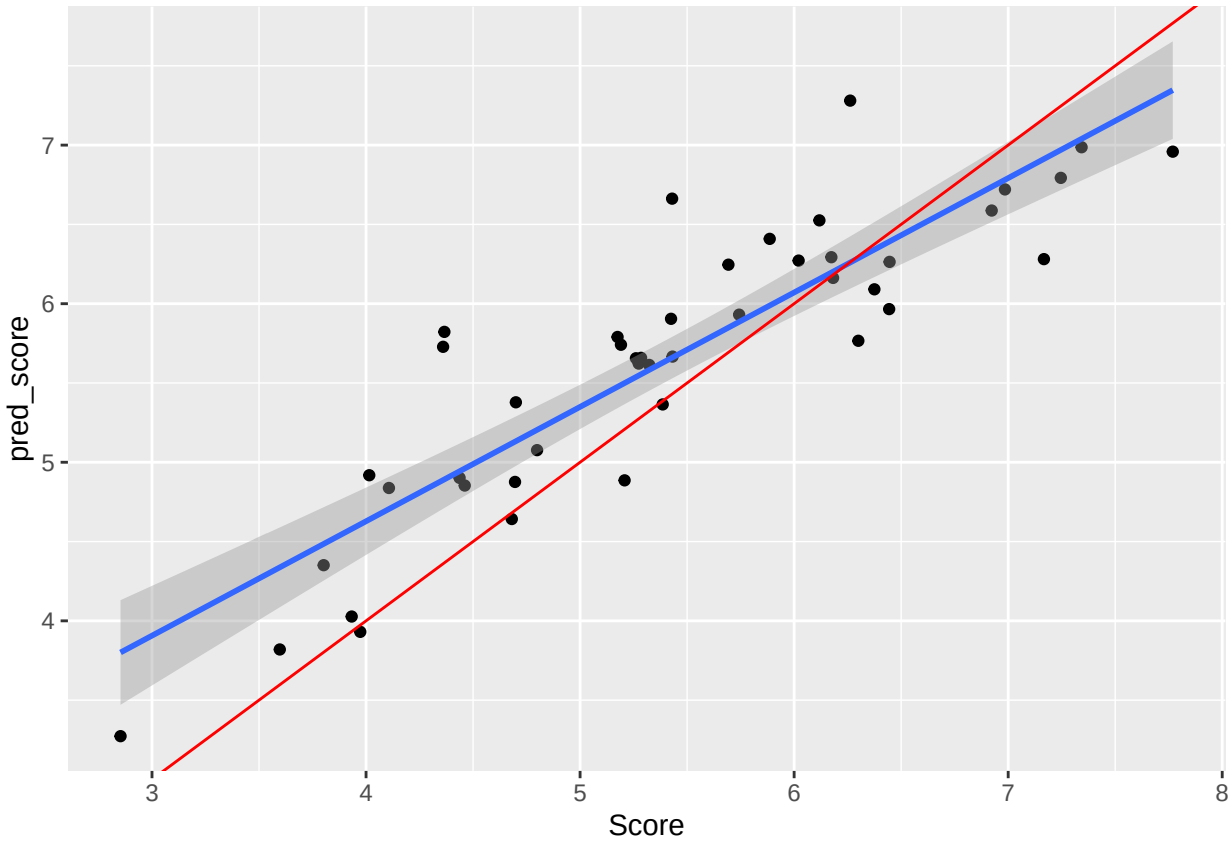
```
# show bottom five observations
results %>%
  select(Overall.rank, Country.or.region, Score, pred_score) %>%
  tail()
```

```
##     Overall.rank Country.or.region Score pred_score
## 140          140             India 4.015   4.918196
## 142          142           Comoros 3.973   3.930634
## 143          143        Madagascar 3.933   4.027239
## 144          144           Lesotho 3.802   4.350990
## 147          147             Haiti 3.597   3.819518
## 156          156       South Sudan 2.853   3.272902
```

The top five and bottom five observations are shown above compared with the actual score. The `results` data frame is plotted below with a line of best fit in blue and a reference line in red at `y = x`. If the model was to work perfectly, the line of best fit would follow the reference line because each predicted score would be equal to the score.

```
# plot predicted scores vs actual scores
# also plot y = x line
ggplot(data = results, aes(Score, pred_score)) +
  geom_point(color='black') +
  geom_smooth(method = "lm", se = TRUE) +
  geom_abline(color='red')
```

