STT 461 Proj

Derien Weatherspoon

2023-04-18

Project Overview

Explanation of the variables in the data set:

- latitude, longitude: GPS Coordinates for a given country
- country: Countries from different parts of the world
- country_code: Abbreviations for countries
- year: year in which happiness data was collected
- value: GINI index value per country per year, ranging from 0 to 1
- score: Happiness score, which is explained by the following: GDP per capita, Healthy Life Expectancy, Social support, Freedom to make life choices, Generosity, Corruption Perception, and Residual error.
- gdp: Gross Domestic Product
- support: Social support
- life_expectancy: Life expectancy
- freedom: How free the citizens of a given country are to make life choices on their own
- trust: How much trust the people have in their government
- generosity: Generosity of the citizens/government

Loading Packages

```
library(dplyr)
library(sqldf)
library(tidyr)
library(MASS)
library(stringr)
library(ggplot2)
library(ggcorrplot)
library(glmnet)
library(tree)
library(rpart)
library(rpart.plot)
library(leaps)
library(randomForest)
```

Import Data

```r
gini <- read.csv("gini1.csv")
coords <- read.csv('coords.csv')
```

Feature Engineering

```r
hap15 <- read.csv('2015.csv')
hap16 <- read.csv('2016.csv')
hap17 <- read.csv('2017.csv')
hap18 <- read.csv('2018.csv')
hap19 <- read.csv('2019.csv')
hap20 <- read.csv('2020.csv')
hap21 <- read.csv('2021.csv')

hap15$year = 2015
hap16$year = 2016
hap17$year = 2017
hap18$year = 2018
hap19$year = 2019
hap20$year = 2020
hap21$year = 2021

colnames(hap15) <- c('country', 'region', 'rank', 'score', 'se', 'gdp', 'support',
                     'life_expectancy', 'freedom', 'trust', 'generosity', 'corruption', 'year')
hap15 <- hap15[,c(1,4,6,7,8,9,10,11,13)]

hap16 <- hap16[,c(1,4,7,8,9,10,11,12,14)]
colnames(hap16) <- c('country', 'score', 'gdp', 'support', 'life_expectancy',
                     'freedom', 'trust', 'generosity', 'year')

hap17 <- hap17[,c(1,3,6,7,8,9,10,11,13)]
colnames(hap17) <- c('country', 'score', 'gdp', 'support', 'life_expectancy',
                     'freedom', 'trust', 'generosity', 'year')

hap18 <- hap18[,c(2,3,4,5,6,7,8,9,10)]
colnames(hap18) <- c('country', 'score', 'gdp', 'support', 'life_expectancy',
                     'freedom', 'generosity', 'trust', 'year')
hap18 <- hap18 %>% relocate('trust', .before='generosity')

hap19 <- hap19[,c(2,3,4,5,6,7,8,9,10)]
colnames(hap19) <- c('country', 'score', 'gdp', 'support', 'life_expectancy',
                     'freedom', 'generosity', 'trust', 'year')
hap19 <- hap19%>%relocate('trust', .before='generosity')

hap20 <- hap20[,c(1,3,7,8,10,11,12,16,21)]
colnames(hap20) <- c('country','score','gdp','support','freedom', 'generosity',
                     'trust', 'life_expectancy', 'year')
hap20 <- hap20%>%relocate('life_expectancy', .before='freedom')

hap21 <- hap21[,c(1,3,7,8,10,11,12,16,21)]
colnames(hap21) <- c('country', 'score', 'gdp', 'support', 'freedom',
```

```
                     'generosity', 'trust', 'life_expectancy', 'year')
hap21 <- hap21%>%relocate('life_expectancy', .before='freedom')
```

Bind all Happy data

```
happy <- rbind(hap15,hap16,hap17,hap18,hap19,hap20,hap21)
```

Match the names for GINI and Happy data

Join the geographical data and GINI data

```
geo_gini <- sqldf('SELECT * from coords INNER JOIN gini ON coords.country = gini.country_name')
geo_gini <- geo_gini[, c(2,3,4,5,7,8)]
```

Join geo_gini with happy data set

```
data <- sqldf('SELECT * from geo_gini INNER JOIN happy ON
              geo_gini.country = happy.country AND geo_gini.year = happy.year ORDER BY year')
df <- data[,c(1,2,3,4,5,6,8,9,10,11,12,13,14)]
```

Write new data as a new csv

```
#write.csv(df, file='gini_geo_happy.csv', row.names = F)
```

Our target variable here is **happiness score**. Other features used include GINI index, health, trust, freedom, and a few others.

The target field for this question will be the happiness index.

Read in CSV and split the data

```
set.seed(1)
df <- read.csv("gini_geo_happy.csv")
df$trust <- as.double(df$trust) # change to a double first
df$trust[is.na(df$trust)] <- mean(df$trust[!is.na(df$trust)])
row_nums <- 1:nrow(df)
train_split <- sample(row_nums, 0.7*length(row_nums)) # 70-30 split
train_data <- df[(train_split),]
test_data <- df[-train_split,]
train_data_no_country <- train_data[,!names(train_data) %in% c("country", "country_code")]
numerics <- df[,c("latitude", "longitude", "value", "score", "gdp", "support",
                  "life_expectancy", "freedom", "trust", "generosity")] # df with numeric features only
```

Description of what you do with the data before modeling.

Before modeling the data, we first split the data into a 70-30 train-test split as seen above, then check for NAs, do a little bit of EDA to check any important factors, such as correlation.

EDA

Check NAs

```
train_data <- na.omit(train_data)
train_data_no_country <- na.omit(train_data_no_country)
sum(is.na(train_data))
```

```
## [1] 0
```

```
sum(is.na(test_data))
```

```
## [1] 0
```

```
sum(is.na(df))
```

```
## [1] 0
```

```
sum(is.na(train_data_no_country))
```

```
## [1] 0
```

There are no null values within the training or test data. So there is no need to impute any missing values with the median for those data sets. There is one missing value for the main df when I changed trust into a numeric variable, so I will remove the NA, since it is only one.

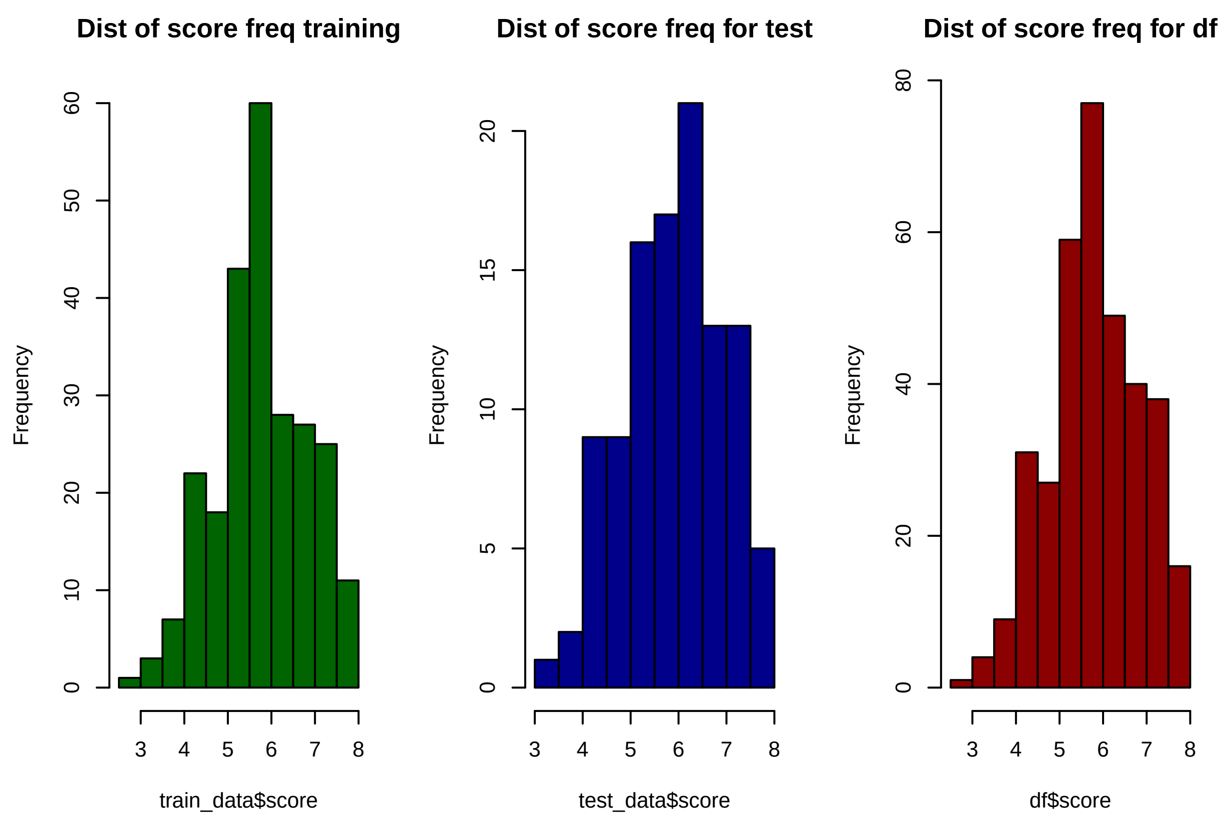
```
scaled.data <- model.matrix(score ~., data = train_data_no_country)[,-1]
head(scaled.data)
```

```
##       latitude  longitude year value      gdp  support life_expectancy
## 324  60.128161  18.643501 2019  29.3 1.387000 1.487000       1.0090000
## 167  46.227638   2.213749 2017  31.6 1.430923 1.387777       0.8444659
## 129 -23.442503 -58.443832 2016  47.9 0.893730 1.111110       0.5829500
## 299  -1.831239 -78.183406 2019  45.7 0.912000 1.312000       0.8680000
## 270  44.016521  21.005859 2018  35.0 0.975000 1.369000       0.6850000
## 187  42.708678  19.374390 2017  36.9 1.121129 1.238376       0.6674647
##       freedom     trust generosity
## 324 0.5740000 0.3730000 0.26700000
## 167 0.4702221 0.1297623 0.17250243
## 129 0.4623500 0.0739600 0.25296000
## 299 0.4980000 0.0870000 0.12600000
## 270 0.2880000 0.0430000 0.13400000
## 187 0.1949891 0.1979110 0.08817419
```

```r
sco <- train_data$score # make the response variable
```

