

Homework 6 - Regression

Solutions

Spring 2022

Set up

```
knitr::opts_chunk$set(echo = TRUE)
pacman::p_load(GGally, tidyverse, broom, rpart, rpart.plot, FNN, caTools)

# Changing the default GGplot theme
theme_set(theme_bw())

# Read in the data as countries
countries <- read.csv("Happiness19.csv")




# Changing the Country from a column to the row names:
countries <- countries %>%
  column_to_rownames(var = "Country")

skimr::skim(countries)
```

Data summary

Name	countries
Number of rows	156
Number of columns	7
<hr/>	
Column type frequency:	
numeric	7
<hr/>	
Group variables	None

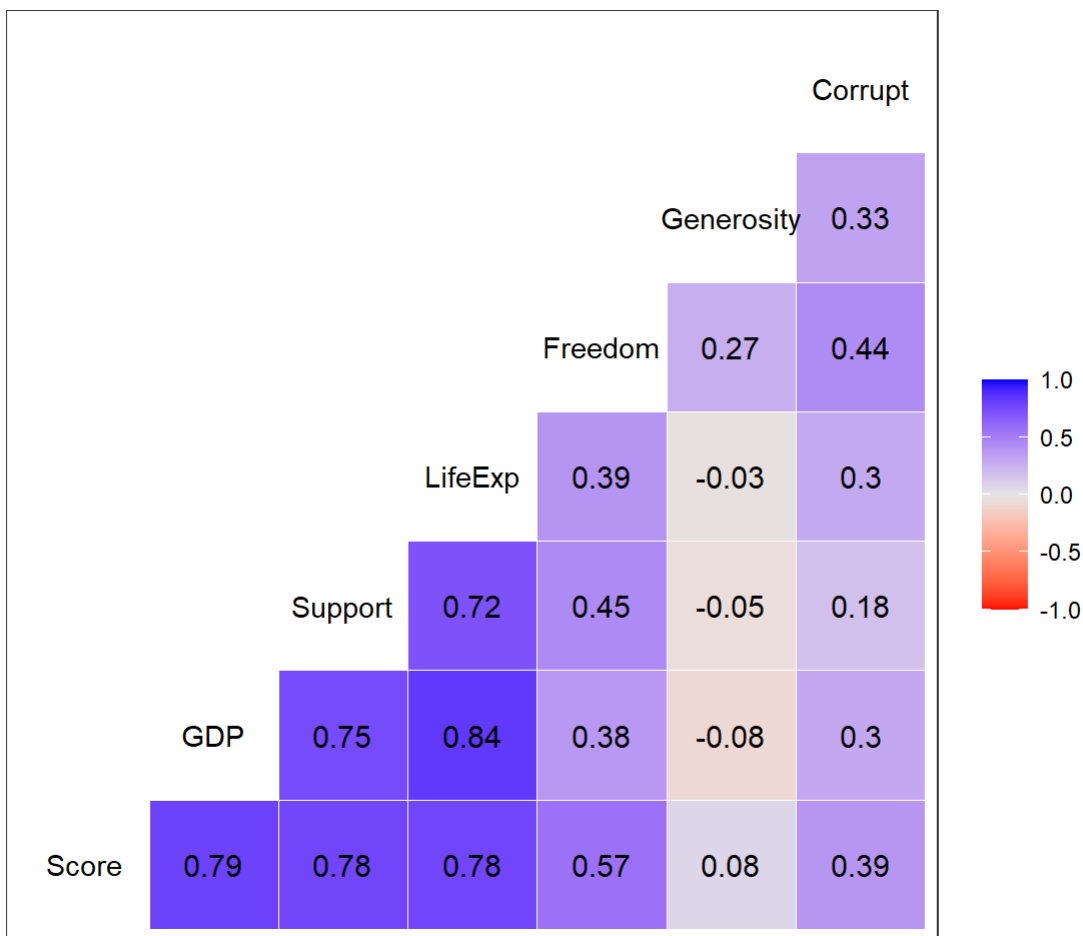
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Score	0	1	5.41	1.11	2.85	4.54	5.38	6.18	7.77	
GDP	0	1	0.91	0.40	0.00	0.60	0.96	1.23	1.68	
Support	0	1	1.21	0.30	0.00	1.06	1.27	1.45	1.62	

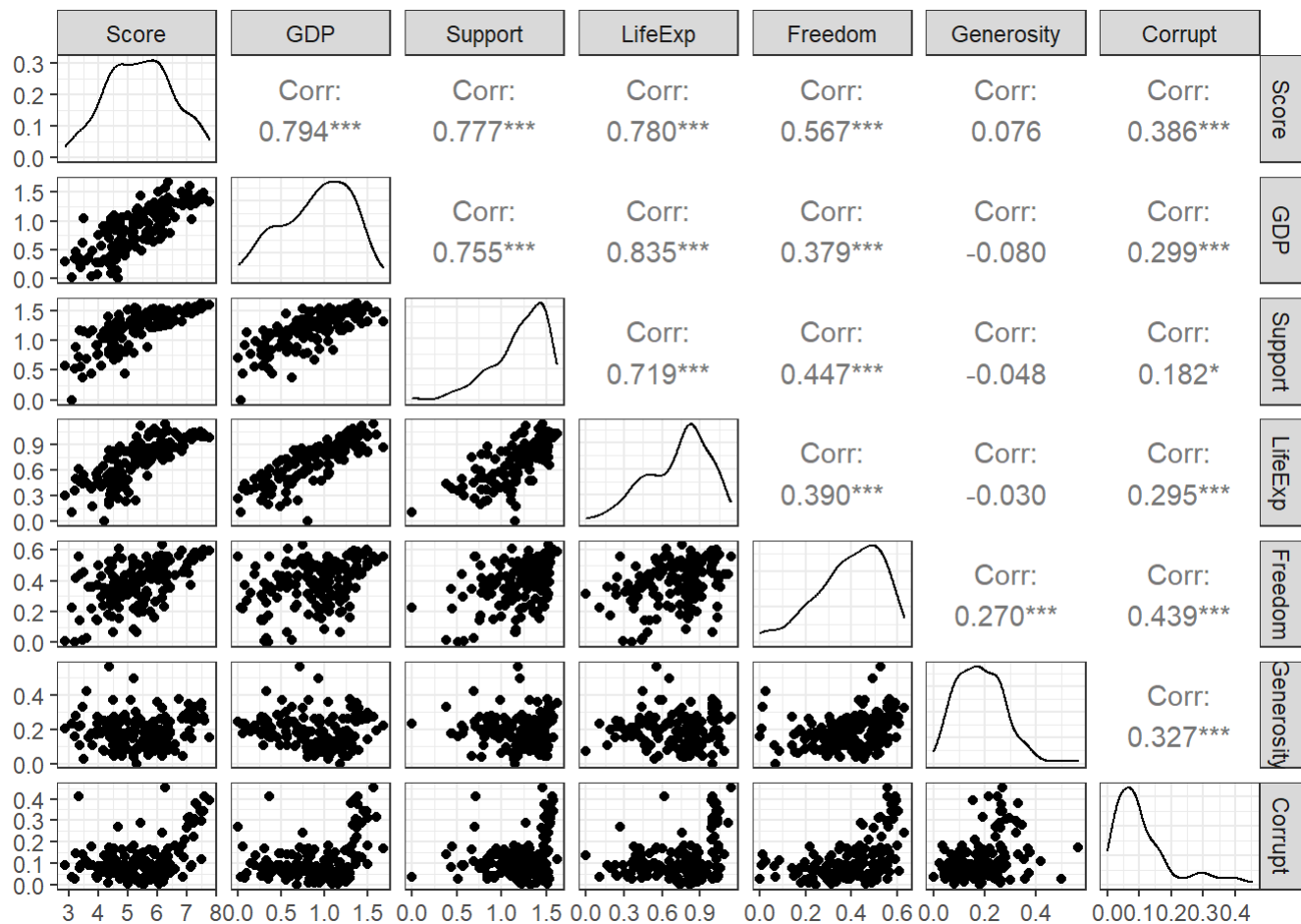
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
LifeExp	0	1	0.73	0.24	0.00	0.55	0.79	0.88	1.14	
Freedom	0	1	0.39	0.14	0.00	0.31	0.42	0.51	0.63	
Generosity	0	1	0.18	0.10	0.00	0.11	0.18	0.25	0.57	
Corrupt	0	1	0.11	0.09	0.00	0.05	0.09	0.14	0.45	