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



Even though results have been shown, it is worth mentioning the use of RMSE instead of other success indicators. RMSE suggests how close (or far) your predicted values are from the actual data you are attempting to model. The use of a success measure for this model, and others in the methods section, is to understand the accuracy and precision of the model's predictions. For this reason, RMSE is used as a success metric over alternatives. The RMSE of this model is 0.5718812. The coefficients of the fitted model are shown below for completeness:

```
# save model 2 RMSE
mod2_rmse_gen <- RMSE(results$Score, results$pred_score)
# save coefficients to use in equation
c <- coefficients(fit)
c[] <- lapply(c, round, 3)
# print coefficients of fitted model
fit$coefficients
```

```
##                   (Intercept)                GDP.per.capita
##                     1.8483663                     0.8230753
##                Social.support       Healthy.life.expectancy
##                     1.0155070                     1.0725756
## Freedom.to.make.life.choices                    Generosity
##                     1.5864416                     0.9939111
##     Perceptions.of.corruption
##                     0.6122442
```

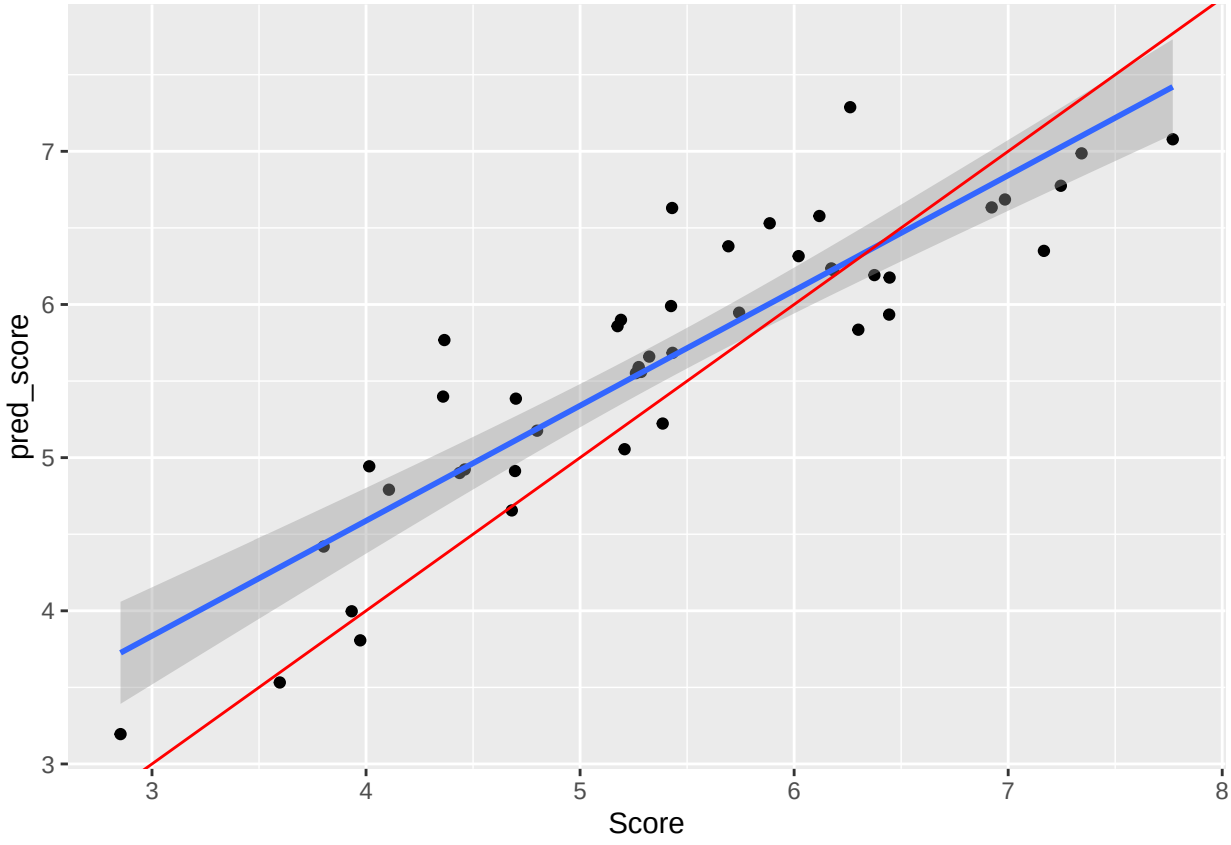
The following equation is the final model equation. Using these coefficients and the following notation predicted score = $\hat{y}$, GDP per capita score = x_{GDP}, Social Support score = x_{SS}, Life Expectancy score = x_{HEA}, Freedom score = x_{FRE}, Generosity score = x_{GEN}, Truth score = x_{TRU}.