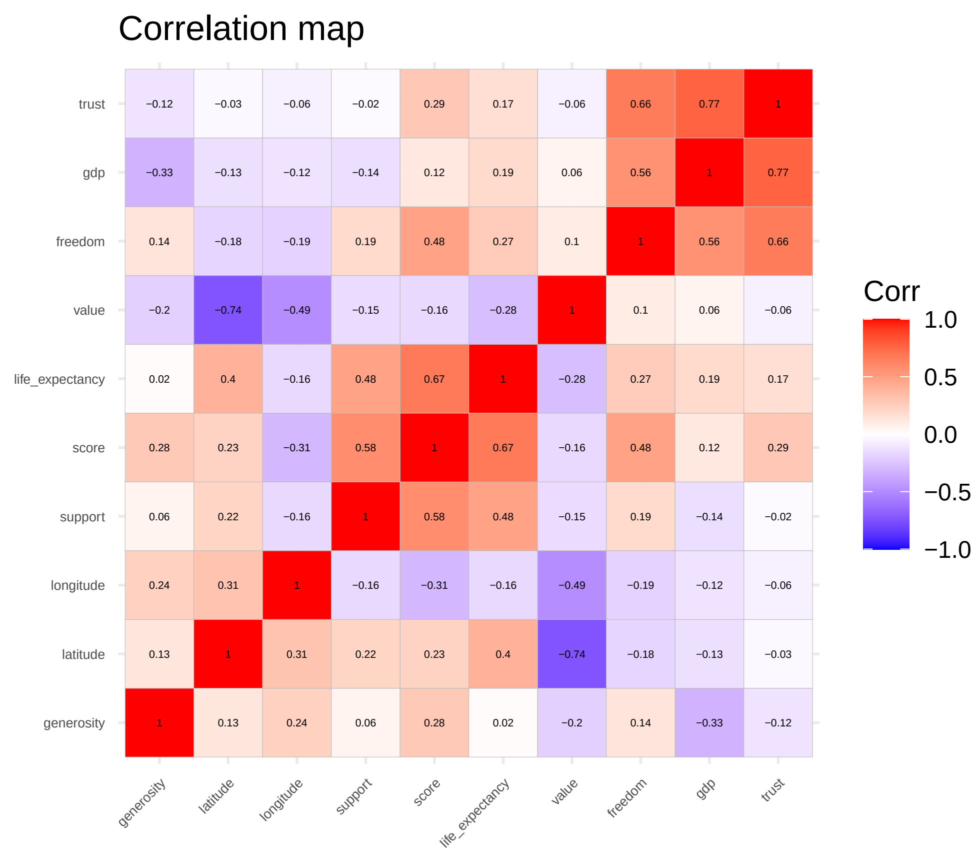
Histograms for response; score

```r
par(mfrow = c(1,3))
hist(x = train_data$score, col = "darkgreen", freq = T, main = "Dist of score freq training")
hist(x = test_data$score, col = "darkblue", freq = T, main = "Dist of score freq for test")
hist(x = df$score, col = "darkred", freq = T, main = "Dist of score freq for df")
```



Correlation matrix

```r
ggcorrplot(cor(numerics), lab_size = 1.5, tl.cex = 5, lab = T, title = "Correlation map",  hc.order = T
```



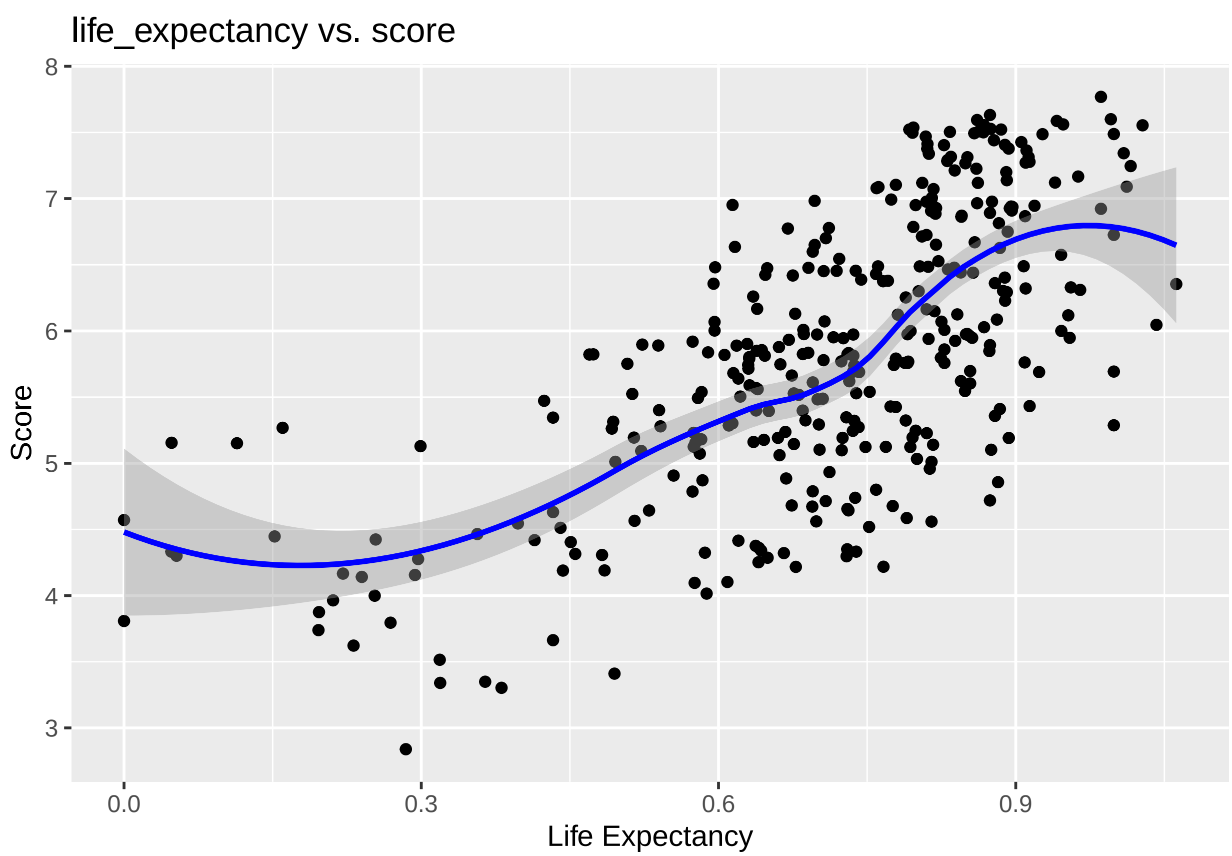

```r
round(cor(numerics),
  digits = 2 # rounded to 2 decimals
)
```

```
##                 latitude longitude value score   gdp support life_expectancy
## latitude            1.00      0.31 -0.74  0.23 -0.13    0.22            0.40
## longitude           0.31      1.00 -0.49 -0.31 -0.12   -0.16           -0.16
## value              -0.74     -0.49  1.00 -0.16  0.06   -0.15           -0.28
## score               0.23     -0.31 -0.16  1.00  0.12    0.58            0.67
## gdp                -0.13     -0.12  0.06  0.12  1.00   -0.14            0.19
## support             0.22     -0.16 -0.15  0.58 -0.14    1.00            0.48
## life_expectancy     0.40     -0.16 -0.28  0.67  0.19    0.48            1.00
## freedom            -0.18     -0.19  0.10  0.48  0.56    0.19            0.27
## trust              -0.03     -0.06 -0.06  0.29  0.77   -0.02            0.17
## generosity          0.13      0.24 -0.20  0.28 -0.33    0.06            0.02
##                 freedom trust generosity
## latitude          -0.18 -0.03       0.13
## longitude         -0.19 -0.06       0.24
## value              0.10 -0.06      -0.20
## score              0.48  0.29       0.28
## gdp                0.56  0.77      -0.33
## support            0.19 -0.02       0.06
## life_expectancy    0.27  0.17       0.02
## freedom            1.00  0.66       0.14
## trust              0.66  1.00      -0.12
## generosity         0.14 -0.12       1.00
```