Create the correlation plot and scatterplot matrix

```
# Correlation plot:
ggcorr(data = countries,
       low = "red",
       high = "blue",
       mid = "grey90",
       label = TRUE,
       label_round = 2)
```



```
# Scatterplot matrix:
ggpairs(data = countries)
```

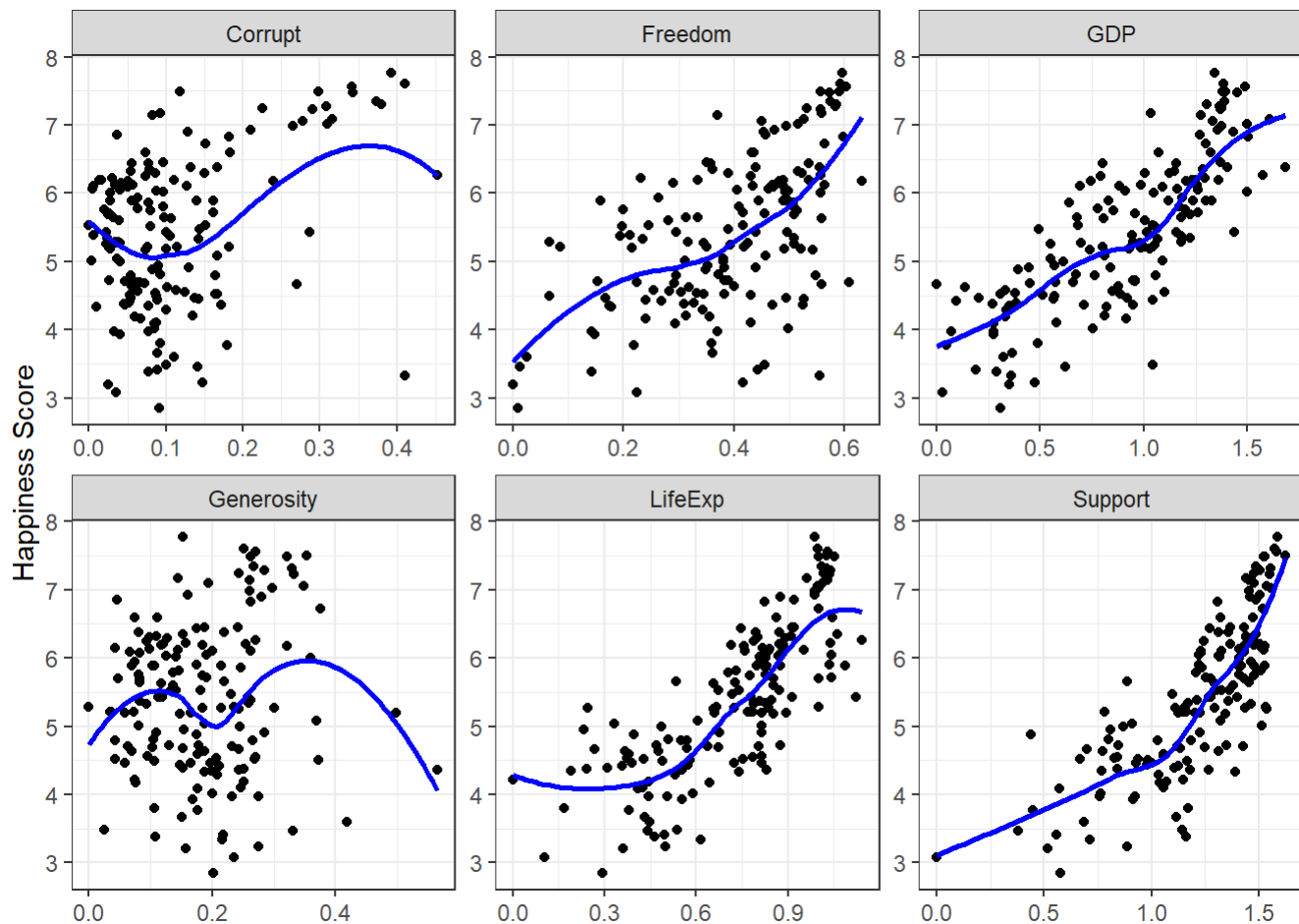


Not required: Scatterplot of Score (Outcome) vs the other 6 predictors
countries %>%

```

pivot_longer(cols = -Score,
              names_to = "features",
              values_to = "values") %>%
ggplot(mapping = aes(x = values,
                     y = Score)) +
  geom_point() +
  geom_smooth(method = "loess",
             se = F,
             color = "blue") +
  facet_wrap(facets = ~ features,
            ncol = 3,
            scales = "free") +
  labs(x = NULL,
       y = "Happiness Score")

```



Question 1: Multiple Regression of Happiness data – Description

1a) Description

We are trying to predict the happiness score of many countries, based on six predictive features: GDP, Support, Life Expectancy, Freedom, Generosity, and Corruption. We'll use the multiple regression, which creates a linear model predicting happiness Score as a linear function of the prediction variables.