$$\hat{y} = 1.848 + 0.823x_{GDP} + 1.016x_{SS} + 1.073x_{HEA} + 1.586x_{FRE} + 0.994x_{GEN} + 0.612x_{TRU} \quad (1)$$

Removing `Generosity` In an attempt to improve the model, we can remove the generosity component because early evaluations pointed to it being the least correlated factor. The previously partitioned data (`p = 0.70`) is used in this model as well.

```
# fit model without generosity
fit <- glm(Score ~ GDP.per.capita +
             Social.support +
             Healthy.life.expectancy +
             Freedom.to.make.life.choices +
             Perceptions.of.corruption,
           data = train)
# add predicted scores to a 'results' data frame
results <- test %>%
  mutate(pred_score = predict.glm(fit, newdata=test))
# plot predicted scores vs actual scores
ggplot(data = results, aes(Score, pred_score)) +
  geom_point(color='black') +
  geom_smooth(method = "lm", se = TRUE) +
  geom_abline(color='red')
```

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



This model yields a RMSE of 0.5588842. The model coefficients are shown below along with the model equation following the same format as before.

```
# save rmse and coefficients
mod2_rmse_nogen <- RMSE(results$Score, results$pred_score)
c <- coefficients(fit)
c[] <- lapply(c, round, 3)
# print coefficients of fitted model
fit$coefficients
```

```
##                   (Intercept)                GDP.per.capita
##                     1.9755643                     0.7931440
##                Social.support       Healthy.life.expectancy
##                     0.9743670                     1.0938625
## Freedom.to.make.life.choices     Perceptions.of.corruption
##                     1.8273152                     0.8300638
```

$$\hat{y} = 1.976 + 0.793x_{GDP} + 0.974x_{SS} + 1.094x_{HEA} + 1.827x_{FRE} + 0.83x_{TRU} \tag{2}$$

Model 3: The 2018/19 GLM Model

In an ideal world, data sets would be infintely large and models can be fitted with as much precision as your CPU can handle. The next model uses data from the 2018 and 2019 surveys. While not exactly an infinite data set, doubling the size of our train and test sets do allow for more training and testing. Combining data from old surveys is appropriate in this scenario because these two years follow the same grading scheme, specifically they both do not have the dystopian residuals mentioned above. The first step is to reproduce new train and test sets, then run a model including all factors.

```
# add 2018 data set and keep only used columns
data18 <- read.csv("2018.csv")
data18$Overall.rank <- NULL
data18$Country.or.region <- NULL
# remove unused columns in 2019 data frame for merging
data$Overall.rank <- NULL
data$Country.or.region <- NULL
# turn corruption column from factor to numeric, turn NAs to 0
data18$Perceptions.of.corruption <- as.numeric(as.character(data18$Perceptions.of.corruption))
```

```
## Warning: NAs introduced by coercion
```

```
data18[is.na(data18)] <- 0
# build full data set of both 2019, 2018 data
full_data <- rbind(data, data18)
# show full data set
glimpse(full_data)
```

```
## Rows: 312
## Columns: 7
## $ Score                          <dbl> 7.769, 7.600, 7.554, 7.494, 7.488, 7.480,~
## $ GDP.per.capita                 <dbl> 1.340, 1.383, 1.488, 1.380, 1.396, 1.452,~
```

```
## $ Social.support               <dbl> 1.587, 1.573, 1.582, 1.624, 1.522, 1.526,~
## $ Healthy.life.expectancy      <dbl> 0.986, 0.996, 1.028, 1.026, 0.999, 1.052,~
## $ Freedom.to.make.life.choices <dbl> 0.596, 0.592, 0.603, 0.591, 0.557, 0.572,~
## $ Generosity                   <dbl> 0.153, 0.252, 0.271, 0.354, 0.322, 0.263,~
## $ Perceptions.of.corruption    <dbl> 0.393, 0.410, 0.341, 0.118, 0.298, 0.343,~
```

