Dawei He, Mingkang Yuan
Dr. Weyne Lee
STAT 5206
April 16, 2021

Exploration of Mutual Funds and ETFs

Abstract

ETFs represent a cheap alternative to Mutual Funds and they are growing fast in the last decade. ETF, as a rising investment method in the past decade, has been a hot topic in the investment circle and highly sought after by the market. Here comes to the question: In this paper, we will take the detailed data of ETFs and MF(Two sources selected from Kaggle) in the past ten years as the source, use Data Wrangling, data manipulation and data visualization to focus on learning the difference in quantization between ETFs and MF, and make models for analysis and optimization to recommend which products investors should choose to invest in.

Presentation link:

https://drive.google.com/file/d/1dsZXegEKEUflAqja2j3oWfqMt9C4HT_0/view?usp=sharing

1. Introduction

Because the tuition at Columbia University is tremendously expensive, we want to study how to choose funds that have high annual returns which can alleviate our financial burden. To put it simply, a mutual fund is a fund-raising investment vehicle that collects a lot of small money and turns it into a big one, and then gives it to someone or a professional organization to operate and manage it for profit. An exchange-traded fund (ETF) was first introduced in 1993 as a branch of a mutual fund, and it is a basket of securities that trade on an exchange, just like a stock. So its price changes throughout the day and it can't be traded after the market is closed. The main difference is that Mutual funds are generally bought directly from investment companies instead of from other Investors on an exchange. Unlike ETFs, they don't have trading commissions, But they do carry an expense ratio and other sales fees.

Here comes to the question: One is the traditional investment method, the other is the investment method which has emerged in the last decade. What are their advantages and disadvantages? How should we choose? We will use two datasets that are both scrapped from *Yahoo Finance* and *Morningstar.com* by Stefano Leone from Kaggle. These two data sets are from Yahoo Finance and Morningstar.com two different sources so that our analysis is more comprehensive and convincing. Our objective is to visualize and model both datasets separately and intravenously to analyze the difference of these two investment methods, and from the perspective of investors to make and compare the model and the parameters so as to give the majority of investors advice on investment.

2. Data preprocessing

All the original data scrapped from *Yahoo Finance* and *Morningstar.com* were processed by Stefano Leone for the first time, so most of the column names are corresponding to each other. However, the original data sets are really large. The mutual_funds_data has dimension of 24821* 173 and has 173 columns, and the etfs_data has dimension of 1680*118 and has 118 columns. Since we aim to compare the difference between Mutual funds and ETFs, we select the intersection columns of two data sets and drop the columns that are not useful for our project. Add a column called fund_type which is used for joining the two dataset afterwards. So finally we create three new datasets. The data wrangling processes are manipulated by using base R function and the tidyverse package. All three new datasets namely mutual_funds, etfs and both_funds, end up with columns of 111. The code of this process are shown in Figure 1

```r
mutual_funds = mutual_funds_data %>%
  select(intersect(names(mutual_funds_data), names(etfs_data))) %>%
  select(-fund_symbol, -fund_extended_name, -fund_family, -inception_date,
         -category, -investment_strategy, -currency, -top10_holdings) %>%
  mutate(fund_type = "mutual funds")

etfs = etfs_data %>%
  select(intersect(names(mutual_funds_data), names(etfs_data)))%>%
  select(-fund_symbol, -fund_extended_name, -fund_family, -inception_date,
         -category, -investment_strategy, -currency, -top10_holdings) %>%
  mutate(fund_type = "etfs")

## Rbind datasets
both_funds = rbind(mutual_funds, etfs)
```

Figure 1: Data Wrangling Process

In order to explore the advantages and disadvantages of each fund, let's visualize the differences of some key variables first. When we are doing unique(mutual_funds$size_type) which gives us the unique value of the column with name(size_type) in the mutual_funds data set, the results are Large, Medium and Small. So we make a bar plot of the number of each investment type with different size types of Mutual Fund and ETFs. Figure 2 and 3 show the barplot for each fund.

