# EthicalInvesting.R

santiagocassalett

2022-07-08

```
#This script explores the question of whether ethical investments lead
# to better returns based on ESG scores

# Here I will open and explore two .csv files
# Funds_info.csv contains basic information about EFTs and mutual funds
# Funds_ESG_ratings.csv contains the corresponding ESG rating and their
# risk of exposure to a list of controversial topics.

# Addressing questions such as:
# 1) Do ethical investments lead to better returns?
# 2) Can we actually predict returns based on these ratings?
# 3) What do these ESG scores reflect in terms of the fund's involvement in certain
# controversial issues?

# The script is split into three main sections:
# 1) Data Exploration
# 2) Modelling
# 3) Discussion

#The data exploration section examines the structure
# of the data and explores the relationships between the variables.
# Ideas of which variables to model will come from this data exploration

# The models sections aims to address the questions of whether
# there is a relationship between the different ESG scores
# and the fund returns.

# It also then goes on to test whether you can predict the returns
# based off of the scores

# The discussion section goes into a discussion of the results

############################
######## PACKAGES ########
############################

#Loading the various packages that I will be using
# to address the above questions

#Tidyverse is a wonderful package for DS - dplyr, ggplot2, and reader will be used
library(tidyverse)
```

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

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

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

```
#ggrepel allows you to plot labels in a more elegant fashion that standard ggplot2
library(ggrepel)

#lattice allows plotting multiple plots at the same time
library(lattice)

#grid, gridExtra,and GGally all improve upon ggplot styles/plotting
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
#Tidymodels is an excellent package for model deployments in R
# that uses the same tidy language as tidyverse
library(tidymodels)
```

```
## -- Attaching packages --------------------------------- tidymodels 0.2.0 --

## v broom        0.8.0     v rsample      0.1.1
## v dials        0.1.1     v tune         0.2.0
## v infer        1.0.0     v workflows    0.2.6
## v modeldata    0.1.1     v workflowsets 0.2.1
## v parsnip      0.2.1     v yardstick    0.0.9
## v recipes      0.2.0

## -- Conflicts ------------------------------------ tidymodels_conflicts() --
## x gridExtra::combine() masks dplyr::combine()
## 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
```

```
#Corrr library is part of the tidy universe and allows for coorelation plots
library(corrr)

#Glmnet is needed in order to run the linear model with penalty terms
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
#ranger is needed in order to run the random forests model
library(ranger)

################################
############ DATA #############
################################

#Here I import the data from the two .csv.gz files:

#First, I import the the Funds_info.csv.gz using readr
# into the variable: fundsInfo

#fundsInfo contains 56606 rows and 15 columns
fundsInfo <- read_csv("Funds_info.csv.gz")
```

```
## Rows: 56606 Columns: 15
```

```
## -- Column specification --------------------------------------------------
## Delimiter: ","
## chr (7): ticker, fund_name, fund_type, category, investment_strategy,...
## dbl (8): fund_return_2019, fund_return_2018, fund_return_2017, risk_r...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#Looking at the column names
colnames(fundsInfo)
```

```
##  [1] "ticker"              "fund_name"           "fund_type"
##  [4] "category"            "investment_strategy" "top5_holdings"
##  [7] "fund_return_2019"    "fund_return_2018"    "fund_return_2017"
## [10] "risk_rating"         "performance_rating"  "asset_stock"
## [13] "asset_bond"          "asset_cash"          "country_exposure"
```

```
#Quick check of the data in the various columns
glimpse(fundsInfo)
```

```
## Rows: 56,606
## Columns: 15
## $ ticker              <chr> "F00000VD0E", "F00000P42D", "F0000108NU", "~
## $ fund_name           <chr> "Valu-Trac Investment Funds ICVC Equity Inc~
## $ fund_type           <chr> "Mutual Fund", "Mutual Fund", "Mutual Fund"~
## $ category            <chr> "Global Equity Income", "Global Emerging Ma~
## $ investment_strategy <chr> "The objective of the Fund is to obtain a y~
## $ top5_holdings       <chr> "United Utilities Group PLC: 5.57, Verizon ~
## $ fund_return_2019    <dbl> 16.25, 3.56, -2.39, 5.71, -3.23, 19.20, 3.1~
## $ fund_return_2018    <dbl> -4.46, -6.68, NA, -5.82, 1.81, -14.08, -8.0~
## $ fund_return_2017    <dbl> 6.78, 10.80, NA, NA, 7.35, NA, 11.63, NA, N~
## $ risk_rating         <dbl> 3, 3, NA, 1, 3, 4, 5, 5, 3, NA, NA, 4, NA, ~
## $ performance_rating  <dbl> 2, 3, NA, 1, 3, 5, 2, 3, 4, NA, NA, 3, NA, ~
## $ asset_stock         <dbl> 98.72, 0.00, -24.68, 37.45, 0.00, 96.76, 0.~
## $ asset_bond          <dbl> 0.00, 96.15, 34.14, 39.42, 96.51, 0.00, 110~
## $ asset_cash          <dbl> 1.28, 3.51, 90.31, 2.26, 3.49, 3.24, -10.33~
## $ country_exposure    <chr> "USA: 35.37, CAN: 19.19, GBR: 18.03, SGP: 6~
```

```
#Quick check of duplicate rows - either in the ticker or fund name category
length(unique(fundsInfo$ticker))
```

```
## [1] 56606
```

```
length(unique(fundsInfo$fund_name))
```

```
## [1] 56606
```

```
#No duplicates found

#Second, I import the Funds_ESG_rating.csv.gz into the variable:

#fundsESG contains 30945 rows and 17 columns
fundsESG <- read_csv("Funds_ESG_ratings.csv.gz")
```

```
## Rows: 30945 Columns: 17
## -- Column specification --------------------------------------------------
## Delimiter: ","
## chr  (1): ticker
## dbl (16): environmental_score, social_score, governance_score, involv...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
#Looking at the column names
colnames(fundsESG)
```

```
##  [1] "ticker"
##  [2] "environmental_score"
##  [3] "social_score"
##  [4] "governance_score"
##  [5] "involvement_abortive_contraceptive"
##  [6] "involvement_alcohol"
##  [7] "involvement_animal_testing"
##  [8] "involvement_controversial_weapons"
##  [9] "involvement_gambling"
## [10] "involvement_gmo"
## [11] "involvement_military_contracting"
## [12] "involvement_nuclear"
## [13] "involvement_palm_oil"
## [14] "involvement_pesticides"
## [15] "involvement_small_arms"
## [16] "involvement_thermal_coal"
## [17] "involvement_tobacco"
```

```r
#Quick check of the data in the various columns
glimpse(fundsESG)
```

```
## Rows: 30,945
## Columns: 17
## $ ticker                             <chr> "F000000KX9", "F000013IA0", ~
## $ environmental_score                <dbl> 0.2701877, 0.1676124, 0.1920~
## $ social_score                       <dbl> 0.4482360, 0.5089647, 0.5164~
## $ governance_score                   <dbl> 0.4386617, 0.5122677, 0.5828~
## $ involvement_abortive_contraceptive <dbl> 2.20, 8.36, 12.44, 4.46, 12.~
## $ involvement_alcohol                <dbl> 0.00, 0.00, 3.64, 0.00, 0.00~
## $ involvement_animal_testing         <dbl> 9.99, 25.77, 31.50, 12.00, 3~
## $ involvement_controversial_weapons  <dbl> 0.00, 0.00, 1.18, 1.73, 2.27~
## $ involvement_gambling               <dbl> 0.00, 0.00, 0.20, 1.01, 0.00~
## $ involvement_gmo                    <dbl> 0.00, 0.00, 0.00, 0.00, 0.00~
## $ involvement_military_contracting   <dbl> 0.00, 1.10, 1.45, 2.22, 3.46~
## $ involvement_nuclear                <dbl> 0.00, 0.55, 2.88, 2.72, 2.62~
## $ involvement_palm_oil               <dbl> 0.00, 0.00, 0.00, 0.00, 0.00~
## $ involvement_pesticides             <dbl> 0.00, 0.00, 1.71, 0.97, 0.00~
## $ involvement_small_arms             <dbl> 0.35, 0.01, 1.18, 0.00, 0.81~
## $ involvement_thermal_coal           <dbl> 3.91, 1.30, 2.16, 1.93, 3.32~
## $ involvement_tobacco                <dbl> 0.01, 0.01, 0.22, 0.00, 1.38~
```

```r
#Quick check of duplicate rows
length(unique(fundsESG$ticker))
```

```
## [1] 30945
```