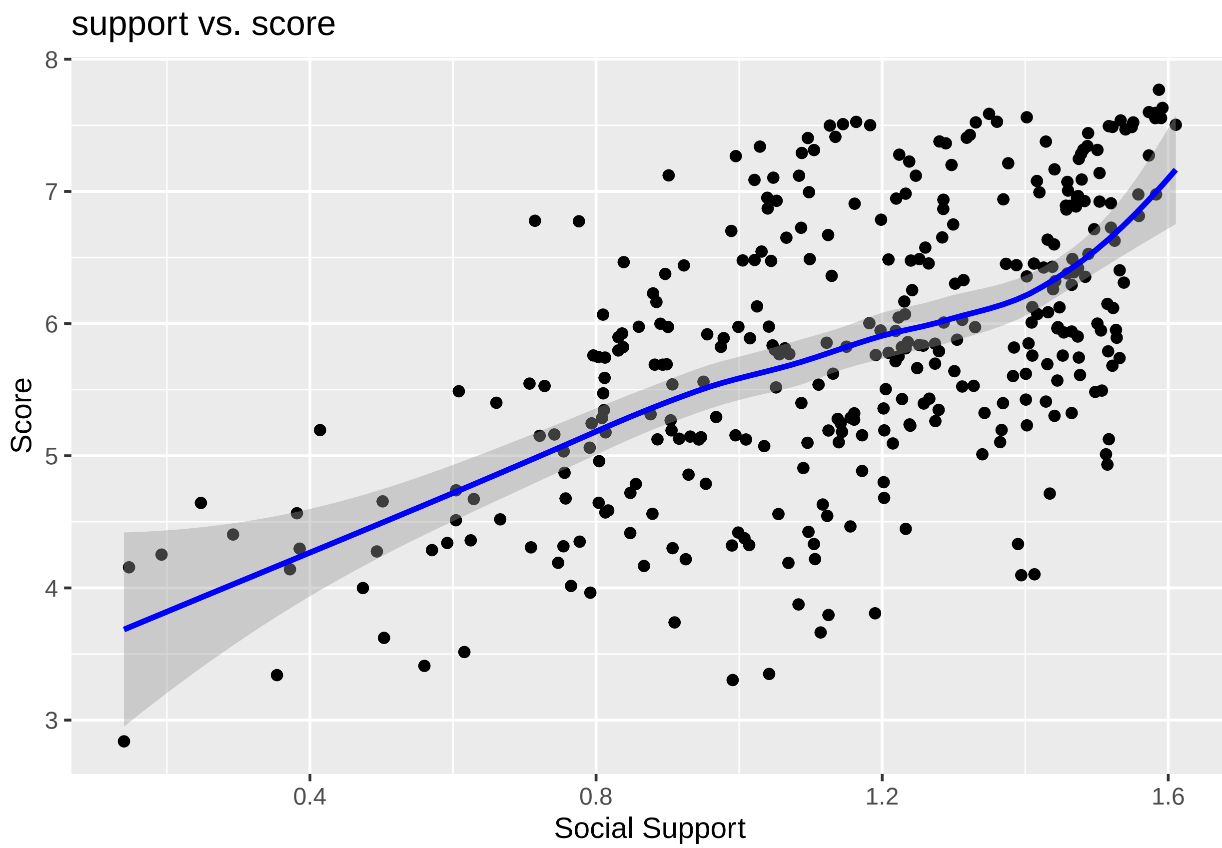
life_expectancy is the variable most correlated with the response variable here, score. Following life_expectancy in highest correlation are support, freedom, longitude.

Plotting the correlated variables vs response

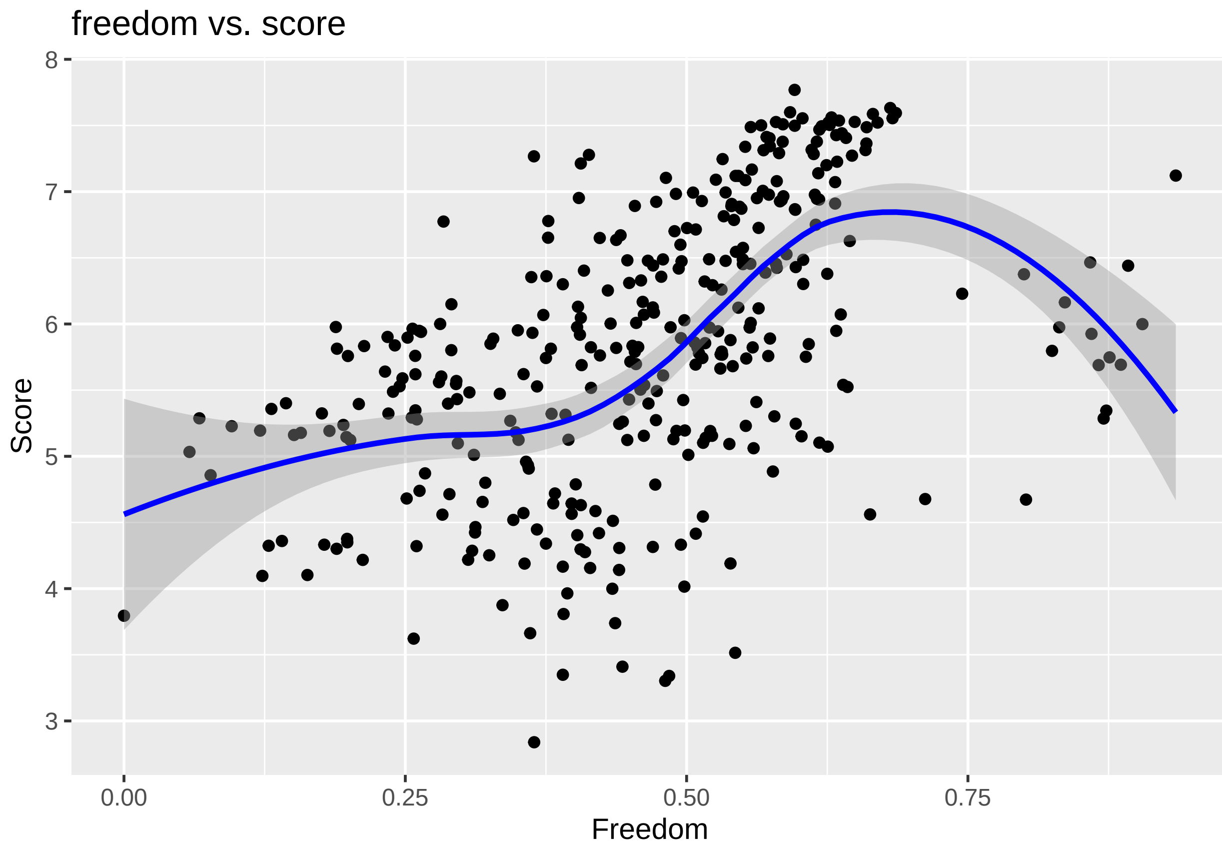
```
ggplot(data = df[df$score > 0,], aes(x = life_expectancy, y = score)) +
  geom_point() +
  geom_smooth(method = NULL, se = T, colour = "blue", linetype = "solid") +
  labs(x = "Life Expectancy",
       y = "Score",
       title = "life_expectancy vs. score"
       )
```



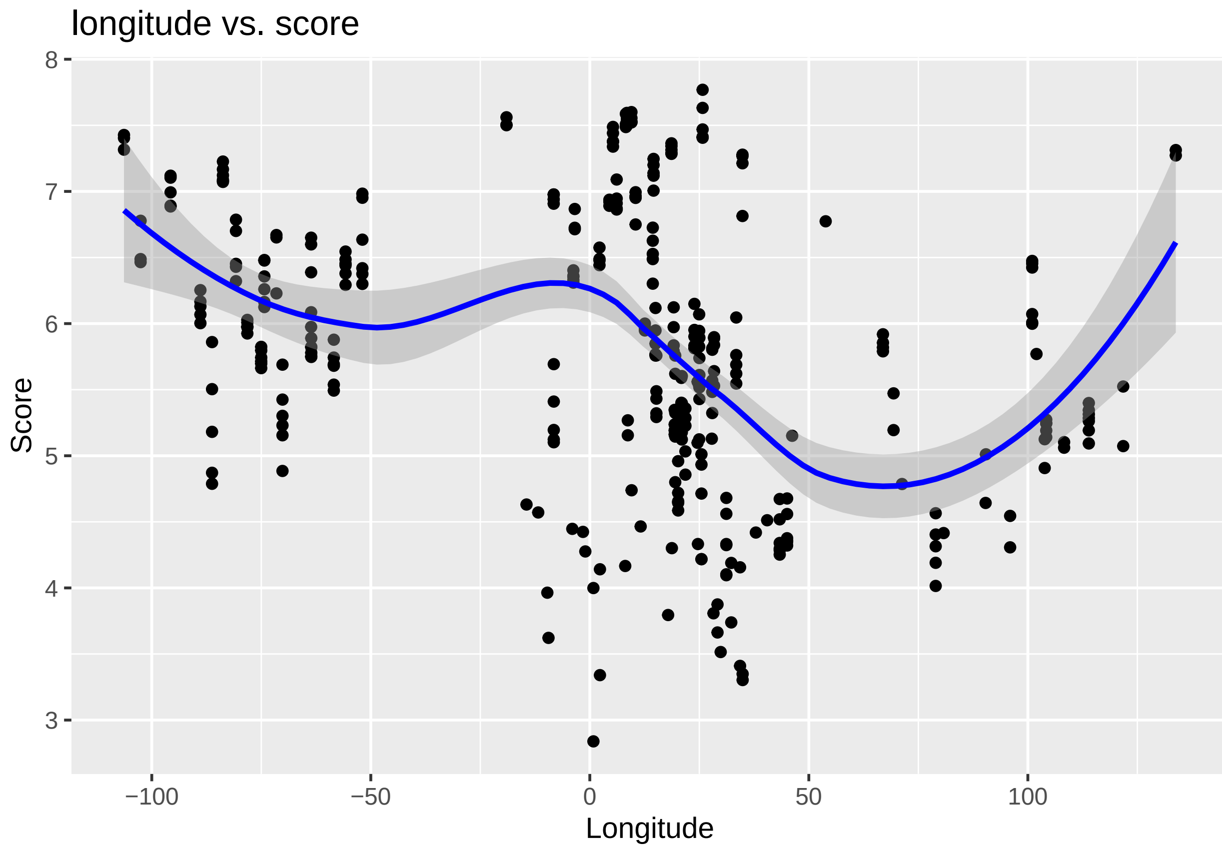

```
ggplot(data = df[df$score > 0,], aes(x = support, y = score)) +
  geom_point() +
  geom_smooth(method = NULL, se = T, colour = "blue", linetype = "solid") +
  labs(x = "Social Support",
       y = "Score",
       title = "support vs. score"
       )
```




```
ggplot(data = df[df$score > 0,], aes(x = freedom, y = score)) +
  geom_point() +
  geom_smooth(method = NULL, se = T, colour = "blue", linetype = "solid") +
  labs(x = "Freedom",
       y = "Score",
       title = "freedom vs. score"
       )
```




```r
ggplot(data = df[df$score > 0,], aes(x = longitude, y = score)) +
  geom_point() +
  geom_smooth(method = NULL, se = T, colour = "blue", linetype = "solid") +
  labs(x = "Longitude",
       y = "Score",
       title = "longitude vs. score"
       )
```



Description of modeling to be done

Modeling to be done will include a mix of regression models looking for the best accuracy scores, including adjusted R^2 values, AIC/BIC where applicable, and prediction scores.