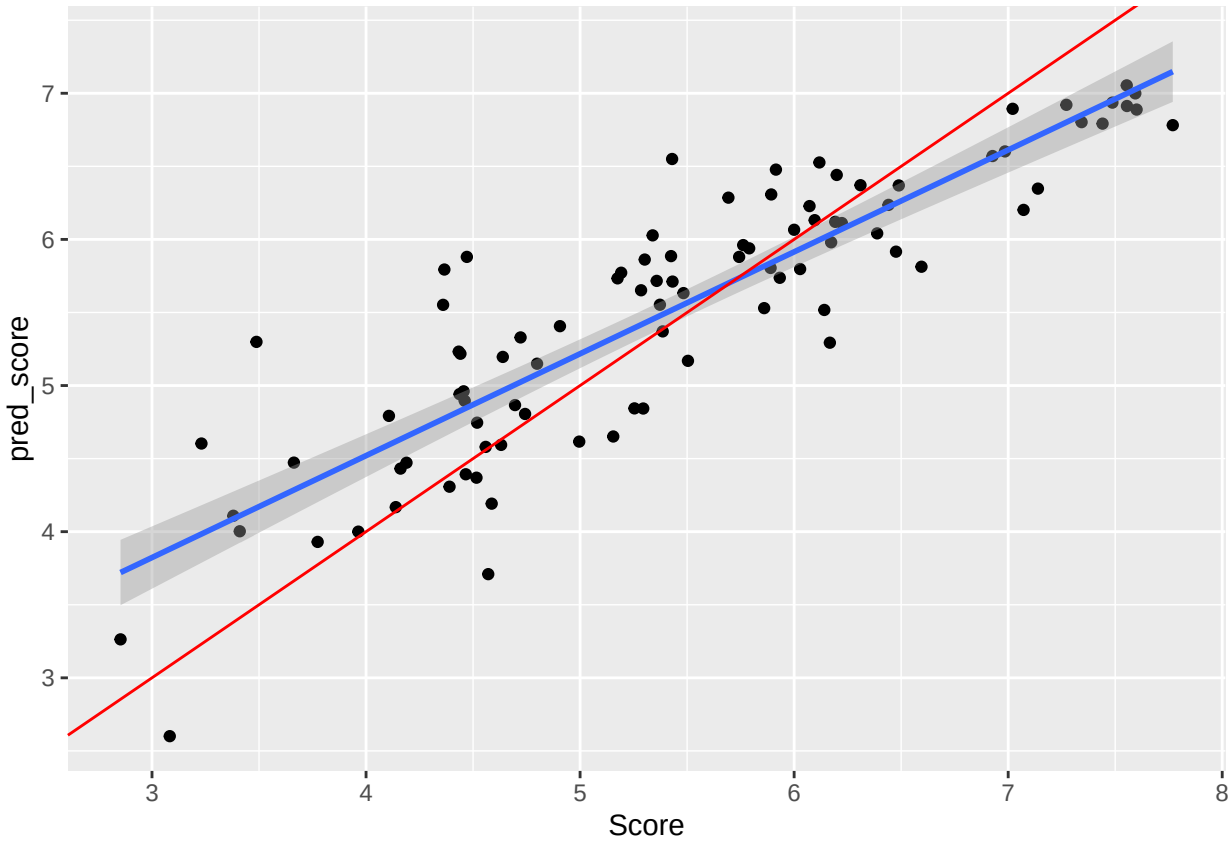
Columns with unused data, such as country name and rank, were removed before merging. NAs are introduced by coercion when converting the corruption column of the 2018 data set from factor to numeric. These are changed to zeros and represent unfound data. A glimpse of the 2018/2019 data set is outputted above.

```
# set seed to keep partitioning consistent
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
# partition data
train_index <- createDataPartition(full_data$Score, times=1, p=0.70, list=FALSE)
train <- full_data[train_index,]
test <- full_data[-train_index,]
# fit a model (including generosity)
fit <- glm(Score ~ GDP.per.capita +
             Social.support +
             Healthy.life.expectancy +
             Freedom.to.make.life.choices +
             Generosity +
             Perceptions.of.corruption,
           data = train)
# add predicted scores to our 'results' data frame
results <- test %>%
  mutate(pred_score = predict.glm(fit, newdata=test))
# plot predicted scores vs actual scores
ggplot(data = results, aes(Score, pred_score)) +
  geom_point(color='black') +
  geom_smooth(method = "lm", se = TRUE) +
  geom_abline(color='red')
```