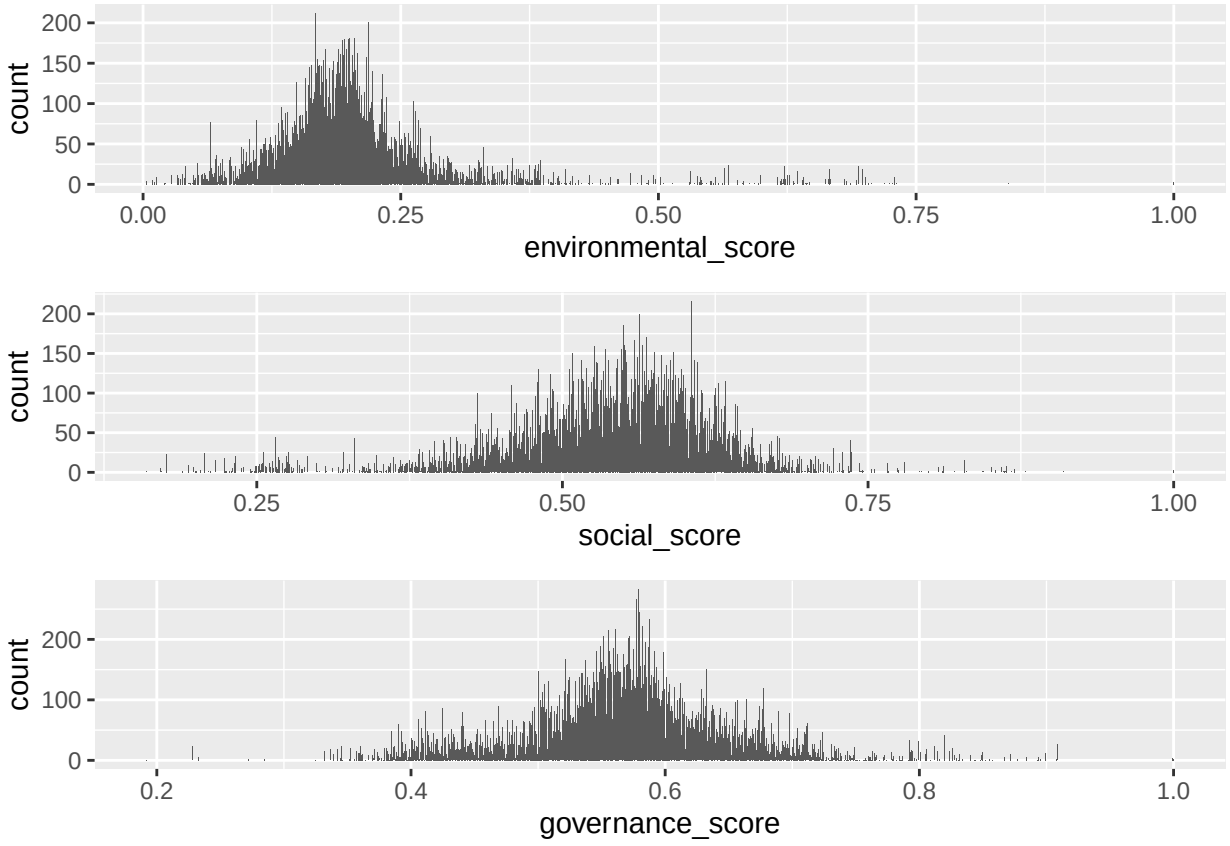
```r
#No duplicates found
```

```r
#ticker appears to the unique key to join together the two datasets
# there are more companies in the fundsInfo dataset than there
# are in the fundsESG set - so when I bind together the
# two dataset, I expect to find the dataset to be 30,945 rows

#From examining the different variables:
# 1) Need to join the two datasets on ticker
# 2) Will exclude (for now) the investment_strategy
# 3) Will keep all of the variables in fundsESG for exploration

##############################################
######### DATA CLEANING AND MERGING #########
##############################################

#Here I join together the two datasets - those "ticker" values in
# fundsInfo that do not have a matching ticker in fundsESG will be
# ignored - without the ESG ratings there is nothing to work on

ESG_and_info <- fundsESG %>%
  left_join(fundsInfo, by = "ticker") %>%
  select(-investment_strategy)

#Checking for NAs in the dataset
# What I find is that there are 13,757 rows with at least one NA
ESG_and_info %>%
  filter(if_any(everything(), ~is.na(.)))
```

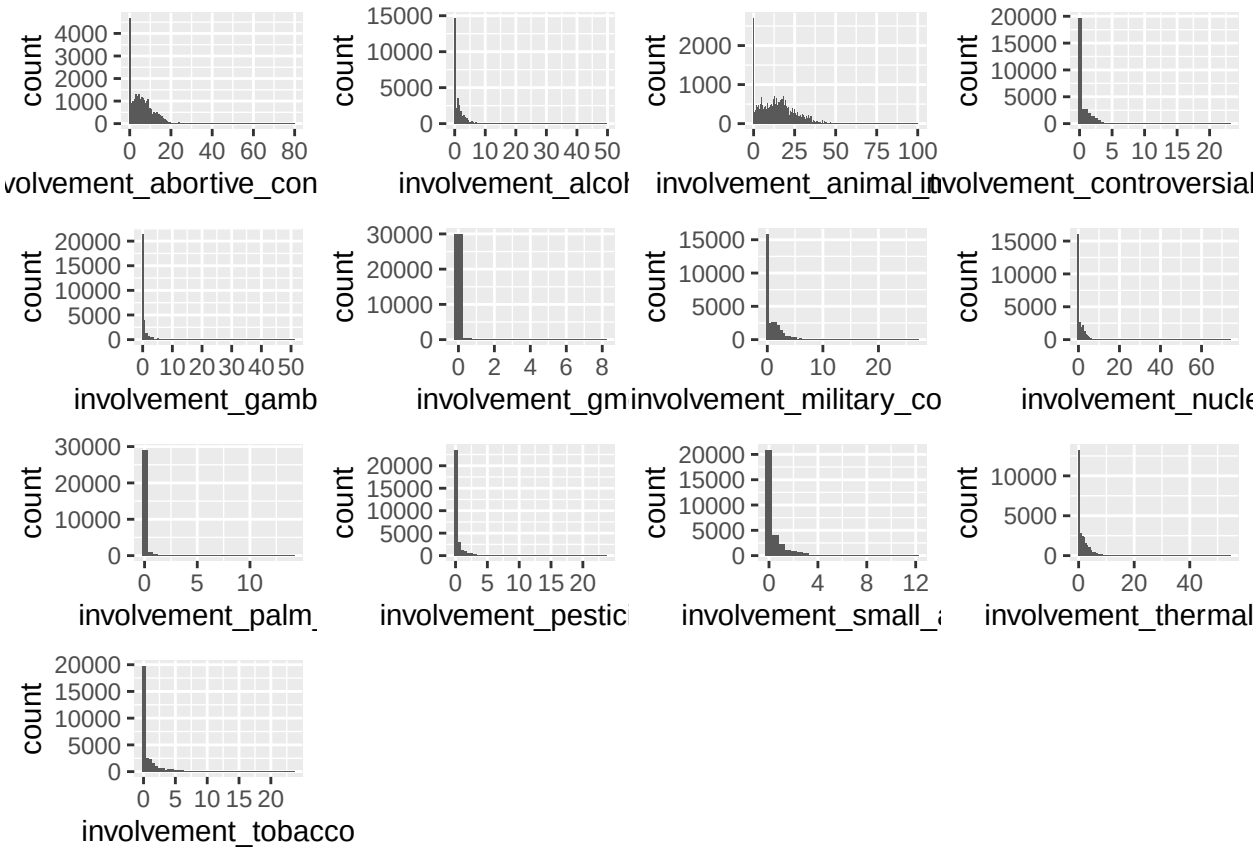
```
## # A tibble: 13,757 x 30
##    ticker environmental_s~ social_score governance_score involvement_abo~
##    <chr>             <dbl>        <dbl>            <dbl>            <dbl>
##  1 F0000~           0.168         0.509            0.512             8.36
##  2 F0000~           0.205         0.539            0.569             4.46
##  3 F0000~           0.192         0.453            0.555             2.38
##  4 F0000~           0.132         0.492            0.523             9.46
##  5 F0000~           0.174         0.566            0.503            10.6
##  6 0P000~           0.0279        0.910            0.598            22.9
##  7 F0000~           0.268         0.450            0.442             2.35
##  8 F0000~           0.105         0.612            0.625             7.48
##  9 F0000~           0.216         0.617            0.698             2.82
## 10 F0000~           0.213         0.447            0.532             0
## # ... with 13,747 more rows, and 25 more variables:
## #   involvement_alcohol <dbl>, involvement_animal_testing <dbl>,
## #   involvement_controversial_weapons <dbl>, involvement_gambling <dbl>,
## #   involvement_gmo <dbl>, involvement_military_contracting <dbl>,
## #   involvement_nuclear <dbl>, involvement_palm_oil <dbl>,
## #   involvement_pesticides <dbl>, involvement_small_arms <dbl>,
## #   involvement_thermal_coal <dbl>, involvement_tobacco <dbl>, ...
```

```r
#Most of the NAs are coming from the fund_return_2017 columns
# as well as the risk_rating and performance_rating columns

#The choice now is to fill the NAs using the median averages
# around them or to fill them using k Nearest Neighbors or to
```

```r
# simply drop the values

# For the sake of time - I am going to combine together
# the three return values when I begin modeling and take
# the averages (median) of the three values to have a single
# "return" value. This removes the majority of the missing data
# values for returns. Those without values still will be dropped.
# This will obvious influence the results of the models
# but will still be informative of the overall relationships
# I am looking at.

##############################################
############## DATA EXPLORATION ##############
##############################################

#First step is I want to plot some of the variables that may be of most
# interest to check for variation within the variables and covaration between
# the variables

# Judging from the variables names - the different scores, involvements,
# risk_rating, performance_rating, and fund returns all may be important variables

#I begin by plotting the distribution of the score variables using barcharts

#Here I am grabbing the column names associated with "score" for plotting
ESGColnames_Score <- colnames(ESG_and_info %>%
                                select(contains("score")))

#This creates the plots for the scores - these are in bar plots
#it looks through the colnames and then outputs each into one output screen grid
plot_list <- list()
for(i in ESGColnames_Score){
  plot <- ggplot(ESG_and_info, aes_string(x = i)) +
    geom_bar()
  plot_list[[i]] <- plot
}

plot_grob <- arrangeGrob(grobs = plot_list)
grid.arrange(plot_grob)
```