Build a regression model to predict a country's happiness index based on all features, soon down to only significant features.

Linear Regression Modeling

```
lm.fit <- lm(score ~ ., data = train_data[,!names(train_data) %in% c("country", "country_code")])
summary(lm.fit)
```

```
##
## Call:
## lm(formula = score ~ ., data = train_data[, !names(train_data) %in%
##     c("country", "country_code")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.55923 -0.30147  0.03209  0.30272  1.84605
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      2.714e+02  5.869e+01   4.625 6.20e-06 ***
## latitude        -5.940e-04  2.039e-03  -0.291 0.771058    
## longitude       -4.763e-03  7.731e-04  -6.161 3.14e-09 ***
## year            -1.333e-01  2.910e-02  -4.582 7.51e-06 ***
## value           -1.337e-02  7.594e-03  -1.761 0.079490 .  
## gdp             -7.007e-03  3.497e-02  -0.200 0.841379    
## support          1.027e+00  1.427e-01   7.198 8.24e-12 ***
## life_expectancy  2.248e+00  2.340e-01   9.609  < 2e-16 ***
## freedom          1.235e+00  3.235e-01   3.816 0.000174 ***
## trust            1.087e+00  3.175e-01   3.426 0.000724 ***
## generosity       1.809e+00  3.156e-01   5.732 3.05e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.5215 on 234 degrees of freedom
## Multiple R-squared:  0.7666, Adjusted R-squared:  0.7567 
## F-statistic: 76.87 on 10 and 234 DF,  p-value: < 2.2e-16
```

```r
lm.fit2 <- lm(score ~ . - latitude - value - gdp, data = train_data[,!names(train_data) %in% c("country
summary(lm.fit2)
```

```
## 
## Call:
## lm(formula = score ~ . - latitude - value - gdp, data = train_data[, 
##     !names(train_data) %in% c("country", "country_code")])
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.59364 -0.30926  0.00863  0.32978  1.94048 
## 
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)      2.740e+02  5.382e+01   5.091 7.26e-07 ***
## longitude       -3.991e-03  6.614e-04  -6.034 6.11e-09 ***
## year            -1.349e-01  2.670e-02  -5.053 8.70e-07 ***
## support          1.067e+00  1.296e-01   8.233 1.23e-14 ***
## life_expectancy  2.379e+00  1.980e-01  12.014  < 2e-16 ***
## freedom          1.096e+00  2.980e-01   3.677 0.000292 ***
## trust            1.168e+00  2.520e-01   4.636 5.87e-06 ***
## generosity       1.901e+00  2.975e-01   6.390 8.69e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.5227 on 237 degrees of freedom
## Multiple R-squared:  0.7626, Adjusted R-squared:  0.7556 
## F-statistic: 108.8 on 7 and 237 DF,  p-value: < 2.2e-16
```

```r
lm.fit3 <- lm(log(score) ~ . -value - latitude - gdp, data = train_data[,!names(train_data) %in% c("cou
summary(lm.fit3) # with log transformation on response
```

```
## 
```

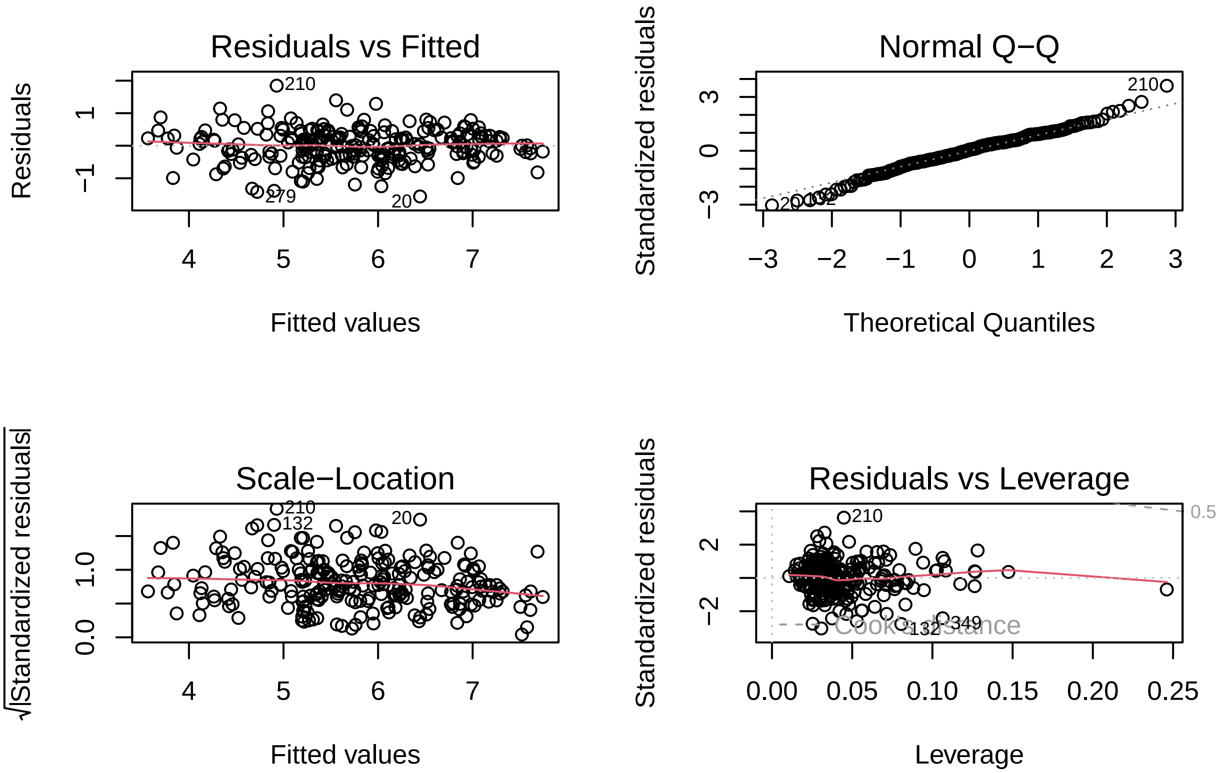
```
## Call:
## lm(formula = log(score) ~ . - value - latitude - gdp, data = train_data[,
##     !names(train_data) %in% c("country", "country_code")])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34091 -0.04625  0.00623  0.06375  0.35244
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     51.0873327 10.2725542   4.973 1.26e-06 ***
## longitude       -0.0006867  0.0001263  -5.439 1.33e-07 ***
## year            -0.0248207  0.0050962  -4.870 2.04e-06 ***
## support          0.1977237  0.0247397   7.992 5.80e-14 ***
## life_expectancy  0.4517290  0.0377909  11.953  < 2e-16 ***
## freedom          0.2122890  0.0568756   3.733 0.000237 ***
## trust            0.1648792  0.0480973   3.428 0.000717 ***
## generosity       0.2754832  0.0567921   4.851 2.23e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09977 on 237 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7329
## F-statistic: 96.67 on 7 and 237 DF,  p-value: < 2.2e-16
```

```
# Lower Adjusted R^2, not worth.
```

It looks like `lm.fit` works just fine, it has the best scores of the 4 linear models. And has latitude excluded, since it is insignificant.