1b) First Model

GDP, Support, Life Expectancy, and Freedom are all positive predictors of Score. The R-squared is 77.9%, indicating that the predictions are fairly reliable

```
# Create a Linear model to predict Score, using all of the other variables, and summarize the model.
happy_lm <- lm(formula = Score ~ .,
               data = countries)

# Showing the model terms
tidy(happy_lm)
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	1.7952202	0.2110734	8.5051942	1.767598e-14
GDP	0.7753716	0.2182254	3.5530776	5.103022e-04
Support	1.1241916	0.2369001	4.7454252	4.833807e-06
LifeExp	1.0781427	0.3345385	3.2227764	1.559604e-03
Freedom	1.4548324	0.3753378	3.8760610	1.586893e-04
Generosity	0.4897834	0.4977455	0.9840036	3.267089e-01
Corrupt	0.9722802	0.5423607	1.7926818	7.505259e-02

7 rows

```
# Displaying the fit statistics
glance(happy_lm)
```

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>	df <dbl>	logLik <dbl>	AIC <dbl>	B <dbl>
0.7791638	0.7702711	0.5335189	87.61804	2.400657e-46	6	-119.7648	255.5295	279.92

1 row | 1-10 of 12 columns

1c)

```
# Finding the variance inflation factor for the predictor features
regclass::VIF(happy_lm) %>% round(digits = 2)
```

##	GDP	Support	LifeExp	Freedom	Generosity	Corrupt
##	4.12	2.74	3.57	1.58	1.22	1.43

Using the correlation plot from the initial set up, no pair of predictor features have an extremely high correlation, with the highest belonging to GDP and LifeExp at 0.84.

Calculating the variance inflation factor for both, none of the predictors have a VIF over 5. GDP has the highest at 4.12, but that isn't large enough to be concerned about multicollinearity.

1d) Second Model

GDP, Support, Life Expectancy, and Freedom are still all positive predictors of Score.

The R squared is 79.92%, which is only 0.01% higher than the model without corruption² and the sigma is essentially the same. Including the squared term doesn't increase the predictive power by a significant margin.

```
# Creating a squared term for Corrupt
countries <- countries %>%
  mutate(Corrupt_sq = Corrupt^2)

# Fitting model 2 with corrupt^2
happy2_lm <- lm(Score ~ ., data = countries)

# Summary of the model
tidy(happy2_lm)
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	1.7708882	0.2397652	7.3859254	1.021143e-11
GDP	0.7723799	0.2193630	3.5210122	5.717787e-04
Support	1.1288067	0.2386169	4.7306237	5.176886e-06
LifeExp	1.0866523	0.3379106	3.2157980	1.597582e-03
Freedom	1.4501710	0.3771601	3.8449748	1.785770e-04
Generosity	0.4918236	0.4994343	0.9847613	3.263485e-01
Corrupt	1.2834075	1.5375629	0.8347025	4.052306e-01
Corrupt_sq	-0.8368758	3.8681510	-0.2163503	8.290126e-01

8 rows

```
# Comparing the fit statistics fo the 2 models
bind_rows(glance(happy_lm),
  glance(happy2_lm))
```

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>	df <dbl>	logLik <dbl>	AIC <dbl>	BIC <dbl>
0.7791638	0.7702711	0.5335189	87.61804	2.400657e-46	6	-119.7648	255.5295	279.92
0.7792336	0.7687920	0.5352336	74.62742	2.296937e-45	7	-119.7401	257.4802	284.92

2 rows | 1-10 of 12 columns

1e) Third Model

GDP, Support, Life Expectancy, and Freedom are still all positive predictors of Score, and also Corruptbin. The fit statistics are a little higher for this model than the first model we fit.

```
# Create a dummy variable, called Corruptbin, that is 1 if Corrupt is > .2 and 0 otherwise.
countries <-
  countries %>%
  mutate(Corruptbin = if_else(Corrupt > 0.2, 1, 0))

# Fitting the third model, replacing corruption with corruptbin
happy3_lm <- lm(Score ~ GDP + Support + LifeExp + Freedom + Generosity + Corruptbin,
  data = countries)

# Checking the summary and fit stats
tidy(happy3_lm)
```

term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
(Intercept)	1.9786409	0.2201918	8.9859880	1.060833e-15
GDP	0.7833083	0.2129887	3.6776987	3.280005e-04
Support	1.0809657	0.2298367	4.7031902	5.790900e-06
LifeExp	1.0122700	0.3312620	3.0557992	2.660507e-03
Freedom	1.4360192	0.3609207	3.9787665	1.078060e-04
Generosity	0.3341929	0.4935968	0.6770563	4.994200e-01
Corruptbin	0.4142093	0.1497214	2.7665333	6.383544e-03
7 rows				

```
# Fit statistics of the 3 models
bind_rows(glance(happy_lm),
  glance(happy2_lm),
  glance(happy3_lm))
```

r.squared <dbl>	adj.r.squared <dbl>	sigma <dbl>	statistic <dbl>	p.value <dbl>	df <dbl>	logLik <dbl>	AIC <dbl>	BIC <dbl>
0.7791638	0.7702711	0.5335189	87.61804	2.400657e-46	6	-119.7648	255.5295	279.92
0.7792336	0.7687920	0.5352336	74.62742	2.296937e-45	7	-119.7401	257.4802	284.92
0.7854229	0.7767822	0.5259038	90.89819	2.863435e-47	6	-117.5221	251.0442	275.44
3 rows 1-10 of 12 columns								

1f) Model Selection

The fit of all 3 models appears to be about equal. If the models predict the outcome variable about equally well, then we choose the simplest one. Usually that is the one with the fewest predictors, but we have 2 models with 6 predictors: Model 1 and Model 3.

Model 3 fits better (and is a little simpler to use since we just need to know if corruption is greater than 0.2 rather than the actual value), so we should pick model 3!