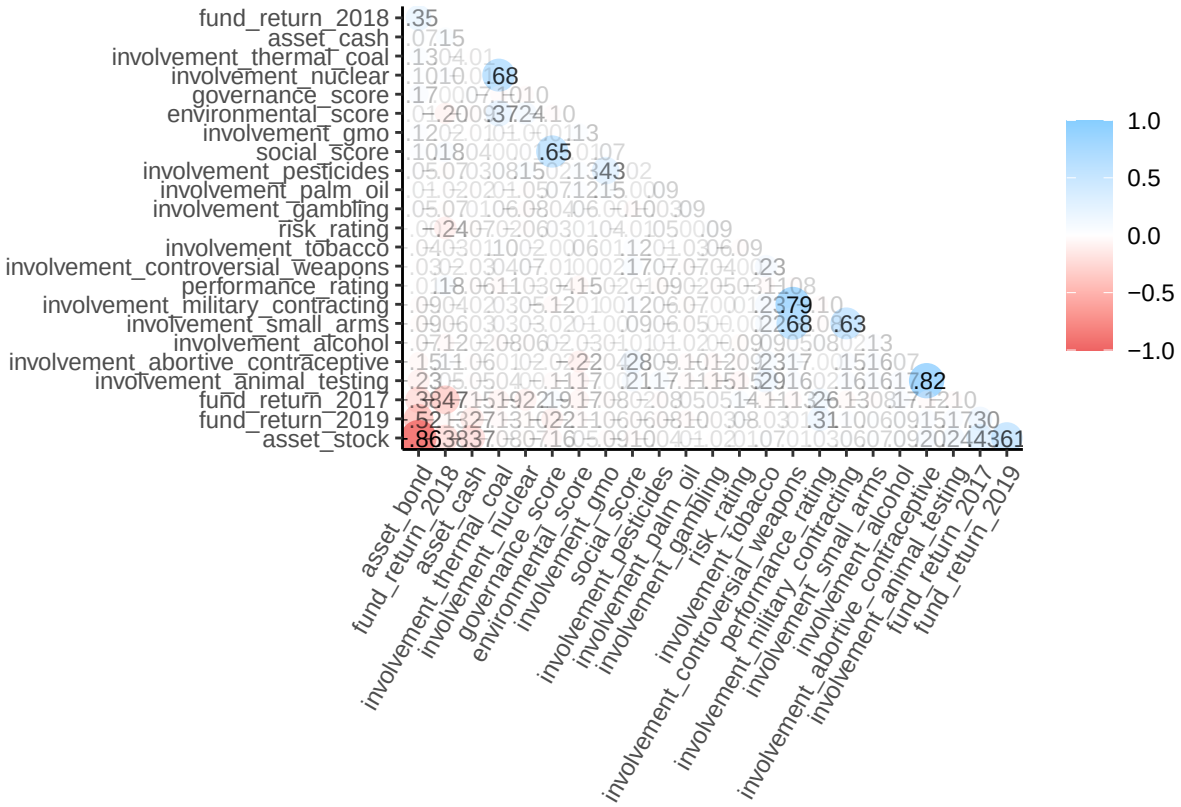

```
#What we see is that most values for environmental_score center around 0.20
# while for social_score and governance_score, they center around 0.60
# This suggests that most companies score relatively poorly on social
# and governance while pretty good on environmental score

# These variables have a bell-curve distribution but environmental
# is skewed towards the left and the other are centered.


#Next I will plot the distribution of the involvements
#Here I am grabbing the column names associated with "involvement" for plotting
ESGColnames_Involvement <- colnames(ESG_and_info %>%
                                      select(contains("involvement")))

#This creates the plots for involvements - these are histograms
# same as above but with histograms
plot_list <- list()
for(i in ESGColnames_Involvement){
  plot <- ggplot(data = ESG_and_info,
                 aes_string(x = i)) +
    geom_histogram(binwidth = .5)
  plot_list[[i]] <- plot
}
plot_grob <- arrangeGrob(grobs = plot_list)
grid.arrange(plot_grob)
```




```
#What we see is that most of the involvement variables do not provide very much
# information - abortive_contraceptive, animal_test, military contracting, nuclear,
# small_arms, thermal_coal, and tobacco may by the ones that have the highest
# involvement and be the most useful for predictions

#The distribution of these involvements are heavily skewed towards zero or
# with a positive skew - these potentially zero-inflated variables
# are largely a consequences of the funds not reporting or claiming
# to have no involvements with these categories. Linear models without
# a possion function may have issues modeling these variables.

################################
######## CORRELATIONS ########
################################

#Another thing to examine is the correlations between some of the variables
# for example - we can examine the correlations between the different scores

#rplot provides a useful was to view correlations between all
# of the variables - not just the ones Ive choosen as being potentially most useful

rplot(ESG_and_info %>%
        select(where(is.numeric)) %>%
        correlate() %>%
        rearrange(absolute = F) %>%
        shave(), print_cor = T) +
```

```
  theme(axis.text.x = element_text(angle = 60, hjust = 1))
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
##
## Don't know how to automatically pick scale for object of type noquote. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type noquote. Defaulting to continuous.
```




```
#What we see is that environmental scores are not correlated or very weakly
#correlated (-0.01) with social or governance scores
# Social and governance however are fairly highly correlated (0.65)
# We also see that there are correlations among the involvements.
# Military_contracting and controversial_weapons are correlated (0.79),
# as are controversial_weapons and small_arms (0.68).
# The highest correlation though is between animal_testing and
# abortive_contraceptive (0.82).

#The ESG scores and these various involvements though all have low correlations
# suggesting that there is not a strong relationship between them

#This goes to question #3 - the ESG scores seems to be largely not tightly
# correlated with the fund's involvements in convertroversial issues. However,
# further analysis is obviously needed.
```

```r
# An additional variable of potential interest are the different asset classes.
# You see a strong negative correlation between asset_bond and asset_stock.
# Also, bonds are associated with negative returns in 2017 and 2019 but positive
# return in 2018. Stocks are positively associated with returns in 2017 and 2019
# but not in 2018. Cash had lower returns in 2017 and 2019 but not as low as
# the other two asset types.

# Moving forward though, I am most interested in examining how
# the various ESG scores and the involvements influence returns and
# whether we can use these variables to predict the potential returns.

# Additional avenues to explore would be to model whether you can
# predict the different scores based on the involvements.

# Further, examining the relationships between the various risk_ratings
# and performance_ratings to the ESG scores would provide a way of highlighting
# whether those funds that are more "ethical" are better performing.

# For this assessment, however, I will use the scores and involvements to
# examine fund returns.

######################################################################
######################## MODELING SECTION ############################
######################################################################

#Below I select just the columns that I am aiming to use for the analyses
# Here the predictor variables will be the scores and involvements, and the
# response variable will be the fund returns

reducedDataset <- ESG_and_info %>%
  select(contains("score") | contains("involvement") | contains("fund_return"))

#Here I averaged the yearly returns and created the new column returns
# I then removed all of the NA values that are left.
#I am left with a dataset of 26,222 values

furtherReduced <- reducedDataset %>%
  mutate(returns = rowMeans(select(., contains("return")), na.rm = T)) %>%
  select(-contains("fund_return")) %>%
  filter(!is.na(returns))

#Preprocessing of the data is listed with the models in the recipe stage.

#Model 1) Linear Regression
# I begin the modeling with a linear regression as they
# often have a good predictive ability and are a simple model for
# providing interpretations of the variables.

#The large sample size (26,222) suggests that many of the
# assumptions of normality needed for a linear model
# may be already met (central limit theorem).
```

```
# This original model will also include all of the scores and
# all of the involvements that have a correlation score below 0.8.
# 0.8 was chosen as that is a very high correlation value.

#Model 2) Linear Regression with regularization penalites
# The second model is similar to the first but with the additional of
# regularization penalties that punish models with too many variables.
# The additional of the regularization (either lasso or ridge) has the potential
# to improve the model.

# Model 3) Random Forests
# The third model is a more complex, random forests model. Random forests
# have been found to have high predictive ability but lose interpretability.


#####################################################################
###################  LINEAR REGRESSION MODEL ########################
#####################################################################

#Now setting up the first model which is just a Linear Model

set.seed(25)

#split is for dividing the dataset into a training and test set
# strata makes the makes sure that the variables are evenally distributed
# across the splits - the split is 80% training and 20% test

split <- initial_split(furtherReduced, prop = 0.8, strata = returns)

train_data <- training(split)

test_data <- testing(split)

#For tidymodels - you can prep a "recipe" that will use for the pre-processing
# The pre-processing that is needed is the removal of the high correlation
# valued variables and to normalize the predictors (mean = 0 and variation = 1).
prep_recipe <-
  #First you set up the model
  recipe(returns ~ ., data = train_data) %>%
  #Next I remove the correlations greater than a threshold - here is 0.8
  step_corr(all_numeric(), threshold = 0.8) %>%
  #Next I normalize the predictor variables minus the variables that are categorical
  step_normalize(all_predictors(), -all_nominal())

prep(prep_recipe)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         16
```

```
##
## Training data contained 20976 data points and no missing data.
##
## Operations:
##
## Correlation filter on involvement_animal_testing [trained]
## Centering and scaling for environmental_score, social_score, gover... [trained]
```