Map residual data from model

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



There are some outliers and non-homoscedasticity points, but due to size of data, and what we've already removed, I will leave them in. ## May remove later ##

Stepwise Selection

```
forward.fit <- regsubsets(score ~ ., data = train_data_no_country, nvmax = ncol(train_data_no_country),
#summary(forward.fit)
forward.summary <- summary(forward.fit)
forward.summary$rsq # [9] has the best rsq value
```

```
##  [1] 0.4685037 0.5755881 0.6314211 0.6934282 0.7117013 0.7410847 0.7626128
##  [8] 0.7665175 0.7665970 0.7666370
```

```
forward.summary
```

```
## Subset selection object
## Call: regsubsets.formula(score ~ ., data = train_data_no_country, nvmax = ncol(train_data_no_country)
##     method = "forward")
## 10 Variables  (and intercept)
##                    Forced in Forced out
## latitude               FALSE      FALSE
## longitude              FALSE      FALSE
## year                   FALSE      FALSE
```

```
## value               FALSE      FALSE
## gdp                 FALSE      FALSE
## support             FALSE      FALSE
## life_expectancy     FALSE      FALSE
## freedom             FALSE      FALSE
## trust               FALSE      FALSE
## generosity          FALSE      FALSE
## 1 subsets of each size up to 10
## Selection Algorithm: forward
##           latitude longitude year value gdp support life_expectancy freedom
## 1  ( 1 )  " "      " "       " "  " "   " " " "     "*"             " "
## 2  ( 1 )  " "      " "       " "  " "   " " " "     "*"             "*"
## 3  ( 1 )  " "      " "       " "  " "   " " "*"     "*"             "*"
## 4  ( 1 )  " "      " "       "*"  " "   " " "*"     "*"             "*"
## 5  ( 1 )  " "      "*"       "*"  " "   " " "*"     "*"             "*"
## 6  ( 1 )  " "      "*"       "*"  " "   " " "*"     "*"             "*"
## 7  ( 1 )  " "      "*"       "*"  " "   " " "*"     "*"             "*"
## 8  ( 1 )  " "      "*"       "*"  "*"   " " "*"     "*"             "*"
## 9  ( 1 )  "*"      "*"       "*"  "*"   " " "*"     "*"             "*"
## 10  ( 1 ) "*"      "*"       "*"  "*"   "*" "*"     "*"             "*"
##           trust generosity
## 1  ( 1 )  " "   " "
## 2  ( 1 )  " "   " "
## 3  ( 1 )  " "   " "
## 4  ( 1 )  " "   " "
## 5  ( 1 )  " "   " "
## 6  ( 1 )  " "   "*"
## 7  ( 1 )  "*"   "*"
## 8  ( 1 )  "*"   "*"
## 9  ( 1 )  "*"   "*"
## 10  ( 1 ) "*"   "*"
```

```
# Rsq increased as more variables were added
```

```
paste(c('RSS:',which.min(forward.summary$rss)))
```

```
## [1] "RSS:" "10"
```

```
paste(c('Adjusted RSquared:',which.max(forward.summary$adjr2)))
```

```
## [1] "Adjusted RSquared:" "8"
```

```
paste(c('Cp:',which.min(forward.summary$cp)))
```

```
## [1] "Cp:" "8"
```

```
paste(c('BIC:',which.min(forward.summary$bic)))
```

```
## [1] "BIC:" "7"
```

The criterion I decide to go with is BIC, choosing the least number of variables

```r
coef(object = forward.fit, id = which.min(forward.summary$bic))
```

```
##     (Intercept)       longitude            year         support life_expectancy 
##    273.97486401     -0.00399115     -0.13490161      1.06705444      2.37858533 
##         freedom           trust      generosity 
##      1.09559702      1.16815506      1.90126539
```

```r
# Use this when predicting with forward stepwise
```

The coefficients the model chose were longitude, year, support, life_expectancy, freedom, trust and generosity.

```r
predict.regsubsets <- function (object, newdata , id, ...){
  form <- as.formula(object$call[[2]])  # formula of null model
  mat <- model.matrix(form, newdata)    # building an "X" matrix from newdata
  coefi <- coef(object, id = id)        # coefficient estimates associated with the object model contai
  xvars <- names(coefi)            # names of the non-zero coefficient estimates
  return(mat[,xvars] %*% coefi)    # X[,non-zero variables] %*% Coefficients[non-zero variables]
}
# Function to predict on forward stepwise
```

```r
fwd.pred <- predict.regsubsets(forward.fit, newdata = test_data, id = which.min(forward.summary$bic))
head(fwd.pred)
```

```
##       [,1]
## 2 4.946054
## 3 7.123265
## 4 6.961660
## 5 5.621817
## 6 4.247023
## 7 5.850724
```

```r
fwd.mse <- mean((fwd.pred - test_data$score)^2)
corr_coef_fwd <- cor(fwd.pred, test_data$score)
paste(c("Forward Stepwise Mean Squared Error:",fwd.mse))
```

```
## [1] "Forward Stepwise Mean Squared Error:"
## [2] "0.313334187427066"
```

```r
paste(c("Forward Stepwise Correlation Coefficient:",corr_coef_fwd))
```

```
## [1] "Forward Stepwise Correlation Coefficient:"
## [2] "0.841019468262666"
```

Use the final model(s), predict the happiness values of countries in the data set, and check deviation from true value.

```r
#  use lm.fit2
lin.pred <- predict(lm.fit2, newdata = test_data)
head(lin.pred)
```

```
##        2        3        4        5        6        7
## 4.946054 7.123265 6.961660 5.621817 4.247023 5.850724
```

```r
# predictions for happiness values from linear model
```

```r
lin.mse <- mean((lin.pred - test_data$score)^2)
corr_coef <- cor(lin.pred, test_data$score)
paste(c("Linear Mean Squared Error:",lin.mse))
```

```
## [1] "Linear Mean Squared Error:" "0.313334187427027"
```

```r
paste(c("Linear Correlation Coefficient:",corr_coef))
```

```
## [1] "Linear Correlation Coefficient:" "0.841019468262674"
```

Forward Stepwise and Linear Regression produced the same predictions. Since these predicted the same values, I will try another Regression method that will eliminate empty values.

Lasso Regression

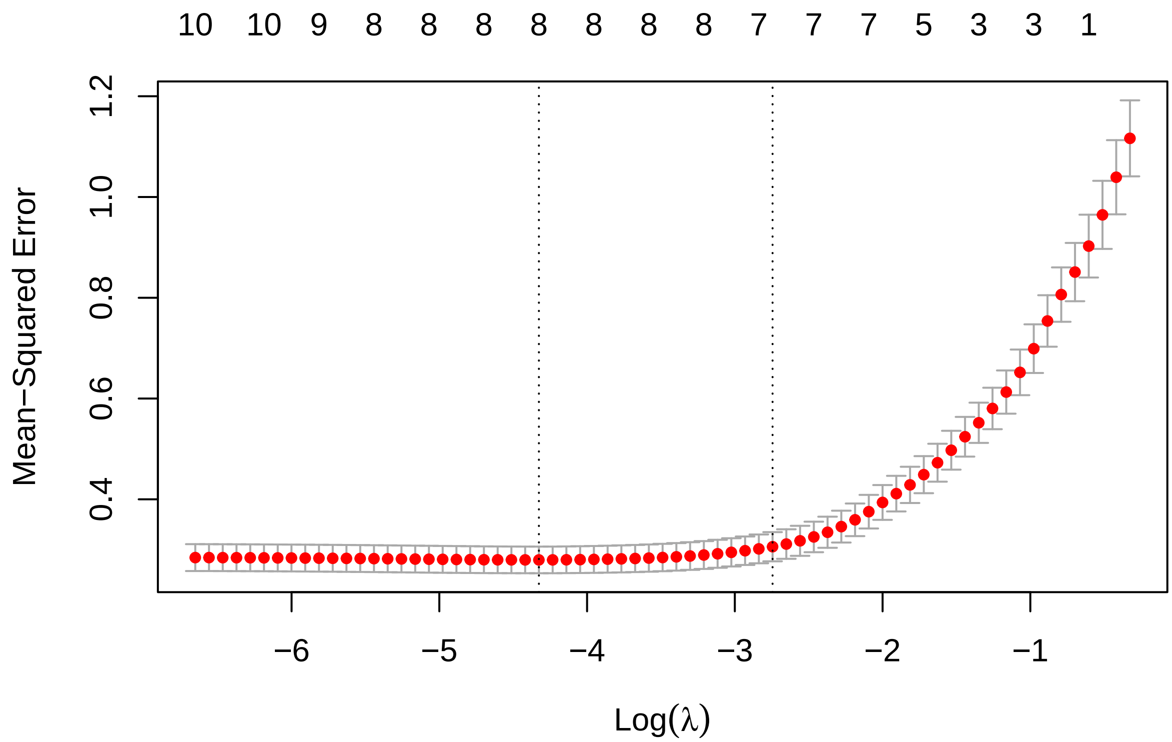
```r
set.seed(1)
lasso.fit <- glmnet(scaled.data, sco, alpha = 1)
names(lasso.fit)
```

```
##  [1] "a0"        "beta"      "df"        "dim"       "lambda"    "dev.ratio"
##  [7] "nulldev"   "npasses"   "jerr"      "offset"    "call"      "nobs"
```

```r
cv.lasso <- cv.glmnet(scaled.data, sco, alpha = 1, nfolds = 10)
best_value <- cv.lasso$lambda.min
best_value
```

```
## [1] 0.01322112
```

```r
plot(cv.lasso)
```




```r
lasso.pred <- predict(lasso.fit, s = best_value, newx = model.matrix(score ~. - country - country_code,
head(lasso.pred)
```

```
##         s1
## 2 4.973467
## 3 7.107515
## 4 6.978274
## 5 5.584036
## 6 4.247329
## 7 5.815944
```

```r
coef(lasso.fit, s = best_value) # latitude and gdp
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                            s1
## (Intercept)     248.368512006
## latitude          .
## longitude        -0.004344675
## year             -0.121947191
## value            -0.009292751
## gdp               .
## support           0.997037306
## life_expectancy   2.236305928
## freedom           1.221756783
```

```
## trust              0.967527314
## generosity         1.756869753
```

```r
lasso.mse <- mean((lasso.pred - test_data$score)^2)
corr_coef_lasso <- cor(lasso.pred, test_data$score)
paste(c("Lasso Mean Squared Error:",lasso.mse))
```

```
## [1] "Lasso Mean Squared Error:" "0.306171906590212"
```

```r
paste(c("Lasso Correlation Coefficient:",corr_coef_lasso))
```

```
## [1] "Lasso Correlation Coefficient:" "0.842478624646353"
```

Slightly different values than the other 2 prediction models, with a slightly lower MSE.