1g) Residual Plot Assess Model Fit

```
# Predicting happiness score, adding it to the original data set, then finding the residuals
countries2 <-
  data.frame(countries,
             pred_score = predict(happy3_lm)) %>%
  mutate(score_res = Score - pred_score)

# Calculating the correlation between y and y_hat
cor_multireg <- cor(countries2$pred_score,
                   countries2$Score)

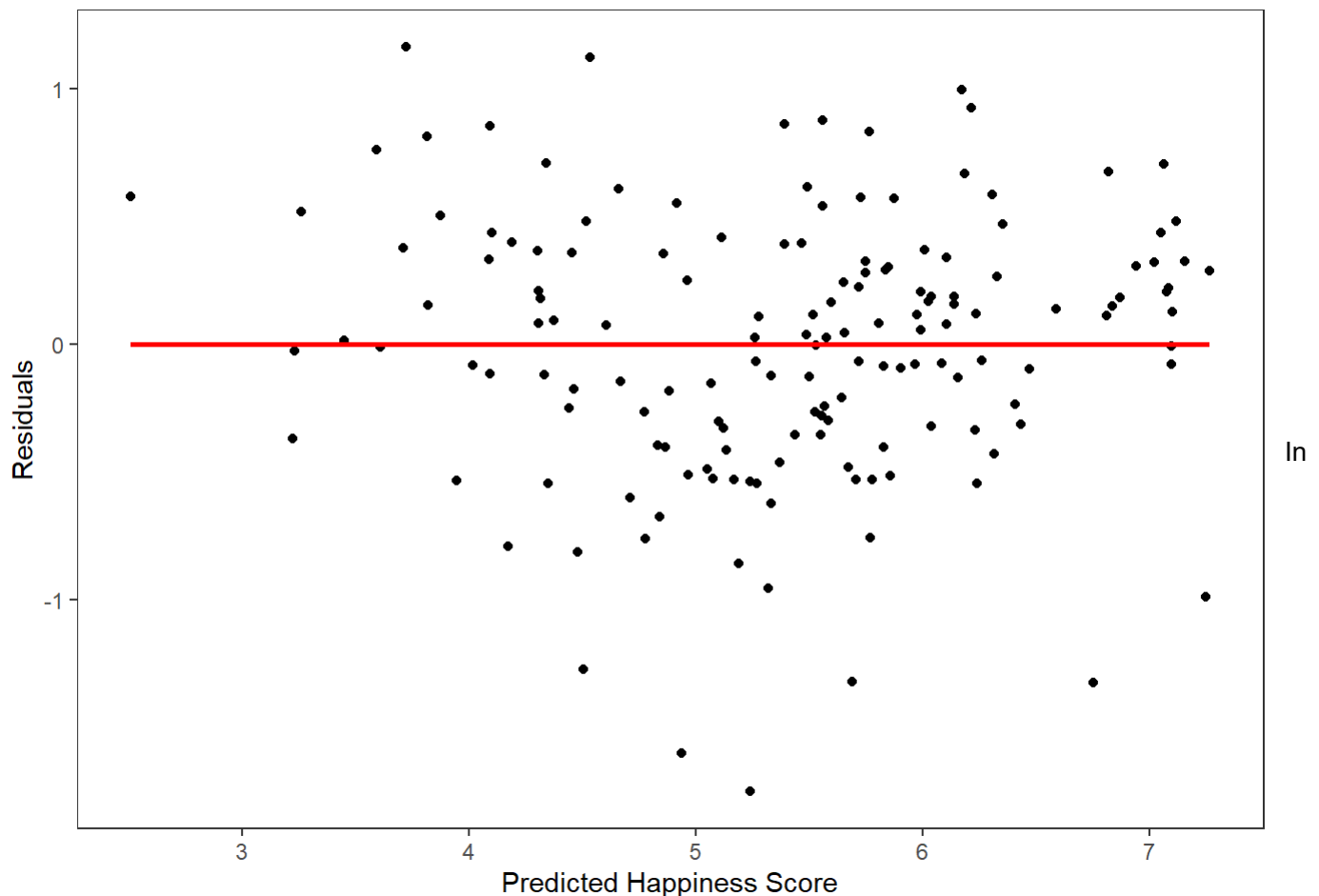
cor_multireg
```

```
## [1] 0.8862409
```

```
# Creating the residual plot: residuals vs predicted outcome values
# But first, we need to calculate the residuals: Actual - predicted

countries2 %>%
  ggplot(mapping = aes(x = pred_score,
                      y = score_res)) +
  geom_point() +
  geom_smooth(method = "lm",
             se = F,
             color = "red") +
  labs(x = "Predicted Happiness Score",
       y = "Residuals") +
  theme_test()
```

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

the best model, the predictions have a correlation of 0.8862409 with the actual happiness scores, which indicates the regression model predict happiness scores accurately.

The residual plot looks appropriate, with no curving patterns or increasing/decreasing spread. There are a couple of larger residuals where the predictions are off by 1 or more. Could be worth checking which countries they are!

1h) Assessing Model Fit

```
# Function for MAE
MAE <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

# the mean of scores that we'll use for the MAE
mean(countries$Score)
```

```
## [1] 5.407096
```

```
# Model MAE
MAE_mlr <- MAE(actual = countries2$Score,
               predicted = countries2$pred_score)

# MAE using the average happiness score
MAE_avg <- MAE(actual = countries2$Score,
               predicted = mean(countries2$Score))

# Calculating the improvement using the MLR model compared to just predicting the mean happiness score
MAE_mlr_improve <- MAE_avg - MAE_mlr

c(multireg = MAE_mlr,
  average = MAE_avg,
  improvement = MAE_mlr_improve,
  improve_prop = MAE_mlr_improve/MAE_avg) %>%

round(digits = 3)
```

```
##      multireg      average  improvement  improve_prop
##      0.401      0.917      0.516      0.563
```

Predictions are off by, on average, 0.400 points (the MAE). For comparison, they would be off by 0.917 if we just used the mean happiness score to predict for all countries, so we have improved MAE by 0.516, an average error reduction of about 56%!

Question 2: Regression Tree

```
countries <-
  read.csv("Happiness19.csv") %>%
  column_to_rownames(var = "Country")
```

Part 2a) Description

Using Regression Tree analysis, we will create a decision tree that will predict the happiness score, based on splits of various predictive features. The prediction is the mean of cases at each final node.

Part 2b) Divide into Training and Test sets

```
# Keep this at the top of the code chunk
RNGversion('4.0.0')
set.seed(187)

# Form the split vector below:

country_split <- sample.split(Y = countries$Score,
                             SplitRatio = 100)

# Printing the first 5 rows of the training data
country_train <- countries[country_split, ]

head(country_train, 5)
```

	Score <dbl>	GDP <dbl>	Support <dbl>	LifeExp <dbl>	Freedom <dbl>	Generosity <dbl>	Corrupt <dbl>
Finland	7.769	1.340	1.587	0.986	0.596	0.153	0.393
Denmark	7.600	1.383	1.573	0.996	0.592	0.252	0.410
Iceland	7.494	1.380	1.624	1.026	0.591	0.354	0.118
Netherlands	7.488	1.396	1.522	0.999	0.557	0.322	0.298
Sweden	7.343	1.387	1.487	1.009	0.574	0.267	0.373

5 rows

```
# Printing the first 5 rows of the testing data
country_test <- countries[!country_split, ]

head(country_test, 5)
```

	Score <dbl>	GDP <dbl>	Support <dbl>	LifeExp <dbl>	Freedom <dbl>	Generosity <dbl>	Corrupt <dbl>
Norway	7.554	1.488	1.582	1.028	0.603	0.271	0.341
Switzerland	7.480	1.452	1.526	1.052	0.572	0.263	0.343
New Zealand	7.307	1.303	1.557	1.026	0.585	0.330	0.380
Austria	7.246	1.376	1.475	1.016	0.532	0.244	0.226
Australia	7.228	1.372	1.548	1.036	0.557	0.332	0.290