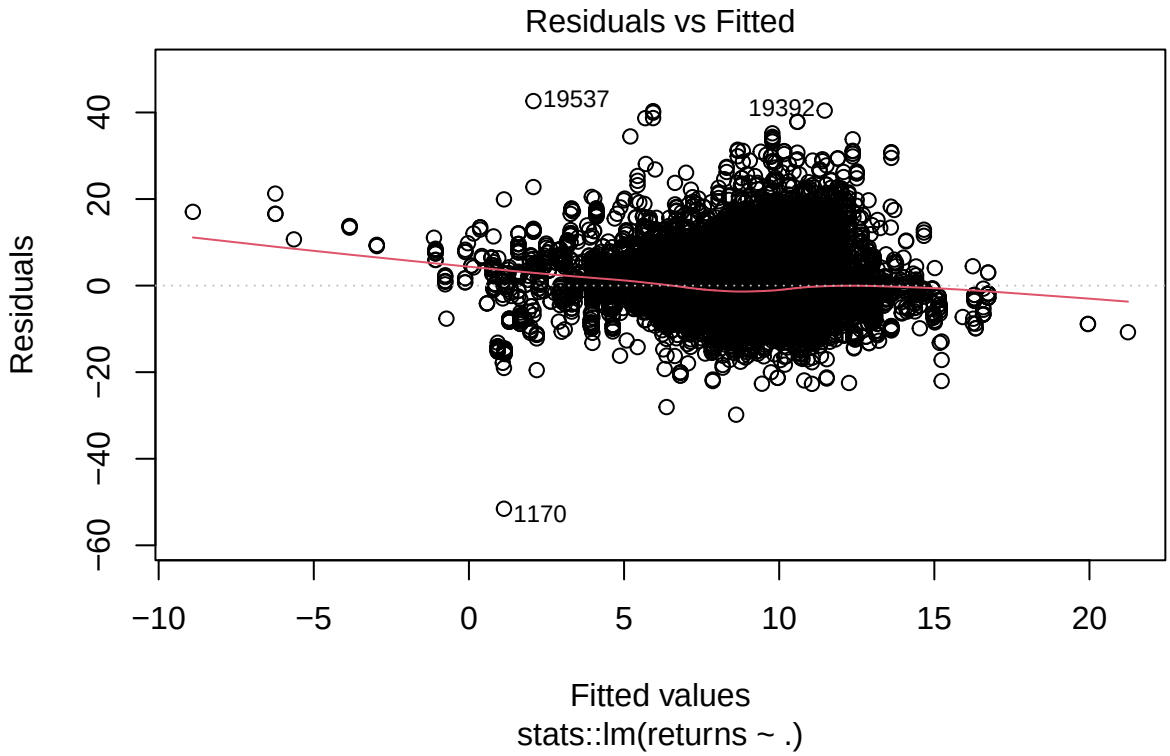
```
#Here I "juice" the recipe - essentially applying the steps above to the dataset
dia_juiced <- juice(prep(prep_recipe))

#Examine the dimensions of the data
dim(dia_juiced)
```