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



We see a better fitting model here, especially around the median, where the confidence bands are small. This model yields a RMSE of 0.5747943. The model coefficients are shown below along with the model equation following the same notation as before.

```
# save rmse and coefficients
mod3_rmse_gen <- RMSE(results$Score, results$pred_score)
c <- coefficients(fit)
c[] <- lapply(c, round, 3)
# print coefficients of fitted model
fit$coefficients
```

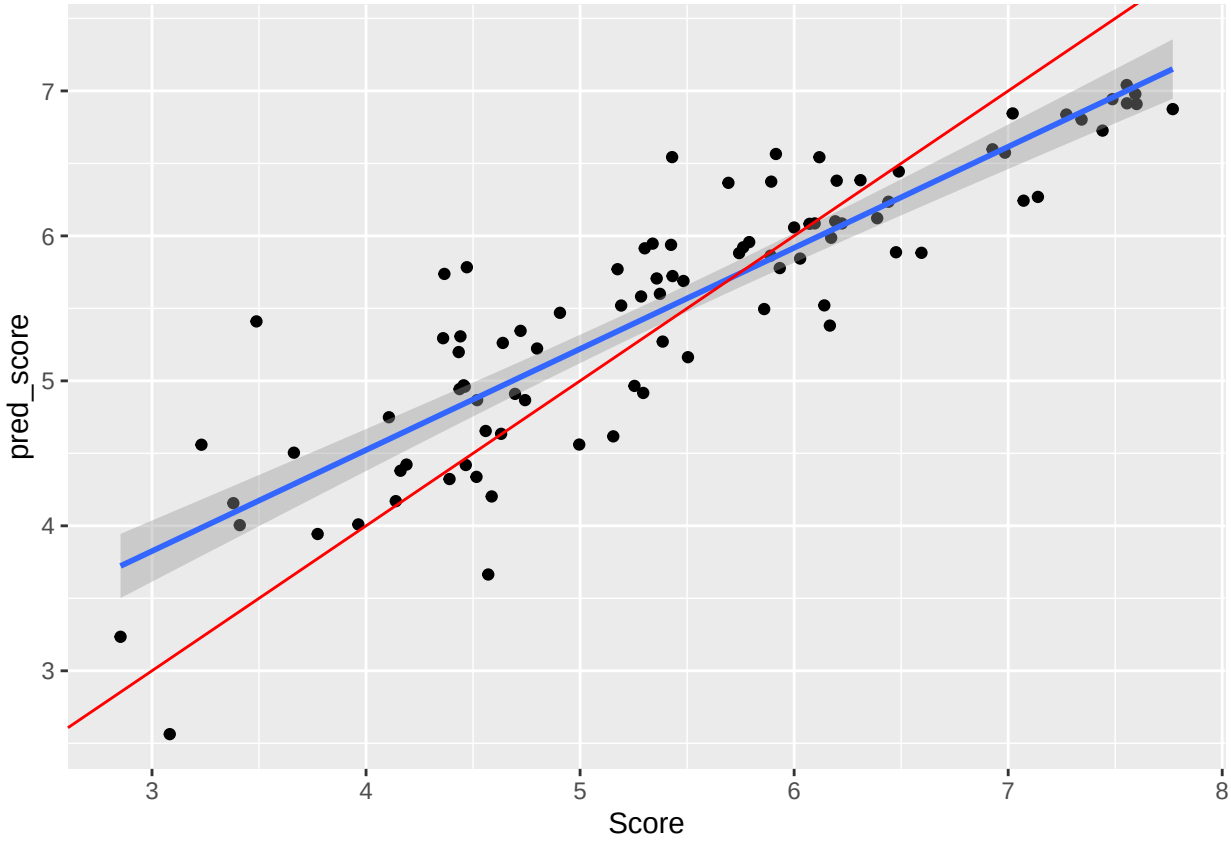
```
##                   (Intercept)                GDP.per.capita
##                    1.95281607                    0.95571540
##                Social.support       Healthy.life.expectancy
##                    0.97041194                    0.96817249
## Freedom.to.make.life.choices                    Generosity
##                    1.52136362                    0.75035656
##     Perceptions.of.corruption
##                    0.08183693
```

$$\hat{y} = 1.953 + 0.956x_{GDP} + 0.97x_{SS} + 0.968x_{HEA} + 1.521x_{FRE} + 0.75x_{GEN} + 0.082x_{TRU} \tag{3}$$