5 rows

Part 2c) Create the Regression Tree for the Training set

Use the code chunk below to fully form the tree and find where to prune the tree

```
# Keep this at the top of the code chunk
RNGversion("4.0.0")
set.seed(12345)

# Fully growing the tree:
country_full_tree <- rpart(Score ~ .,
                           data = country_train,
                           method = "anova",
                           minbucket = 1,
                           minsplit = 2,
                           cp = -1)

## Finding where to split the tree
# First find min(xerror) + xstd
prune_point <-
  country_full_tree$cptable %>%
  data.frame() %>%
  filter(xerror == min(xerror)) %>%
  mutate(prune_xerror = xerror + xstd)

prune_point
```

	CP <dbl>	nsplit <dbl>	rel.error <dbl>	xerror <dbl>	xstd <dbl>	prune_xerror <dbl>
7	0.01412069	6	0.1689957	0.3212295	0.04438937	0.3656189
1 row						

```
# Now find the cp value used to prune the tree
country_full_tree$cptable %>%
  data.frame() %>%
  filter(xerror <= prune_point$prune_xerror)
```

	CP <dbl>	nsplit <dbl>	rel.error <dbl>	xerror <dbl>	xstd <dbl>
3	0.04098785	2	0.3051507	0.3623409	0.04642459
4	0.04007536	3	0.2641628	0.3589736	0.04509930
5	0.02969972	4	0.2240875	0.3490858	0.04326619
6	0.02539201	5	0.1943877	0.3368813	0.04251701
7	0.01412069	6	0.1689957	0.3212295	0.04438937
8	0.01259375	7	0.1548750	0.3529126	0.05075871
9	0.01022148	8	0.1422813	0.3613608	0.05577327
7 rows					

```
# Find the rows 2 & 3 to find the cp value to prune the tree
country_full_tree$cptable %>%
  data.frame() %>%
  slice(2:3)
```

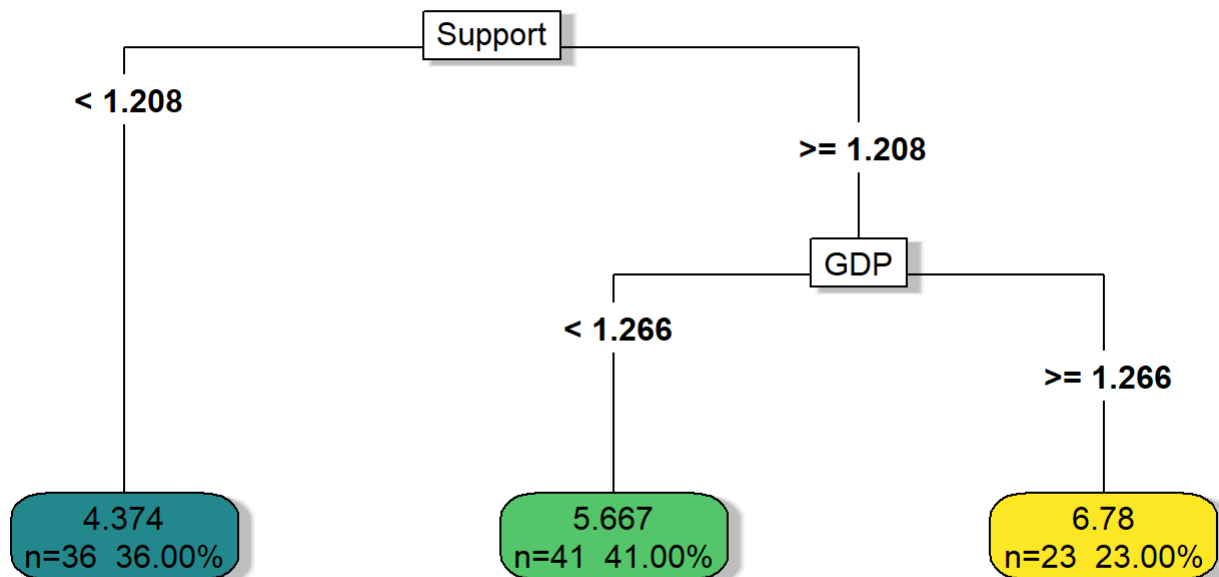
	CP <dbl>	nsplit <dbl>	rel.error <dbl>	xerror <dbl>	xstd <dbl>
2	0.15035898	1	0.4555096	0.5002500	0.05996219
3	0.04098785	2	0.3051507	0.3623409	0.04642459
2 rows					

Should use $cp = 0.05$ (or any choice between 0.0409 to 0.15)

```
# Prune the tree below
country_regtree <- prune(tree = country_full_tree,
  cp = 0.05)
```

Part 2d) Create Visual of Regression Tree

```
rpart.plot(country_regtree,
  digits = 4,
  type = 5,
  extra = 101,
  box.palette = 'BlGnYl',
  shadow.col = 'gray')
```



We can see that Support is an important predictor: Countries with higher support have substantially higher happiness score predictions.

GDP also makes a big difference: higher GDP predicts higher score.