Decision Tree Method

Since the Regression models produced nearly identical results, I will fit a Decision Tree model to see if that implies anything different.

```r
tree.fit <- tree(score ~., data = train_data_no_country)
summary(tree.fit)
```

```
##
## Regression tree:
## tree(formula = score ~ ., data = train_data_no_country)
## Variables actually used in tree construction:
## [1] "gdp"             "support"         "freedom"         "generosity"
## [5] "longitude"       "latitude"        "trust"           "life_expectancy"
## Number of terminal nodes:  12
## Residual mean deviance:  0.1866 = 43.48 / 233
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -1.21500 -0.28060  0.07689  0.00000  0.24660  2.09300
```

```r
tree.fit
```

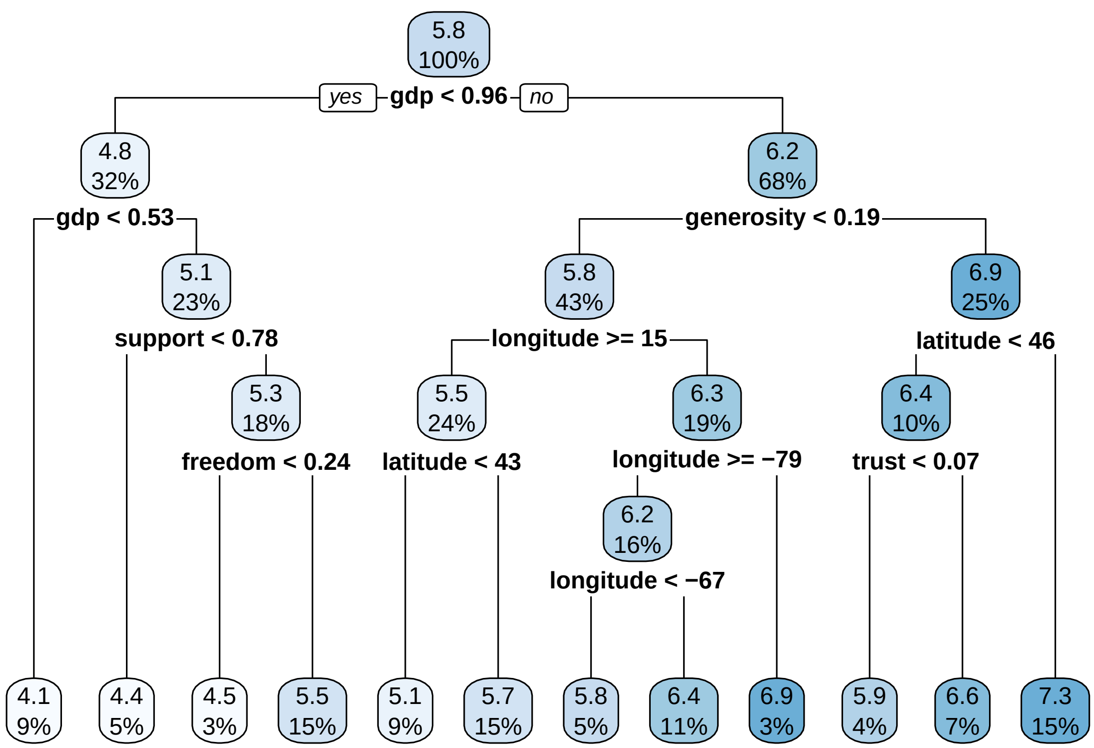
```
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 245 272.7000 5.778
##    2) gdp < 0.95554 78  49.8800 4.814
##      4) gdp < 0.531396 22   7.3870 4.054 *
##      5) gdp > 0.531396 56  24.7900 5.113
##       10) support < 0.778085 13   0.6366 4.393 *
##       11) support > 0.778085 43  15.3800 5.330
##         22) freedom < 0.199687 5   0.2149 4.140 *
##         23) freedom > 0.199687 38   7.1550 5.487 *
##    3) gdp > 0.95554 167 116.6000 6.228
```

```
##      6) generosity < 0.18517 105  46.1200 5.824
##       12) longitude < 14.6854 46  12.3400 6.285
##         24) longitude < -82.2678 6   0.3214 6.918 *
##         25) longitude > -82.2678 40   9.2500 6.190 *
##       13) longitude > 14.6854 59  16.4100 5.465
##         26) latitude < 43.3752 22   4.3420 5.111 *
##         27) latitude > 43.3752 37   7.6780 5.676 *
##      7) generosity > 0.18517 62  24.3800 6.911
##       14) latitude < 46.4847 25  10.1700 6.391
##         28) trust < 0.069655 9   0.5541 5.931 *
##         29) trust > 0.069655 16   6.6420 6.649
##           58) life_expectancy < 0.7427 5   2.2730 5.949 *
##           59) life_expectancy > 0.7427 11   0.8021 6.968 *
##       15) latitude > 46.4847 37   2.8640 7.263 *
```

For an example analysis of the tree, we can look at node 3. This node asked whether the gdp was more than 0.95554, with a number of observations of 167. If gdp was in this threshold, then the tree split the average, and if the gdp from this point was more than 0.95554, then the tree predicted that the happiness score for this gdp was 6.228.

Plot tree using rpart

```r
r.tree <- rpart(score ~ ., data = train_data_no_country)
rpart.plot(r.tree)
```



According to the decision tree, it looks like gdp has the most importance when determining happiness score.

Make predictions based on tree

```
tree.pred <- predict(tree.fit, newdata = test_data)
head(tree.pred)
```

```
##        2        3        4        5        6        7
## 4.393000 7.263081 7.263081 5.111436 4.053773 5.486816
```

Still, even a decision tree gives more or less the same results when it comes to predictions.

Since this is the case, I will try one more tree based method, to check other feature importance.

Tree MSE

```
tree.mse <- mean((tree.pred - test_data$score)^2)
corr_coef_tree <- cor(tree.pred, test_data$score)
paste(c("Tree Mean Squared Error:",tree.mse))
```

```
## [1] "Tree Mean Squared Error:" "0.347662134792833"
```

```
paste(c("Tree Correlation Coefficient:",corr_coef_tree))
```

```
## [1] "Tree Correlation Coefficient:" "0.827366627649833"
```

Random Forest

```
set.seed(1)
sqrt(ncol(train_data_no_country) - 1) # 3.16... ~ 3 >> mtry to be 3
```

```
## [1] 3.162278
```

```
rf.fit <- randomForest(score ~ ., data = train_data_no_country, mtry = 3, importance = T, ntree = 1000)
```

```
rf.fit
```

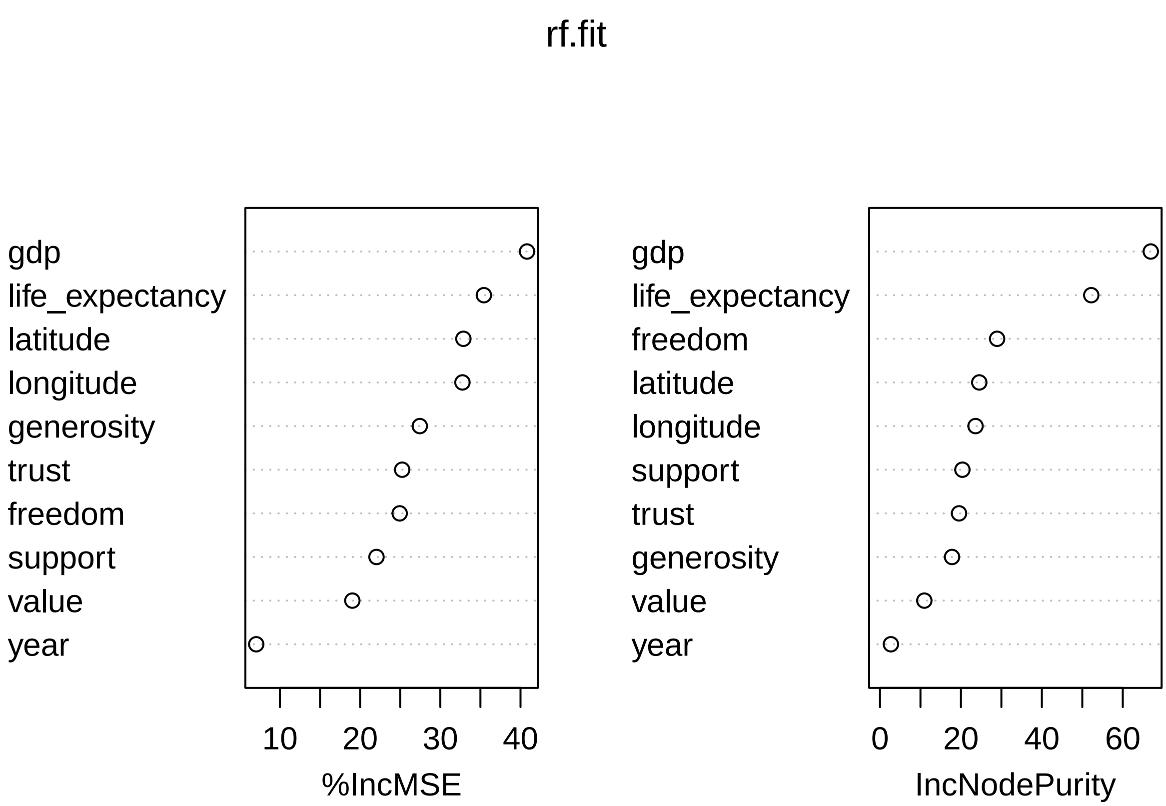
```
##
## Call:
##  randomForest(formula = score ~ ., data = train_data_no_country,      mtry = 3, importance = T, ntree
##               Type of random forest: regression
##                     Number of trees: 1000
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.1969982
##                     % Var explained: 82.3
```

Check variable importance

```
importance(rf.fit)
```

```
##                   %IncMSE IncNodePurity
## latitude        32.876577     24.537561
## longitude       32.757785     23.613281
## year             7.045148      2.693898
## value           19.026878     10.959194
## gdp             40.811750     66.919630
## support         22.039997     20.369165
## life_expectancy 35.431589     52.212861
## freedom         24.932116     28.945372
## trust           25.239232     19.535604
## generosity      27.441628     17.805915
```

```
varImpPlot(rf.fit)
```



So again, we see that gdp is included as the most important variable in the tree based method. latitude and longitude, as well as life expectancy come in as close runner-ups. Now I will see if the predictions match the other methods used.