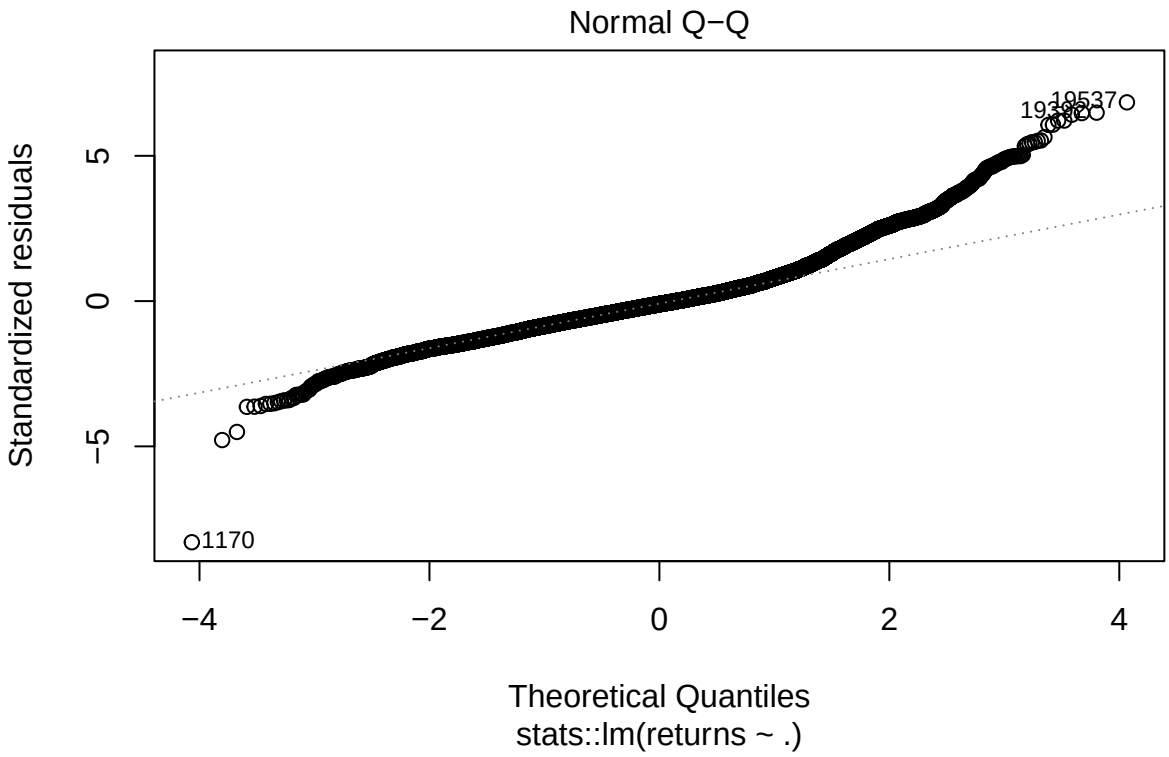
```
## [1] 20976    16
```

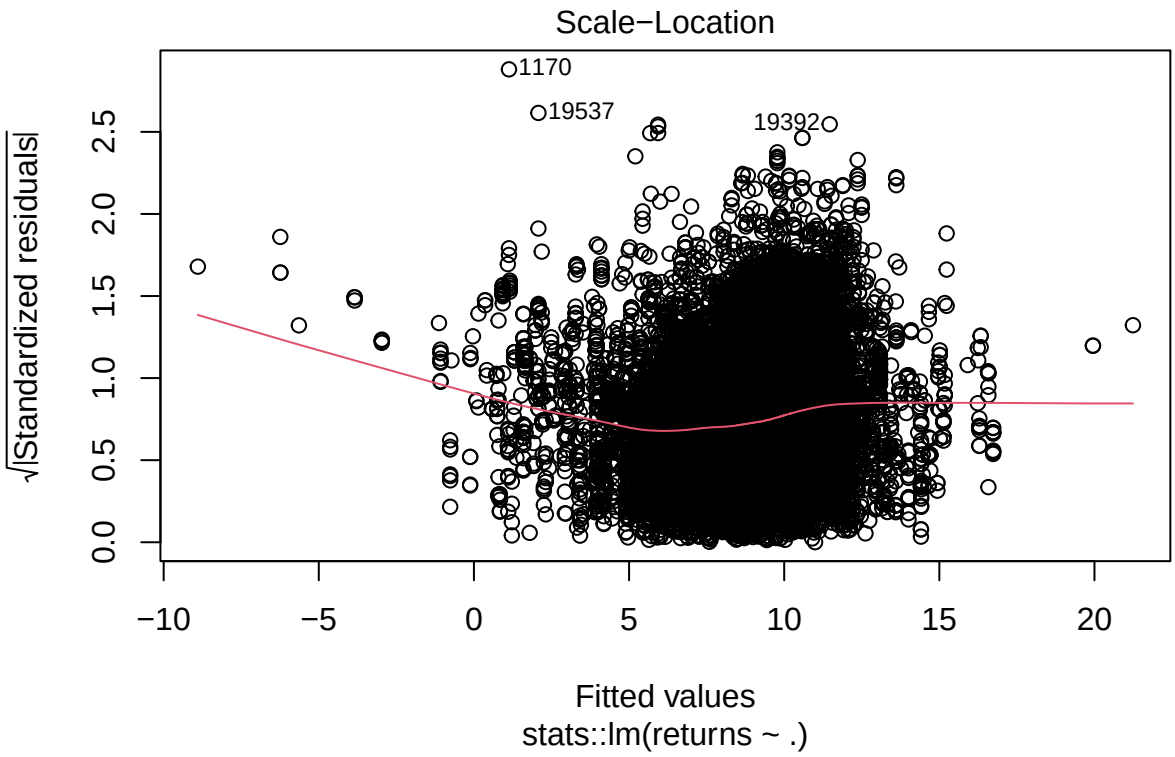
```
#Here I am using the basic linear regression model with the
# base engine (lm) - I am initializing the model below

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

#Fitting the juiced dataset to the model using all variables
lm_fit1 <- fit(lm_model, returns ~.,dia_juiced)
lm_fit1
```

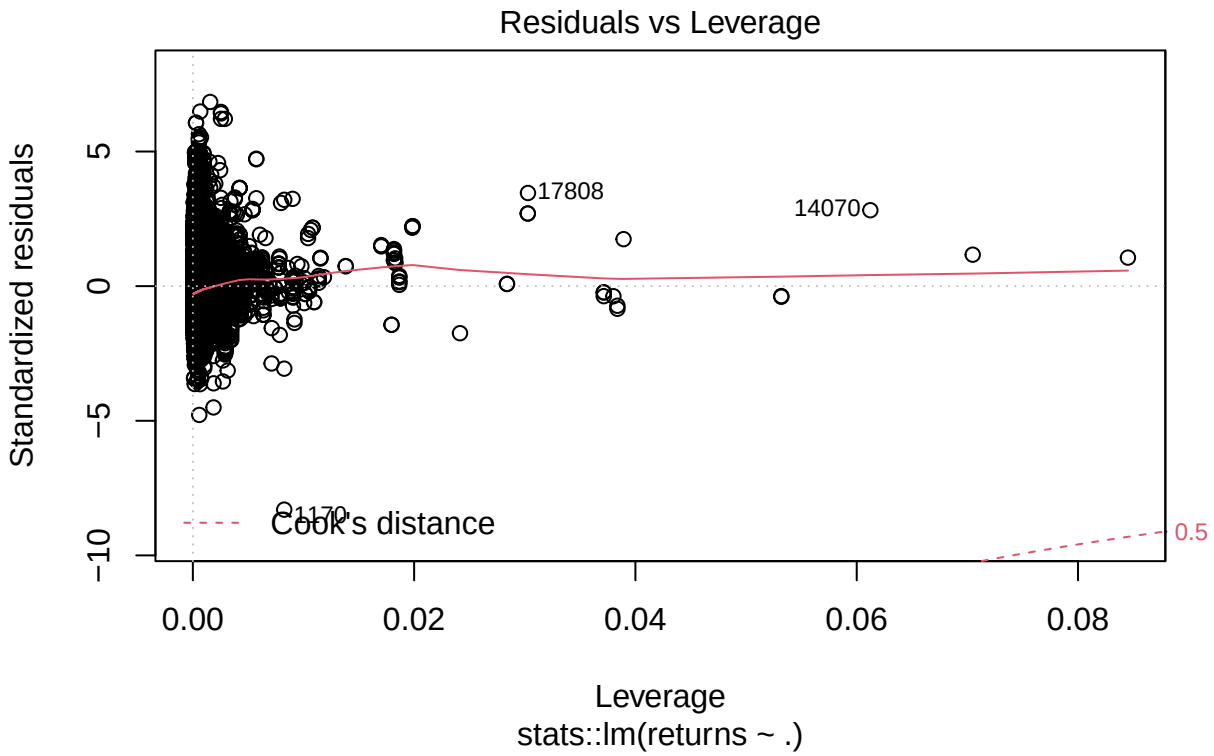
```
## parsnip model object
##
##
## Call:
## stats::lm(formula = returns ~ ., data = data)
##
## Coefficients:
##                          (Intercept)                 environmental_score
##                              8.73704                             -1.20503
##                         social_score                    governance_score
##                              0.84302                             -1.17320
## involvement_abortive_contraceptive                 involvement_alcohol
##                              0.14992                              0.52447
##  involvement_controversial_weapons                involvement_gambling
##                             -0.73106                             -0.22149
##                     involvement_gmo    involvement_military_contracting
##                             -0.15519                              0.29508
##                involvement_nuclear                involvement_palm_oil
##                             -0.24875                             -0.04101
##             involvement_pesticides              involvement_small_arms
##                             -0.49972                              0.13070
##           involvement_thermal_coal                 involvement_tobacco
##                             -0.27072                             -0.65246
```

```
#Plot to examine the assumptions of the linear model - diagnostic plots
plot(lm_fit1$fit)
```




Normal Q−Q
Standardized residuals
5
0
−5
19537
1170
−4
−2
0
2
4
Theoretical Quantiles
stats::lm(returns ~ .)