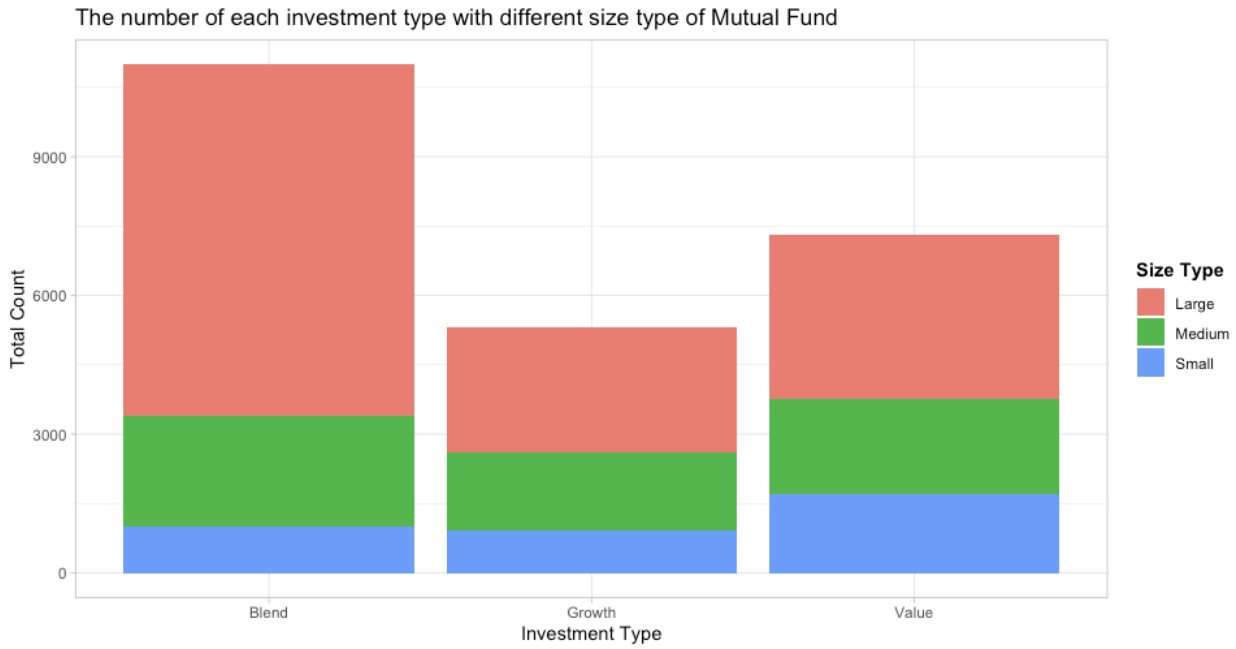


Figure 2: Barplot of the number of each investment type with different size type of Mutual Fund