Removing `Generosity` For curiosity's sake, the previous model may be improved by removing the somewhat uncorrelated generosity factor seen in earlier sections. This is completed with the 2018-2019 data set, `full_data`.

```r
# fit a model (not including generosity)
fit <- glm(Score ~ GDP.per.capita +
             Social.support +
             Healthy.life.expectancy +
             Freedom.to.make.life.choices +
             Perceptions.of.corruption,
           data = train)
# add predicted scores to our 'results' data frame
results <- test %>%
  mutate(pred_score = predict.glm(fit, newdata=test))
# plot predicted scores vs actual scores
ggplot(data = results, aes(Score, pred_score)) +
  geom_point(color='black') +
  geom_smooth(method = "lm", se = TRUE) +
  geom_abline(color='red')
```

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



We see a better fitting model here, especially around the median, where the confidence bands are small. This model yields a RMSE of 0.5716972. The model coefficients are shown below along with the model equation following the same notation as before.

```r
# save rmse and coefficients
mod3_rmse_nogen <- RMSE(results$Score, results$pred_score)
c <- coefficients(fit)
```

```
c[] <- lapply(c, round, 3)
# print coefficients of fitted model
fit$coefficients
```

```
##                   (Intercept)                GDP.per.capita
##                     2.0640555                     0.9153533
##                Social.support       Healthy.life.expectancy
##                     0.9602797                     0.9904659
## Freedom.to.make.life.choices     Perceptions.of.corruption
##                     1.5991209                     0.3312910
```

$$\hat{y} = 2.064 + 0.915x_{GDP} + 0.96x_{SS} + 0.99x_{HEA} + 1.599x_{FRE} + 0.331x_{TRU} \tag{4}$$

Results

The results of our five models are shown in the table below. It is clearly shown that the best model in terms of RMSE is the sum of factors model. Even though the data seemed incomplete with dystopian residuals, taking the sum of the factor and the standard dystopian scores yield the closest score predictions.

```
# compiles the list of RMSEs
rmse_list <- list(mod1_rmse,
                  mod2_rmse_gen,
                  mod2_rmse_nogen,
                  mod3_rmse_gen,
                  mod3_rmse_nogen)
rmse_list[] <- lapply(rmse_list, round, 3)
```

Method	RMSE
Sum of Factors Model	0.528
GLM Model 2019	0.572
GLM Model 2018/2019	0.575
GLM Model - No Generosity 2019	0.559
GLM Model - No Generosity 2018/2019	0.572

For completeness, it may be useful to check the summation model with the full 2018-2019 data set. This is shown below and reported in the final table.

```
# find our predicted score and calculate the corresponding RMSE
sum_model <- full_data %>% mutate(pred_score = GDP.per.capita +
                                    Social.support +
                                    Healthy.life.expectancy +
                                    Freedom.to.make.life.choices +
                                    Generosity +
                                    Perceptions.of.corruption +
                                    1.85,
                                  RMSE = RMSE(Score, pred_score))
# save RMSE for the first model
mod1_rmse_full <- RMSE(sum_model$Score, sum_model$pred_score)
```

```
# recompiles the list of RMSEs
rmse_list <- list(mod1_rmse,
                  mod2_rmse_gen,
                  mod2_rmse_nogen,
                  mod3_rmse_gen,
                  mod3_rmse_nogen,
                  mod1_rmse_full)
rmse_list[] <- lapply(rmse_list, round, 3)
```

It is clearly seen that the best model is the summation model. Interestingly the GLM models do not improve when the data set size doubles. This may be due to overfitting in smaller data size that does not occur in the larger data.

Method	RMSE
Sum of Factors Model	0.528
Sum of Factors Model 2018/2019	0.526
GLM Model 2019	0.572
GLM Model 2018/2019	0.575
GLM Model - No Generosity 2019	0.559
GLM Model - No Generosity 2018/2019	0.572

Conclusion

Machine learning algorithms have become a paramount tool in computer science and analysis. This report explores this use with regression analysis on a dataset of national happiness. Results are found that are rather unexpected: a glaring error in the data did not impede the simplest model, increasing training data set size did not decrease the error, and reducing the effectiveness of uncorrelated variables did not decrease the error.

This report felt limited in the amount of data being trained and tested. If more consistent data was available, it is my belief that the summation model would out perform the GLM models. The omitted residuals left a hole in the data that may have provided more insight into the validity of the summation model. The discussion boards were helpful in this respect by providing insight on the purpose of the data. Further work on a project such as this would include calculating and using dystopian residuals for data sets where they were lost, combining previous years reports, and comparing more adaptive models on the large data sets.