Scale−Location
1170
19537
19392
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
−10
−5
0
5
10
15
20
Fitted values
stats::lm(returns ~ .)




```
#What we see is that by including all of the variables.
# You have a rather terrible model.
# You have a very high AIC which is terrible, a RMSE which is also not
# good (6.25) and we explain little of the variance r.squared (0.0934).
# Interested to see how we can improve this.

#Glance provides a tidy way to view the fit,
# provides the r.squared, RMSE, p value, and AIC.
# We have a significant model but it does not explain much

glance(lm_fit1$fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df  logLik     AIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>   <dbl>   <dbl>
## 1    0.0983        0.0976  6.24      152.       0    15 -68153. 136339.
## # ... with 4 more variables: BIC <dbl>, deviance <dbl>,
## #   df.residual <int>, nobs <int>
```

```
#Tidy provides a way to examine the importance of the variables
tidy(lm_fit1) %>%
  arrange(desc(abs(statistic)))
```

```
## # A tibble: 16 x 5
##    term                          estimate std.error statistic   p.value
```

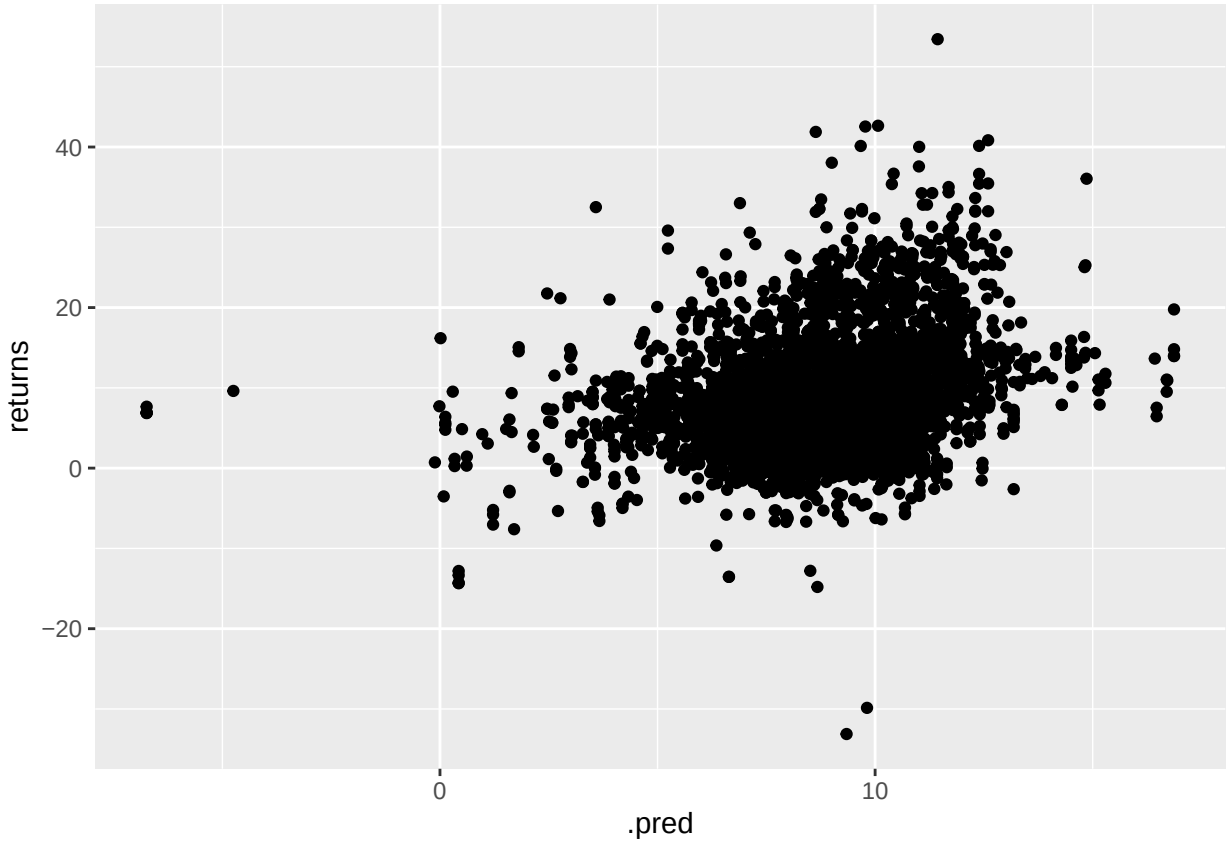
```
##    <chr>                              <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)                        8.74      0.0431   203.    0
##  2 environmental_score               -1.21      0.0495   -24.4   3.28e-129
##  3 governance_score                  -1.17      0.0623   -18.8   1.41e- 78
##  4 involvement_tobacco               -0.652     0.0460   -14.2   2.12e- 45
##  5 social_score                       0.843     0.0642    13.1   2.85e- 39
##  6 involvement_alcohol                0.524     0.0440    11.9   1.13e- 32
##  7 involvement_pesticides            -0.500     0.0486   -10.3   8.71e- 25
##  8 involvement_controversial_weap~   -0.731     0.0777    -9.41  5.23e- 21
##  9 involvement_gambling              -0.221     0.0444    -4.99  6.22e-  7
## 10 involvement_thermal_coal          -0.271     0.0625    -4.33  1.47e-  5
## 11 involvement_nuclear               -0.249     0.0606    -4.11  4.05e-  5
## 12 involvement_military_contracti~    0.295     0.0749     3.94  8.12e-  5
## 13 involvement_gmo                   -0.155     0.0477    -3.25  1.15e-  3
## 14 involvement_abortive_contracep~    0.150     0.0511     2.93  3.37e-  3
## 15 involvement_small_arms             0.131     0.0611     2.14  3.24e-  2
## 16 involvement_palm_oil              -0.0410    0.0441    -0.929 3.53e-  1
```

```r
#The most important variables are environmental score and governance score
# which are both negative suggesting that the better the ESG score
# the better the return - the reverse appears true in terms of social score
# as social score increases - the return increases

#The results of this model would suggest that those funds with
# the lower environmental scores do have better returns, but
# that social_scores seems to be less important
# additionally - the fewer involvements with the controversial
# categories leads to higher returns except military_contracting,
#small_arms, and abortive_contraceptives potentially increase the returns.

#Now will prep the recipe for the test_data
prep_recipe <-
  #First you set up the model
  recipe(returns ~ ., data = test_data) %>%
  #Next I remove the correlations greater than a threshold - here is 0.8
  step_corr(all_numeric(), threshold = 0.8) %>%
  #Next I normalize the predictor variables minus the variables that are categorical
  step_normalize(all_predictors(), -all_nominal())

test_juiced <- juice(prep(prep_recipe))

#Here I try to predict the returns from the test_data set
lm_predict <- predict(lm_fit1, new_data = test_juiced)
lm_predict$returns <- test_data$returns

#The model is pretty poor - 6.15 RMSE, and an r.squared of 0.103
# It performs slightly better on the test_data set than on the training
# data set suggest that it is not overfit but the low r.squared also
# suggests that I missing much of the variance
metrics(lm_predict, truth = returns, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
```

```
## 1 rmse    standard      6.22
## 2 rsq     standard      0.0857
## 3 mae     standard      4.55
```

```
#Plot of the predicted versus the actual values for the returns
# What we see is essentially a large blob - there is a slight positive relationship
# but it is clear that the model is not capturing the data perfectly.
ggplot(data = lm_predict,
       aes(x = .pred, y = returns)) +
  geom_point()
```




```
#########################################################################
######### RERUNNING THE LM WITH PENALTY and RIDGE REGRESSION #########
#########################################################################

#Here I am attempting to run a linear model using the glmnet
# which allows for hyperparameter optimization
# This is an attempt to use feature selection/dimensonality reduction.

lm_model <-
  #Glmnet has two parameters that we will tune via the tune package
  #penalty is the same as a the lambda value - tunes how much of a penalty
  #mixture is the same as alpha and determines whether the model
  # is a pure ridge regression (0) or lasso (1) model
  linear_reg(penalty = tune(),
```

```
            mixture = tune()) %>%
  #mode is set as regression
  set_mode("regression") %>%
  #engine is set as glmnet
  #Requres the package glmnet be installed and loaded
  set_engine("glmnet")

#Here I am creating a workflow - funnels everything into a pipeline
LM_wflow <-
  #initializes the original workflow
  workflow() %>%
  #add the model - the glmnet model
  add_model(lm_model) %>%
  #adds the recipe that is the same the one used for the original LM
  add_recipe(prep_recipe)

#Provides information about the workflow and the steps that will be taken
# and the order
# The parameters have yet to be tuned
LM_wflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 2 Recipe Steps
##
## * step_corr()
## * step_normalize()
##
## -- Model -----------------------------------------------------------------------
## Linear Regression Model Specification (regression)
##
## Main Arguments:
##   penalty = tune()
##   mixture = tune()
##
## Computational engine: glmnet
```

```
#Here I am updating the parameters that will be used for the model.
#I am setting the search parameters for the grid itself.
# The grid will use 5 penalty terms and 5 mixture values
LM_param <-
  LM_wflow %>%
  parameters() %>%
  update(penalty = penalty(range = c(-3, 0)),
         mixture = mixture(range = c(0, 1)))
```