2e) Generate predictions and correlations

i) Scatterplot

```

pred_country_regtree <- predict(country_regtree,
                                newdata = country_test)

# Creating the scatterplot:
data.frame(Actual = country_test$Score,
            pred_country_regtree) %>%

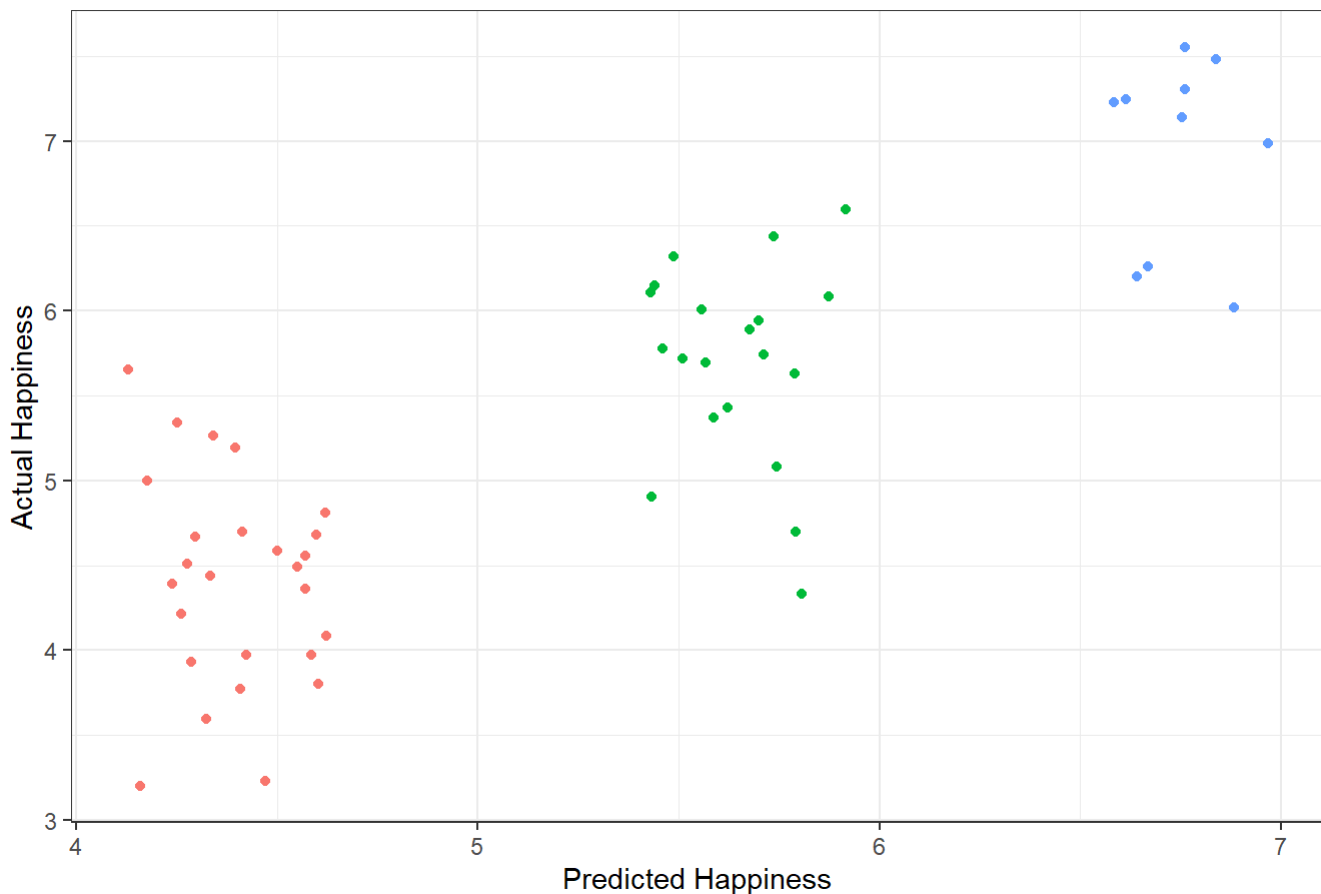
  ggplot(mapping = aes(x = pred_country_regtree,
                        y = Actual,
                        color = factor(pred_country_regtree))) +
  geom_jitter(show.legend = F,
              width = 0.25,
              height = 0) +

  labs(x = "Predicted Happiness",
        y = "Actual Happiness",
        title = "Predicted Happiness Using Regression Tree") +

  theme_bw()

```

Predicted Happiness Using Regression Tree



Since the pruned tree only has 3 leaf nodes, the tree will only make 3 predictions, around 4.5, 5.5, and 6.75.

ii) Summary Stats

```
# Adding the predictions for knn with k = 27 in the same data set as the Actual Score
data.frame(country_test,
            tree_pred = pred_country_regtree) %>%

# Calculating the 5 number summary for both variables using quantile
summarize(Actual = quantile(Score, probs = 0:4/4),
          Predicted = quantile(tree_pred, probs = 0:4/4)) %>%

mutate(Actual = round(Actual, digits = 2),
       Predicted = round(Predicted, digits = 2)) %>%

# Adding a column to indicate what the summary is
add_column(Summary = c("Min", "Q1", "Median", "Q3", "Max"),
           .before = "Actual") # Moving the Summary column before the Actual Column
```

Summary <chr>	Actual <dbl>	Predicted <dbl>
Min	3.20	4.37
Q1	4.48	4.37
Median	5.30	5.67
Q3	6.09	5.67
Max	7.55	6.78
5 rows		

Because there are only 3 leaf nodes, the 5 number summary won't make much sense because there are only 3 predicted values!

iii) Correlation

```
# Correlation between y and y_hat
cor_regtree <- cor(pred_country_regtree,
                  country_test$Score)

cor_regtree
```

```
## [1] 0.8534811
```

The correlation of predictions with actual scores is 0.853, which is much higher than would be expected for only having 3 different predicted values!

Part 2f) Calculate the Mean Absolute Error


```
# Finding the Mean Average Error for y_hat and y_bar
MAE_regtree <- MAE(country_test$Score,
                  pred_country_regtree)

MAE_ybar <- MAE(country_test$Score,
               mean(country_test$Score))

# Checking the improvement in MAE for the regression tree
MAE_regtree_improve <- MAE_ybar - MAE_regtree

c(RegTree = MAE_regtree,
  average = MAE_ybar,
  improvement = MAE_regtree_improve,
  improve_prop = MAE_regtree_improve/MAE_ybar) %>%
  round(3)
```

##	RegTree	average	improvement	improve_prop
##	0.482	0.944	0.462	0.490

Happiness Score predictions using the Regression Tree are off by about 0.482 points, on average. This doesn't seem unreasonable. If we used the overall average of the scores as a prediction, we'd be off by 0.9440536, on average, so the model improves MAE by 0.4621963, an improvement of 49 percent.

Question 3: KNN Regression

Part 3a) Description

Using k-Nearest-Neighbors Regression, we will predict the happiness score for countries using the average happiness score of the k closest countries.

Which countries are considered the closest is based on distance using rescaled predictive features.

Part 3b) Training and Testing Data for KNN Regression

```
# Standardizing the data:
countries_stan <-
  countries %>%
  mutate(across(.cols = -Score,
                .fns = ~ (. - mean(.)) / sd(.)))

## Use the same split vector from question 2 to form the training and testing data

# Training:
country_train3 <- countries_stan[country_split, ]
head(country_train3)
```

Score	GDP	Support	LifeExp	Freedom	Generosity	Corrupt
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>

	Score <dbl>	GDP <dbl>	Support <dbl>	LifeExp <dbl>	Freedom <dbl>	Generosity <dbl>	Corrupt <dbl>
Finland	7.769	1.091526	1.2640266	1.076954	1.419710	-0.3343272	2.98713668
Denmark	7.600	1.199461	1.2172338	1.118255	1.391794	0.7049944	3.16695887
Iceland	7.494	1.191931	1.3876933	1.242159	1.384816	1.7758106	0.07824842
Netherlands	7.488	1.232092	1.0467744	1.130646	1.147534	1.4398683	1.98224801
Sweden	7.343	1.209501	0.9297924	1.171947	1.266175	0.8624674	2.77558117
Canada	7.278	1.154279	0.9899546	1.295850	1.335963	1.0514349	2.08802576
6 rows							

Training:

```
country_test3 <- countries_stan[!country_split, ]
head(country_test3)
```