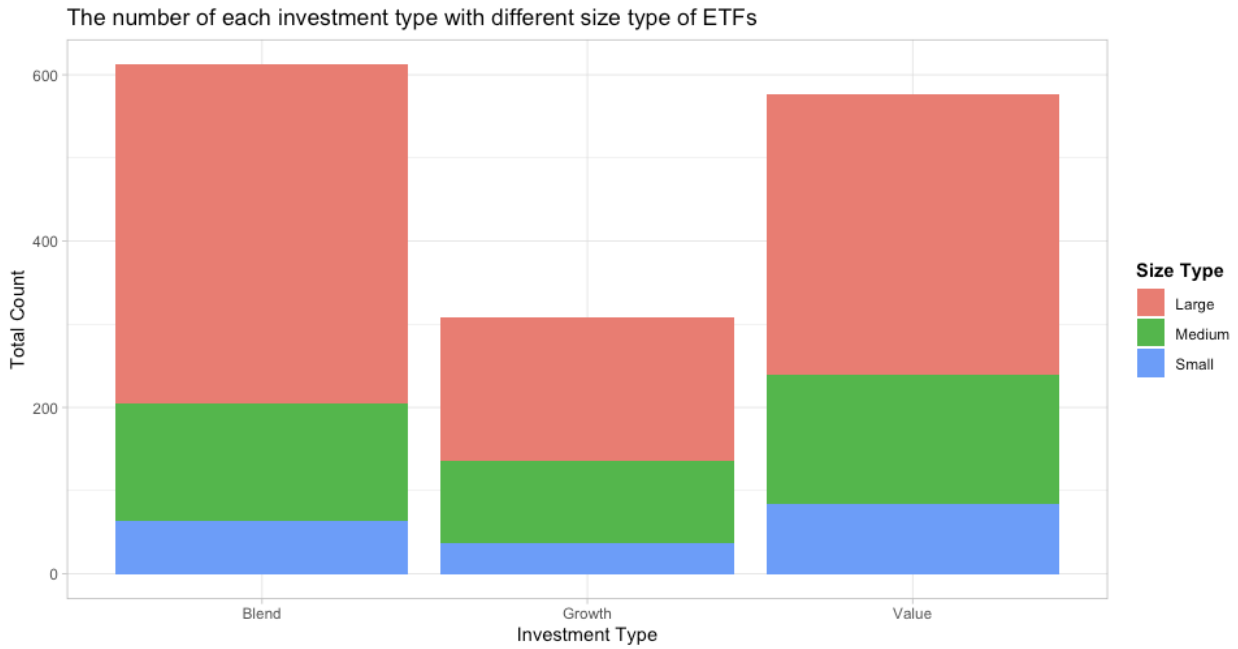


Figure 3: Barplot of the number of each investment type with different size type of ETFs

As we can see, in blend investment type, large size types account for a large proportion in both mutual funds and ETFs. And the large size type always has a large proportion in every investment type. To visually see these two funds separately, we found that the proportion of the size type and the investment type of each fund are almost the same. Next, to explore the relation of investment type with each variable, we plot the density plot of each variable with corresponding investment type of Mutual Funds and ETFs. We found that the blend investment type accounted for a high proportion in most variables. We will use these density plots for analysing the models and make recommendations later. Figure 4 and 5 show the density plot for each fund.

Figure 4: Density plot of each variables with corresponding investment type of Mutual Funds

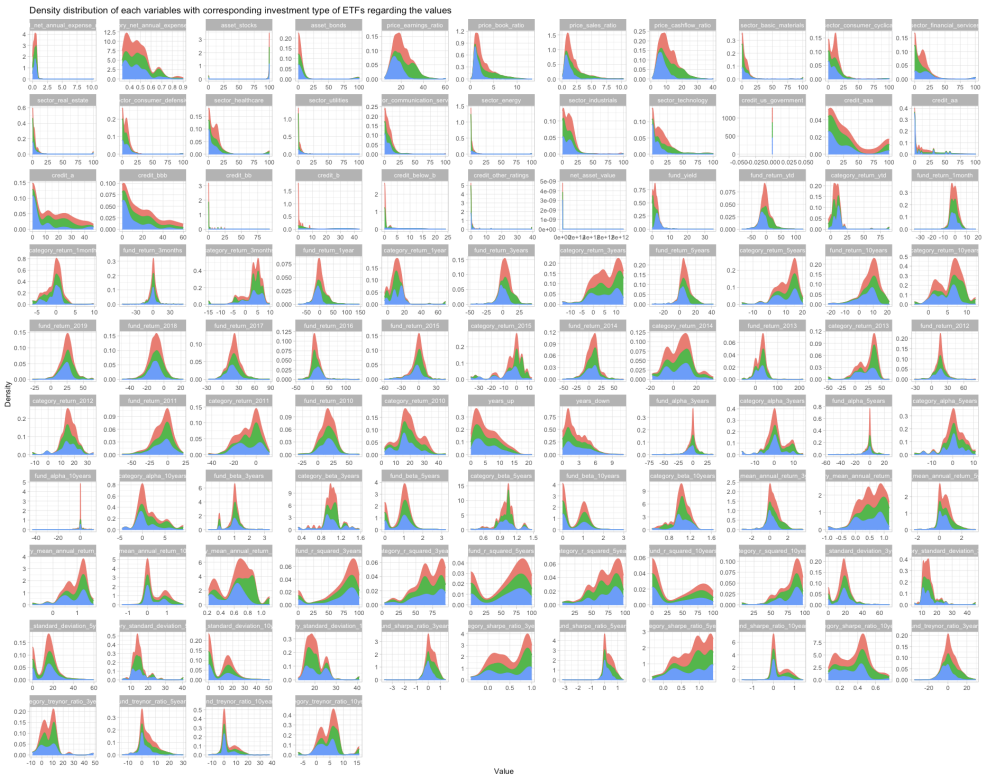


Figure 5: Density plot of each variables with corresponding investment type of ETFs