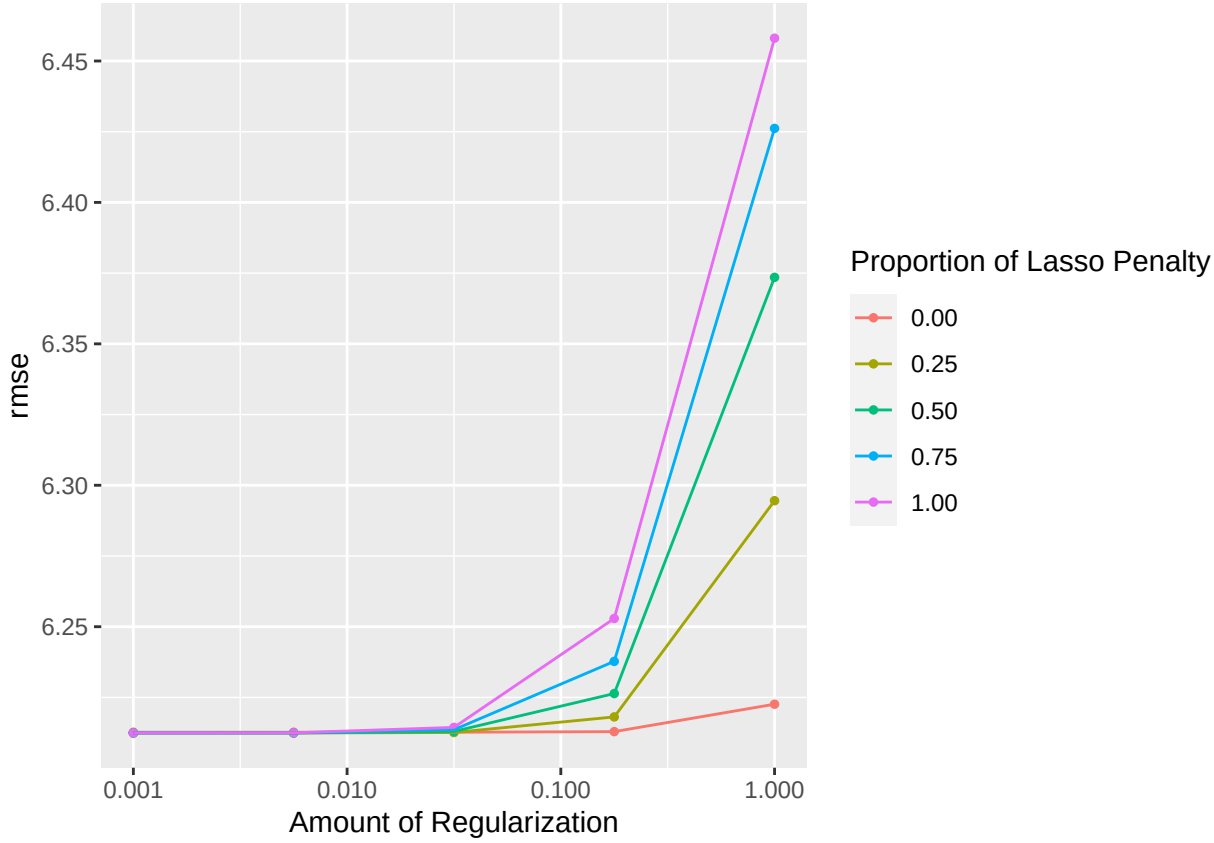
```
## Warning: `parameters.workflow()` was deprecated in tune 0.1.6.9003.
## Please use `hardhat::extract_parameter_set_dials()` instead.
```

```r
#Here I am setting the number of terms that will be used
LM_grid <- grid_regular(LM_param, levels = 5)


#Set a seed for reproducibility
set.seed(1559)

#Here I will be working with a reduced dataset again

furtherReduced
```

```
## # A tibble: 26,222 x 17
##    environmental_score social_score governance_score involvement_abortiv~
##                  <dbl>        <dbl>            <dbl>                <dbl>
##  1               0.270        0.448            0.439                 2.2
##  2               0.192        0.516            0.583                12.4
##  3               0.205        0.539            0.569                 4.46
##  4               0.210        0.586            0.544                12.8
##  5               0.195        0.506            0.477                 3.05
##  6               0.192        0.453            0.555                 2.38
##  7               0.202        0.533            0.572                 7.93
##  8               0.132        0.492            0.523                 9.46
##  9               0.163        0.631            0.590                 8.57
## 10               0.229        0.562            0.607                14.1
## # ... with 26,212 more rows, and 13 more variables:
## #   involvement_alcohol <dbl>, involvement_animal_testing <dbl>,
## #   involvement_controversial_weapons <dbl>, involvement_gambling <dbl>,
## #   involvement_gmo <dbl>, involvement_military_contracting <dbl>,
## #   involvement_nuclear <dbl>, involvement_palm_oil <dbl>,
## #   involvement_pesticides <dbl>, involvement_small_arms <dbl>,
## #   involvement_thermal_coal <dbl>, involvement_tobacco <dbl>, ...
```

```r
#Split is for dividing the dataset into a training and test set.
# Strata makes the makes sure that the variables are evenly distributed
# across the splits - the split is 80% training and 20% test.

split <- initial_split(furtherReduced, prop = 0.8, strata = returns)
train_data <- training(split)

test_data <- testing(split)

#Here I am using 10-fold cross-validation on the training data set
# for parameter tuning

train_cv <- vfold_cv(train_data, v = 10, strata = returns)

#Here I am tuning the grid - using the workflow and the
# 10 fold training dataset.

#A potentially better way to do the grid search would be
# via a Baysian grid search approach.

return_glmnet_grid <-
```

```
  tune_grid(LM_wflow, grid = LM_grid, resamples = train_cv,
            param_info = LM_param)

#What you can see with the autoplot is that as the you increase
# the parameterization towards a lasso regression - you have
# a steep increase in the rmse score - this suggests that it starts
# to drop out too many variables.

autoplot(return_glmnet_grid, metric = "rmse")
```




```
show_best(return_glmnet_grid, "rmse", n = 9)
```

```
## # A tibble: 9 x 8
##   penalty mixture .metric .estimator  mean     n std_err .config
##     <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 0.001      0.75 rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 2 0.001      0.25 rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 3 0.00562    0.25 rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 4 0.001      0.5  rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 5 0.001      1    rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 6 0.00562    0.5  rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 7 0.00562    0.75 rmse    standard    6.21    10  0.0239 Preprocessor1_M~
## 8 0.00562    1    rmse    standard    6.21    10  0.0238 Preprocessor1_M~
## 9 0.0316     0.25 rmse    standard    6.21    10  0.0237 Preprocessor1_M~
```

```r
#What we find is that the best model is one with a penalty
# term of 0.001, and a mixture of 0.75 - this is close to a
# lasso regression model - however at the penalty level of 0.001
# the lasso penalty is not really enforced

select_best(return_glmnet_grid, metric = "rmse")
```

```
## # A tibble: 1 x 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1   0.001    0.75 Preprocessor1_Model16
```

```r
#Another way to examine is by using the one that contributes the
# smallest std_error - here you have the same penalty but complete
# ridge regression (0) - the mean metric (rmse) is the same as the other
# model

select_by_one_std_err(return_glmnet_grid, penalty, mixture, metric = "rmse")
```

```
## # A tibble: 1 x 10
##   penalty mixture .metric .estimator  mean     n std_err .config    .best
##     <dbl>   <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>      <dbl>
## 1   0.001       0 rmse    standard    6.21    10  0.0237 Preproces~  6.21
## # ... with 1 more variable: .bound <dbl>
```

```r
#I will now use the hyperparameters that were selected to fit the
# whole training dataset rather than the cross validated
LM_params_final <- select_by_one_std_err(return_glmnet_grid, penalty, mixture,
                                         metric = "rmse")

#Finalize the workflow using the updated parameters
LM_wflow_final <- finalize_workflow(LM_wflow, LM_params_final)

#Now fitting the training data set
LM_wflow_final_fit <- fit(LM_wflow_final, data = train_data)

# I will now use the predict function on the test data
# to examine how the model performed
finalTest <- predict(LM_wflow_final_fit, new_data = test_data)

#Need to add in the actual returns data to the predicted model
# the finalTest has a .pred with the predictions -
# this add back in the returns to compare
finalTest$returns <- test_data$returns

metrics(finalTest, truth = returns, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard      6.32
## 2 rsq     standard      0.0899
## 3 mae     standard      4.57
```

```r
#When predicting with the final model - it is pretty terrible.
# RMSE = 6.32, rsq = 0.089, mae = 4.54.
# This is worse than running the model without the parameterizations.
# Potentially this is because certain variables were pushed to zero or close
# to zero because of the ridge regression - potentially too much so.

##########################################################
################ RANDOM FORESTS MODELS ###################
##########################################################

#Here I will attempt to use the same data to model via random forests

furtherReduced
```

```