	Sc... <dbl>	GDP <dbl>	Support <dbl>	LifeExp <dbl>	Freedom <dbl>	Generosity <dbl>	Corrupt <dbl>
Norway	7.554	1.4630220	1.2473149	1.250419	1.4685621	0.9044602	2.4370924
Switzerland	7.480	1.3726582	1.0601438	1.349542	1.2522168	0.8204746	2.4582479
New Zealand	7.307	0.9986523	1.1637564	1.242159	1.3429423	1.5238539	2.8496256
Austria	7.246	1.1818901	0.8896843	1.200857	0.9730616	0.6210088	1.2206482
Australia	7.228	1.1718497	1.1336753	1.283460	1.1475336	1.5448502	1.8976258
Israel	7.139	0.9308794	0.8228375	1.254549	-0.1505380	0.7994782	-0.3025515
6 rows							

Part 3c) Find the best choice of k by using R-Squared

```
# Use the choices of k below:
k_choice <- 5:30

# Create a vector to store the R-squared results
R2_k <- NULL

# Looping through the choices:
for (i in k_choice) {
  knn_temp <- knn.reg(train = country_train3 %>% dplyr::select(-Score),
                      test = country_test3 %>% dplyr::select(-Score),
                      y = country_train3$Score,
                      k = i)

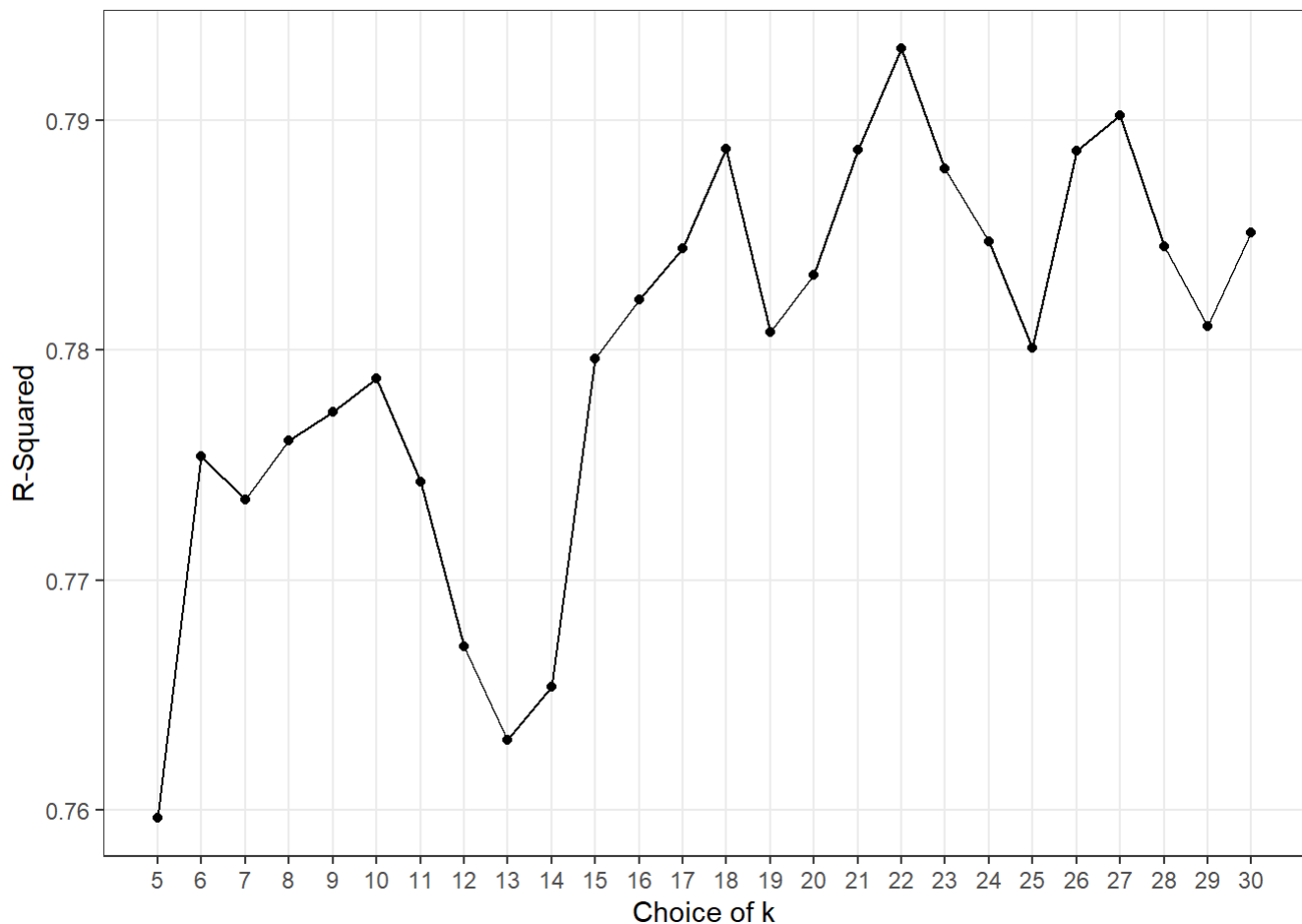
  R2_k <- c(R2_k, cor(knn_temp$pred, country_test3$Score)^2)
}

# Create the Line graph
data.frame(k = k_choice,
           R2 = R2_k) %>%

  ggplot(mapping = aes(x = k,
                       y = R2)) +
  geom_point() +
  geom_line() +

  labs(x = "Choice of k",
       y = "R-Squared") +

  scale_x_continuous(breaks = k_choice,
                     minor_breaks = NULL) +
  scale_y_continuous(minor_breaks = NULL)
```



Part 3d) KNN: Predicted vs Actual

```
# correlation between the predicted and true values
knn27_pred <- knn.reg(train = country_train3 %>% dplyr::select(-Score),
                      test = country_test3 %>% dplyr::select(-Score),
                      y = country_train3$Score,
                      k = 22)$pred
```