Sector shows how much funds invest in specific areas. Our next step is to visualize the distribution of each sector between mutual funds and ETFs. So we select every column that starts with the string “sector” from mutual_funds and etfs, use the function rbind() to row combine them together, and add another column called type which indicates the type of investment using mutate() function. Then we plot the relationship of each sector and each fund. The version I (figure 6) shows the comparison of distribution between ETFs and Mutual funds of each sector, version II (figure 7) provides a clearer comparison of the distribution of each sector and the version III (figure 8) provides a comparison of overall distribution of each sector. (Be aware of the differences between these three versions). We observe that ETFs are mostly used for financial services, consumer cyclical, technology, industrial and health care. And mutual funds are mostly used for technology, financial services, health care, and consumer cyclical.. And the overall funds are mostly used for technology, financial services, health care, consumer cyclical and industrial. (The order goes from the largest to the smallest).

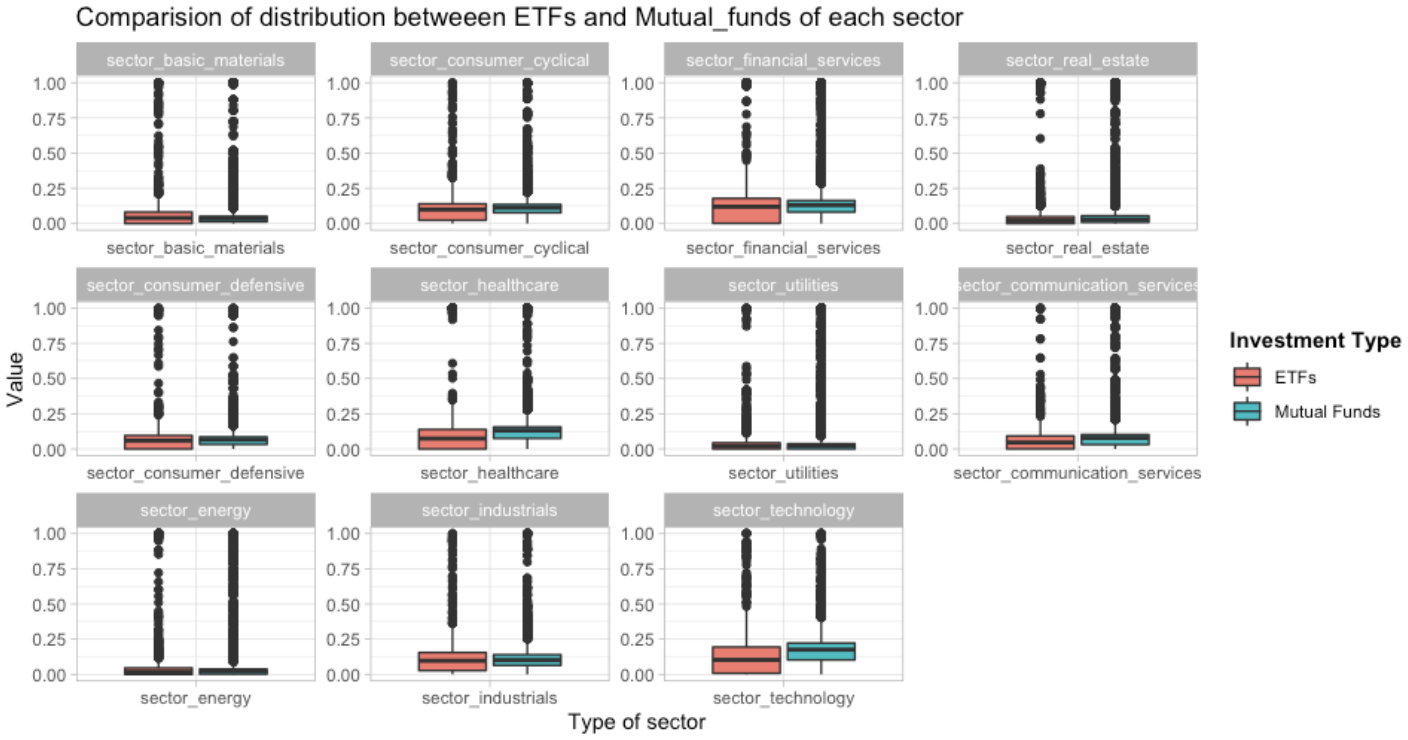


Figure 6: Comparison of distribution between ETFs and Mutual_funds of each sector