Random Forest Predictions

```
rf.pred <- predict(rf.fit, newdata = test_data)
head(rf.pred)
```

```
##        2        3        4        5        6        7
## 4.736155 7.147611 6.990075 5.166695 4.186670 5.453177
```

Random Forest MSE

```
rf.mse <- mean((rf.pred - test_data$score)^2)
corr_coef_rf <- cor(rf.pred, test_data$score)
paste(c("RF Mean Squared Error:",rf.mse))
```

```
## [1] "RF Mean Squared Error:" "0.139186869478371"
```

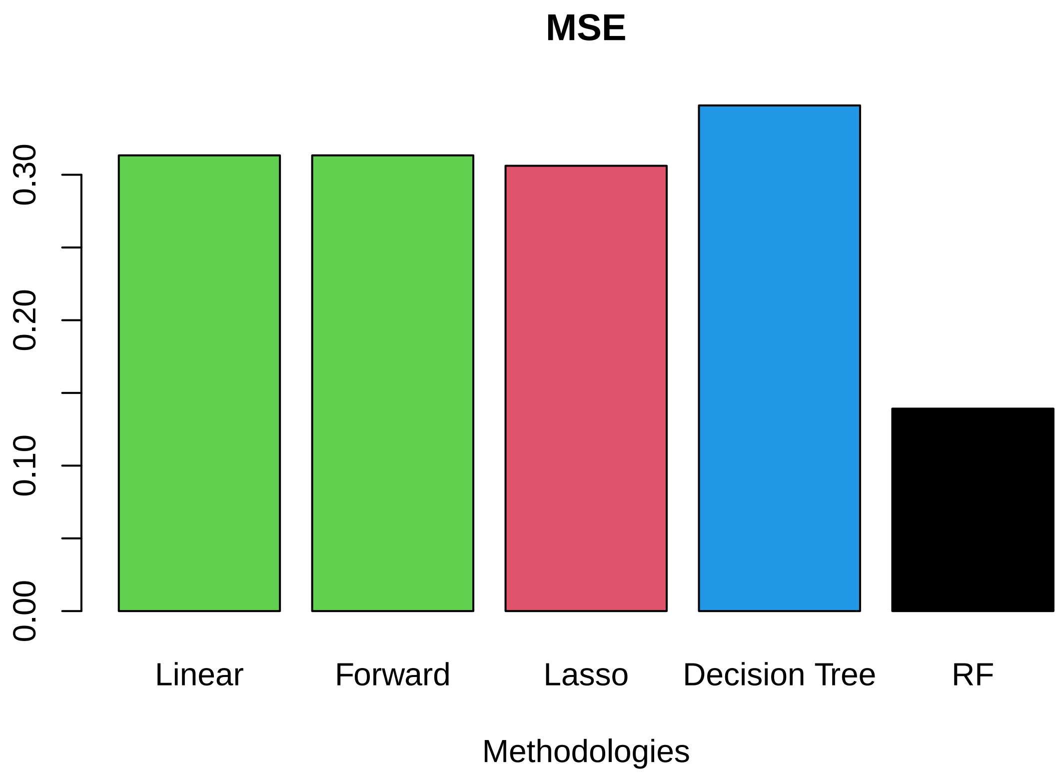
```
paste(c("RF Correlation Coefficient:",corr_coef_rf))
```

```
## [1] "RF Correlation Coefficient:" "0.936428036094232"
```

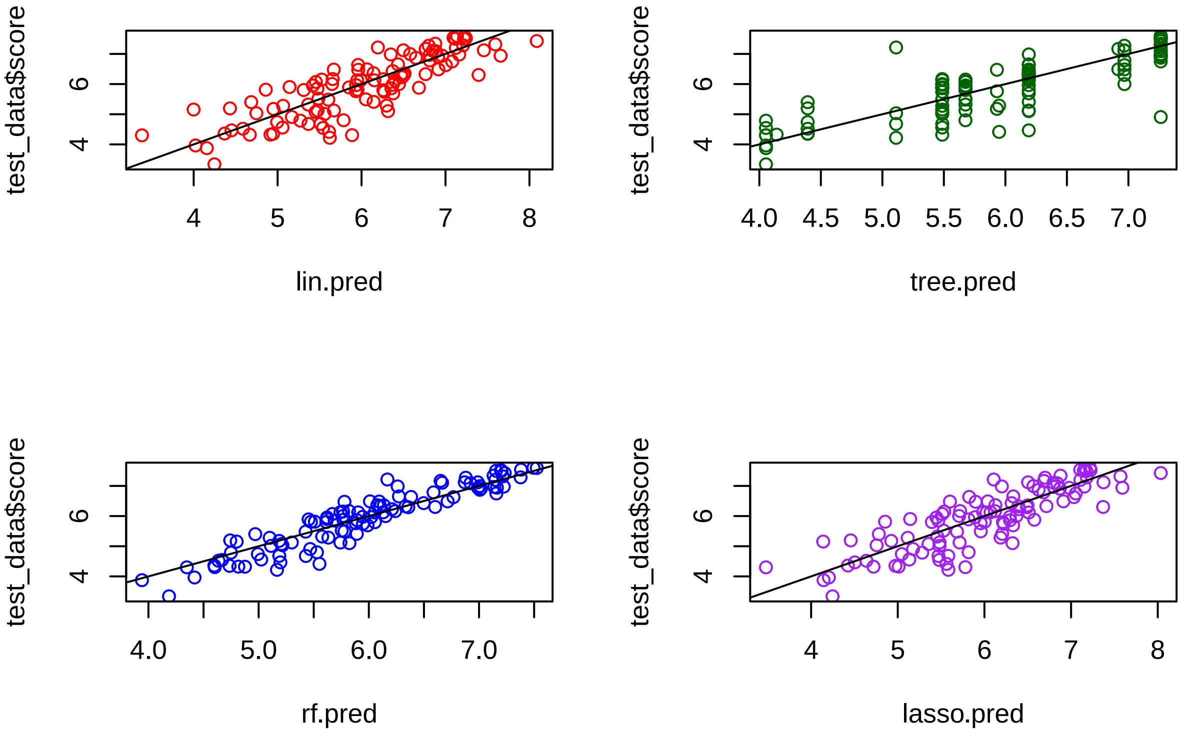
And they slightly do match up. But the random forest model has the most variance in predictions from the rest of the models made.

MSE Plot

```
mse <- c(0.3133342, 0.3133342, 0.3061719, 0.3476621, 0.1391869)

#hist(mse, main = "Mean Square Errors", xlab = "Linear // Forward // Lasso // Decision Tree // Random F
barplot(mse, main = "MSE", col = as.factor(mse), names.arg = c("Linear", "Forward", "Lasso", "Decision '
        beside = F, xlab = "Methodologies",width = 1, axisnames = T)
```

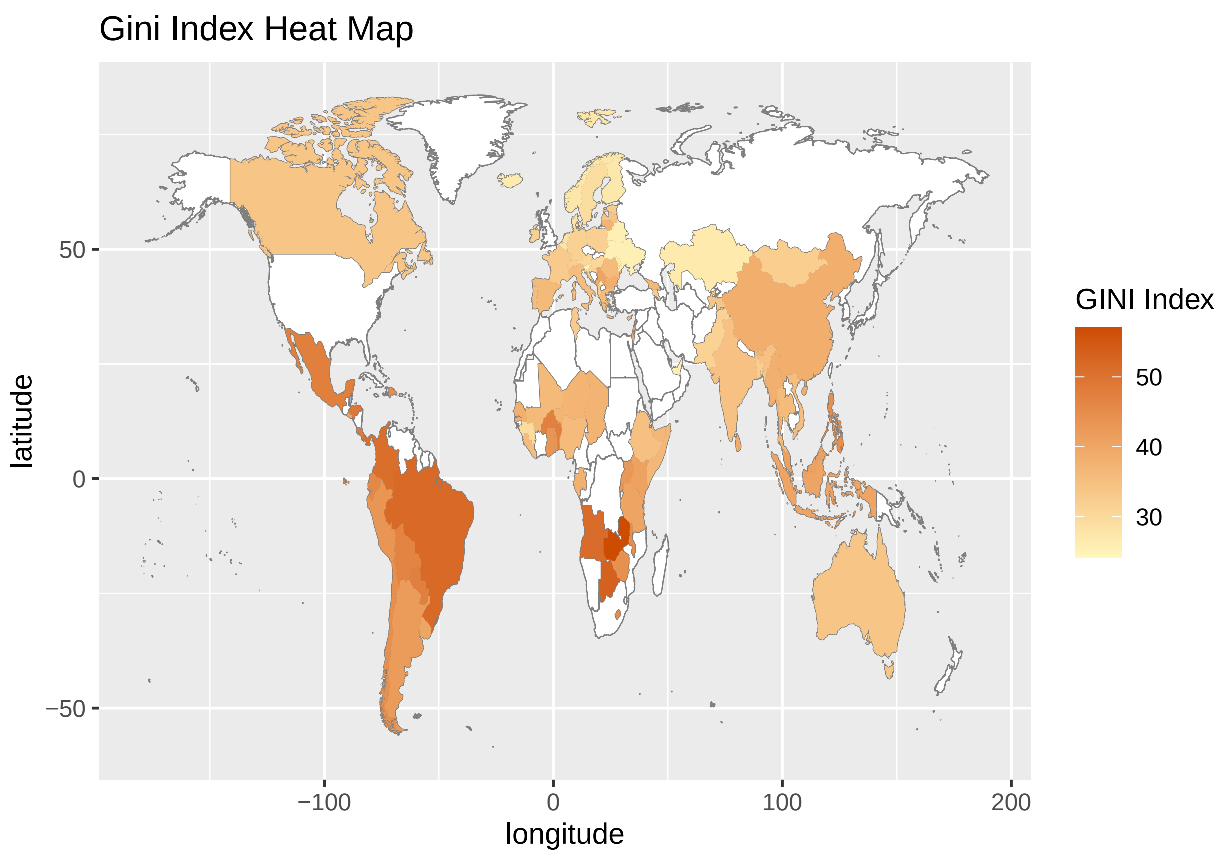



```
par(mfrow = c(2,2))
plot(lin.pred, test_data$score, col = "red")
abline(0,1)

plot(tree.pred, test_data$score, col = "darkgreen")
abline(0,1)

plot(rf.pred, test_data$score, col = "blue")
abline(0,1)

plot(lasso.pred, test_data$score, col = "purple")
abline(0,1)
```