## # A tibble: 26,222 x 17
##    environmental_score social_score governance_score involvement_abortiv~
##                  <dbl>        <dbl>            <dbl>                <dbl>
##  1               0.270        0.448            0.439                 2.2
##  2               0.192        0.516            0.583                12.4
##  3               0.205        0.539            0.569                 4.46
##  4               0.210        0.586            0.544                12.8
##  5               0.195        0.506            0.477                 3.05
##  6               0.192        0.453            0.555                 2.38
##  7               0.202        0.533            0.572                 7.93
##  8               0.132        0.492            0.523                 9.46
##  9               0.163        0.631            0.590                 8.57
## 10               0.229        0.562            0.607                14.1
## # ... with 26,212 more rows, and 13 more variables:
## #   involvement_alcohol <dbl>, involvement_animal_testing <dbl>,
## #   involvement_controversial_weapons <dbl>, involvement_gambling <dbl>,
## #   involvement_gmo <dbl>, involvement_military_contracting <dbl>,
## #   involvement_nuclear <dbl>, involvement_palm_oil <dbl>,
## #   involvement_pesticides <dbl>, involvement_small_arms <dbl>,
## #   involvement_thermal_coal <dbl>, involvement_tobacco <dbl>, ...
```

```r
#Now setting up the dataset for splitting

set.seed(25)

#Split is for dividing the dataset into a training and test set.
# Strata makes the makes sure that the variables are evenally distributed
# across the splits - the split is 80% training and 20% test.

split <- initial_split(furtherReduced, prop = 0.8, strata = returns)

train_data2 <- training(split)
test_data2 <- testing(split)

#The random forests model
show_model_info("rand_forest")
```

```
## Information for `rand_forest`
##  modes: unknown, classification, regression, censored regression
```

```
##
##  engines:
##    classification: randomForest, ranger, spark
##    regression:     randomForest, ranger, spark
##
##  arguments:
##    ranger:
##       mtry  --> mtry
##       trees --> num.trees
##       min_n --> min.node.size
##    randomForest:
##       mtry  --> mtry
##       trees --> ntree
##       min_n --> nodesize
##    spark:
##       mtry  --> feature_subset_strategy
##       trees --> num_trees
##       min_n --> min_instances_per_node
##
##  fit modules:
##          engine           mode
##          ranger classification
##          ranger     regression
##    randomForest classification
##    randomForest     regression
##           spark classification
##           spark     regression
##
##  prediction modules:
##             mode       engine                    methods
##    classification randomForest           class, prob, raw
##    classification       ranger class, conf_int, prob, raw
##    classification        spark                class, prob
##        regression randomForest               numeric, raw
##        regression       ranger     conf_int, numeric, raw
##        regression        spark                     numeric
```

```
rf_model <-
  #The main parameter - mtry is the number of random selected
  # predictors to try at each branching
  rand_forest(trees = 200, min_n = 5) %>%
  #set for regression and not classification
  set_mode("regression") %>%
  #default engine
  set_engine("ranger")

#Again I prepare the recipe for pre-processing the data
prep_recipe2 <-
  #First you set up the model
  recipe(returns ~ ., data = train_data2) %>%
  #Next I remove the correlations greater than a threshold - here is 0.8
  step_corr(all_numeric(), threshold = 0.8) %>%
  #Next I normalize the predictor variables minus the variables that are categorical
  step_normalize(all_predictors(), -all_nominal())
```

```r
#Here I "juice" the recipe - essentially applying the steps above to the dataset
dia_juiced2 <- juice(prep(prep_recipe2))

#Now to fit the data
set.seed(1)
rf_reg_fit <- rf_model %>%
  fit(returns ~., data = dia_juiced2)

#The results of the RF fit
rf_reg_fit
```

```
## parsnip model object
##
## Ranger result
##
## Call:
##  ranger::ranger(x = maybe_data_frame(x), y = y, num.trees = ~200,      min.node.size = min_rows(~5, x
##
## Type:                             Regression
## Number of trees:                  200
## Sample size:                      20976
## Number of independent variables:  15
## Mtry:                             3
## Target node size:                 5
## Variable importance mode:         none
## Splitrule:                        variance
## OOB prediction error (MSE):       13.57942
## R squared (OOB):                  0.6850169
```

```r
#Now to prep the test_data2
prep_recipe2 <-
  #First you set up the model
  recipe(returns ~ ., data = test_data2) %>%
  #Next I remove the correlations greater than a threshold - here is 0.8
  step_corr(all_numeric(), threshold = 0.8) %>%
  #Next I normalize the predictor variables minus the variables that are categorical
  step_normalize(all_predictors(), -all_nominal())

test_juiced <- juice(prep(prep_recipe2))

PredRF_model <- predict(rf_reg_fit, test_juiced)
PredRF_model$returns <- test_data2$returns


metrics(PredRF_model, truth = returns, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       4.33
## 2 rsq     standard       0.600
## 3 mae     standard       2.70
```

```
#The results of the random forest are similar rmse values (4.33) to the LM
# but with a better .60 rsquared.
# Overall the model is still fairly poor at predicting but much more
# of the variation is accounted for.


################################################
################## DISCUSSION ##################
################################################

#Going back to the original questions:
# 1) Do ethical investments lead to better returns?
# 2) Can we actually predict returns based on these ratings?

#For the first question:
#We can see in the results is that ethical investments appear to lead
# to potentially better returns. As you increased along the environmental
# or governance scores you had a lower return. The opposite is true for social
# scores. This may reflect the historical view of the past 50 years that you
# need to appear to be a good steward of the environment or involved in good
# governance. The social score pattern however is more difficult to explain.
# Generally, for the involvements, the fewer involves the better the returns.
# This suggests that these controversial topics are potentially avoided
# by the funds. This however, is not true for military contracting or
# alcohol suggesting that the profit to be made by investing in these categories
# outwise the potential negative public view.

#For the second question:
# The predictive ability of these models appears to be relatively limited
# with a high RMSE value. However, the random forest model was able to explain
# over half the variance see in the returns suggesting that the variables
# do explain the returns to some degree. A boosted random forest model
# would be a potential next step to improve the models.

#Future Direction:
# Additional avenues to explore would be to include the risk_ratings and
# performance_ratings. Are the scores and involvements able to predict
# the performance ratings? Another question to explore the degree of relatedness
# between the ESG scores and the involvements?
```