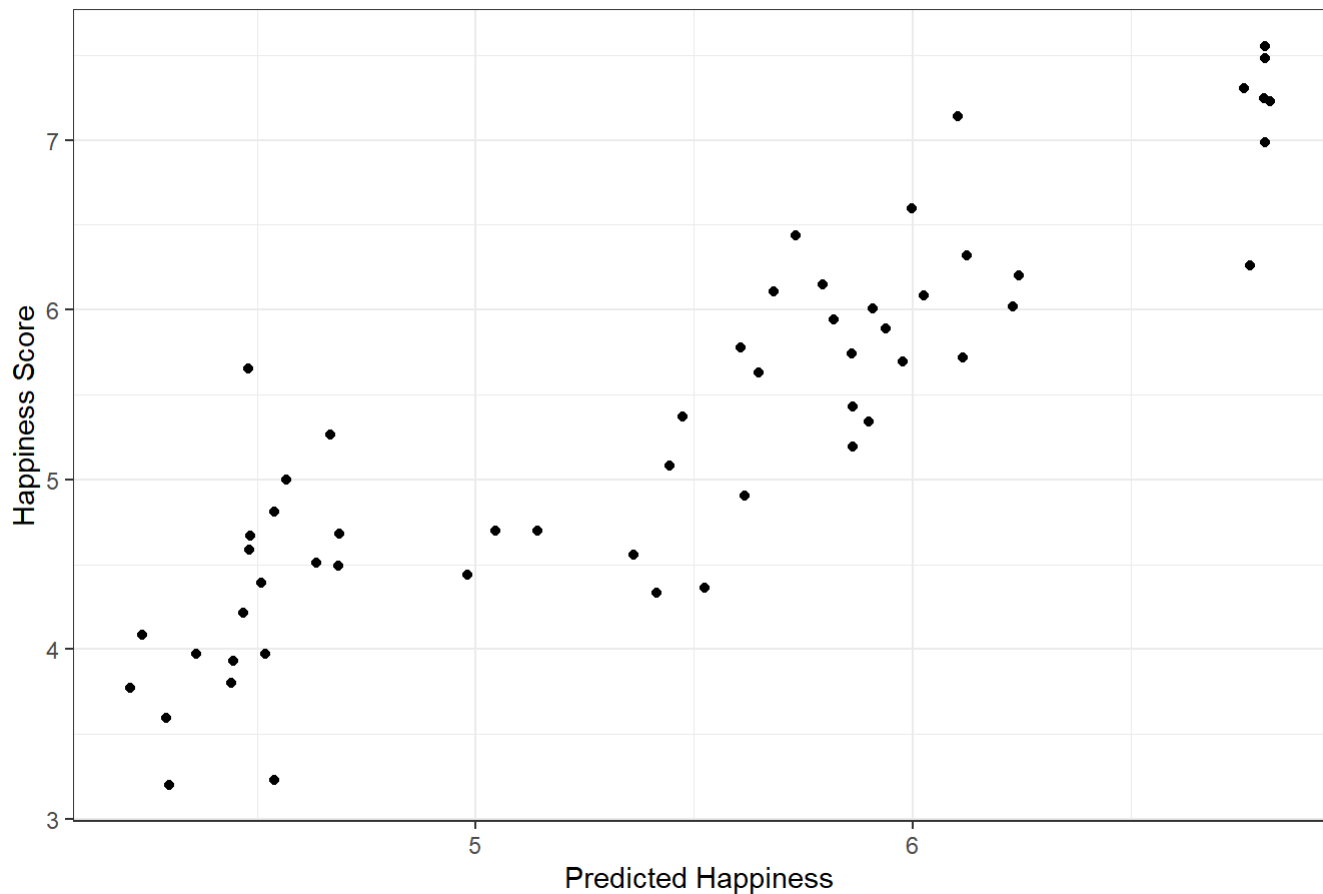
Part 3d i) Scatterplot Actual vs Predict

```
data.frame(country_test,
            knn_pred = knn27_pred) %>%

ggplot(mapping = aes(x = knn_pred,
                     y = Score)) +
geom_point() +

labs(y = "Happiness Score",
     x = "Predicted Happiness",
     title = "Actual vs Predicted Happiness with KNN")
```

Actual vs Predicted Happiness with KNN



The scatterplot shows the predicted happiness scores are fairly close to the actual happiness scores of the 56 test countries, but might be able to be improved!

Part 3d ii) Calculate the summary statistics for actual and predicted

```
# Adding the predictions for knn with k = 27 in the same data set as the Actual Score
data.frame(country_test,
            knn_pred = knn27_pred) %>%

# Calculating the 5 number summary for both variables using quantile
summarize(Actual = quantile(Score, probs = 0:4/4),
          Predicted = quantile(knn_pred, probs = 0:4/4)) %>%

mutate(Actual = round(Actual, digits = 2),
       Predicted = round(Predicted, digits = 2)) %>%

# Adding a column to indicate what the summary is
add_column(Summary = c("Min", "Q1", "Median", "Q3", "Max"),
           .before = "Actual") # Moving the Summary column before the Actual Column
```

Summary <chr>	Actual <dbl>	Predicted <dbl>
Min	3.20	4.21
Q1	4.48	4.54

Summary <chr>	Actual <dbl>	Predicted <dbl>
Median	5.30	5.56
Q3	6.09	5.98
Max	7.55	6.82
5 rows		

Part 3d iii) Correlation between Actual and Predicted

```
# Calculate and save correlation
cor_knn <-
  cor(country_test$Score,
      knn27_pred)

# Correlation and R2
c("r" = cor_knn,
  "R2" = cor_knn^2) %>%
  round(digits = 3)
```

```
##      r      R2
## 0.891 0.793
```

The correlation and R-squared show that KNN Regression predictions are closely related to the actual values, indicating accurate predictions!

Part 3e) KNN: Mean Absolute Error

```
# mean absolute error of predicted and true values
MAE_knn <- MAE(country_test$Score,
               knn27_pred)

MAE_knn_improve <- MAE_ybar - MAE_knn

c(multireg = MAE_knn,
  average = MAE_ybar,
  improvement = MAE_knn_improve,
  improve_percent = MAE_knn_improve/MAE_ybar*100)
```

```
##      multireg      average      improvement      improve_percent
##      0.4435909      0.9440536      0.5004627      53.0121041
```

Using KNN regression, the predictions are off by an average of 0.444. Happiness scores would have been off by 0.944 if we'd just used the mean happiness score, so the method improves MAE by 53.

Conclusions

```

model_comparison <-
  rbind(multireg = c(cor_multireg,
                    MAE_mlr,
                    MAE_mlr_improve),

        regtree = c(cor_regtree,
                    MAE_regtree,
                    MAE_regtree_improve),

        knn = c(cor_knn,
                MAE_knn,
                MAE_knn_improve)) %>%
  round(digits = 3)

colnames(model_comparison) <- c("Correlation", "MAE", "MAE_Improvement")
model_comparison

```

##	Correlation	MAE	MAE_Improvement
## multireg	0.886	0.401	0.516
## regtree	0.853	0.482	0.462
## knn	0.891	0.444	0.500

(answers may vary!)

KNN Regression has the highest correlation between the predicted scores and actual scores, but just slightly higher than multiple regression, while **Multiple Linear Regression** has the lowest *mean absolute error* by about 0.065.

For people familiar with regression analysis, Multiple Linear Regression would probably be the best. Judging the methods using MAE, the predictions are more accurate than the other two methods and the correlation is only slightly lower. In addition, MLR gives a model that can be used to understand the relationship between Happiness Score and the predictor variables.

For people not as familiar with regression, the regression tree might be useful due to its simplicity without sacrificing that much accuracy compared to the other 2 models.