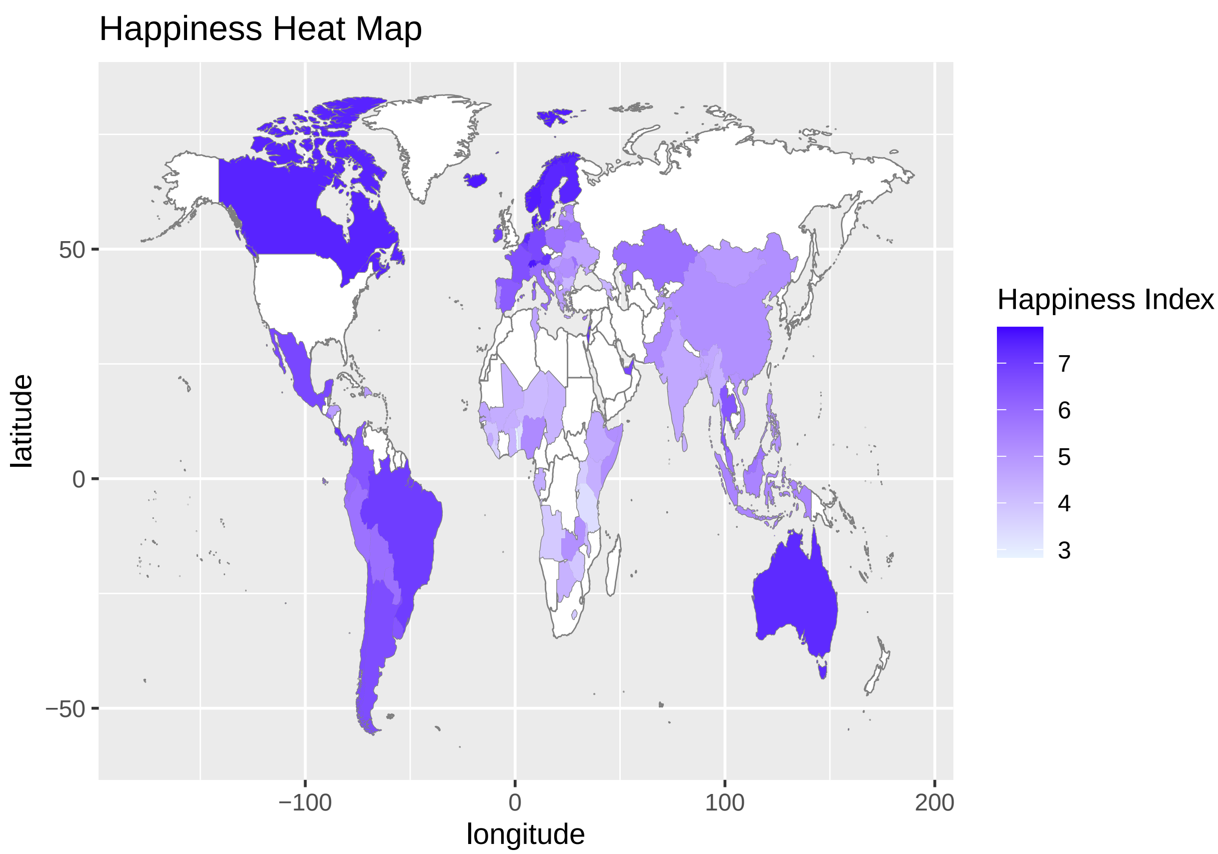
Map GINI and Happiness scores

```
library(maps)

world_map <- map_data('world')
world_map <- subset(world_map, region != 'Antarctica')

ggplot(df) + geom_map(dat=world_map, map = world_map, aes(map_id = region),
                      fill='white', color='#7f7f7f', linewidth=0.25)+ geom_map(map=world_map, aes(map_i
  labs(title='Gini Index Heat Map') + xlab('longitude') + ylab('latitude')
```



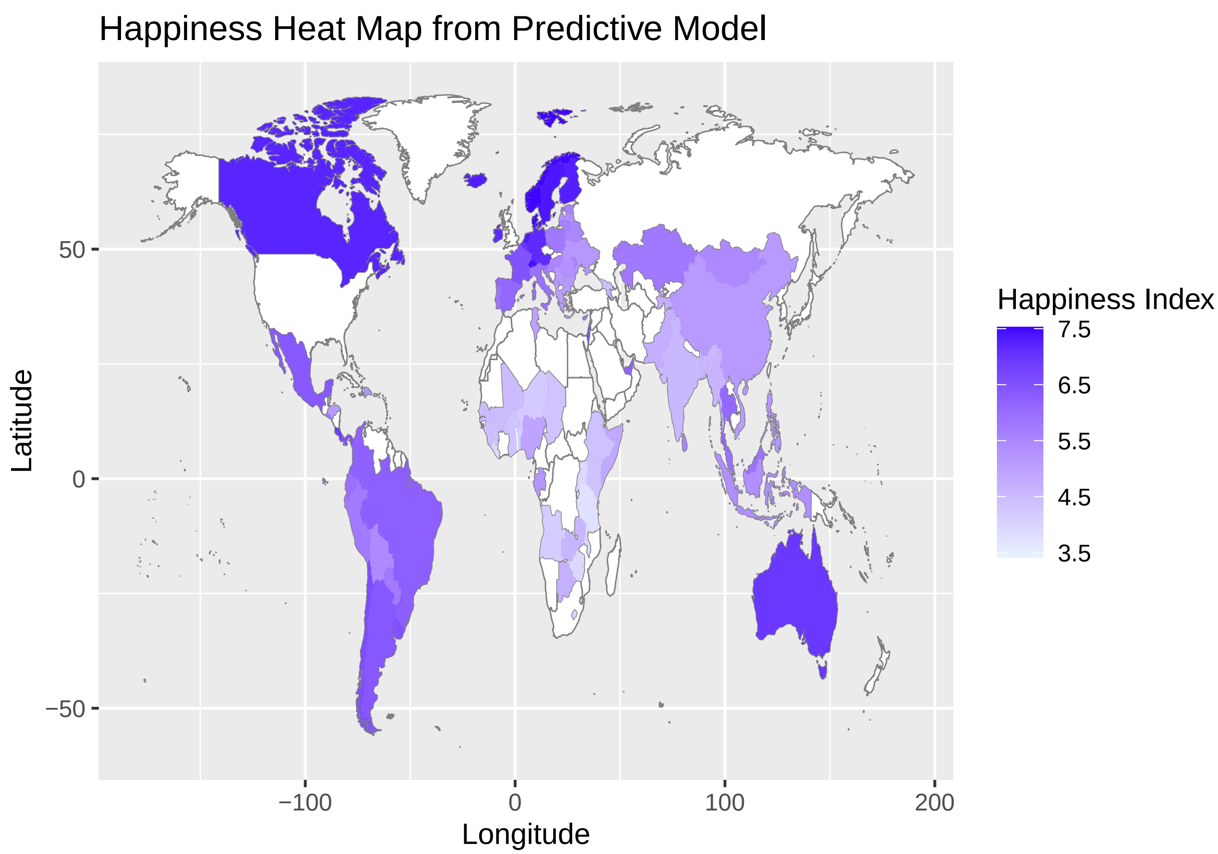
High GINI index (inequality) scores are heavily concentrated in South America and South Africa.

```
ggplot(df) + geom_map(dat=world_map, map = world_map, aes(map_id = region),
            fill='white', color='#7f7f7f', linewidth=0.25) + geom_map(map=world_map,
          aes(map_id = country, fill = score), linewidth=0.25) +
  scale_fill_gradient(low='#e8f3ff', high='#3f00ff', name='Happiness Index') +
  expand_limits(x=world_map$long, y=world_map$lat) + labs(title='Happiness Heat Map') +
  xlab('longitude') + ylab('latitude')
```



High Happiness scores found in South America, Western Europe, Australia, and Canada.

```
ggplot(df) + geom_map(dat=world_map, map = world_map, aes(map_id = region),
                     fill='white', color='#7f7f7f', linewidth=0.25) +
  geom_map(map=world_map, aes(map_id = country, fill = predict(rf.fit, df)),
           linewidth=0.25) + scale_fill_gradient(low='#e8f3ff',
            high='#3f00ff', name='Happiness Index') + expand_limits(x=world_map$long, y=world_map$lat)
```



Modeled happiness values are highly accurate. The model most frequently over-predicts happiness values, but general scores are close to actual values.