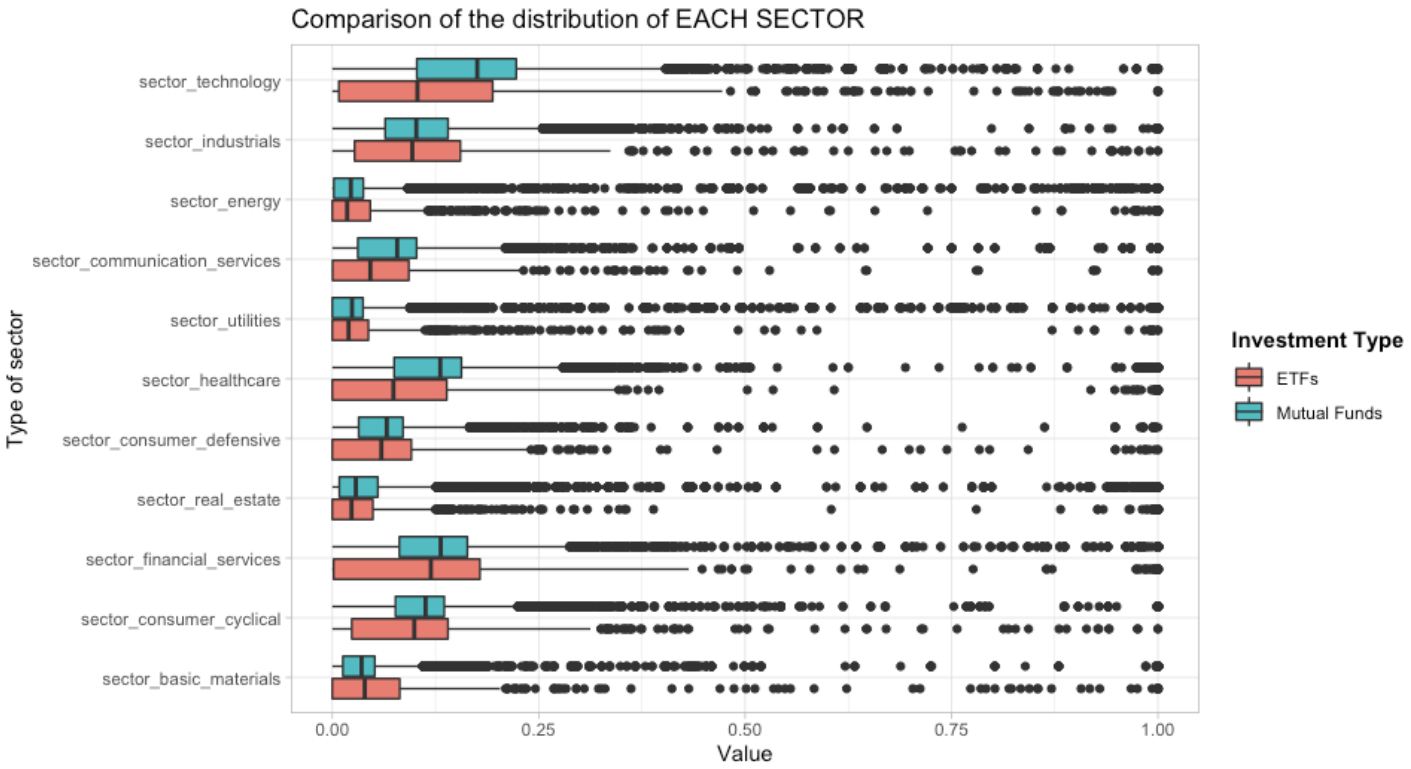


Figure 7: Comparison of the distribution of each sector

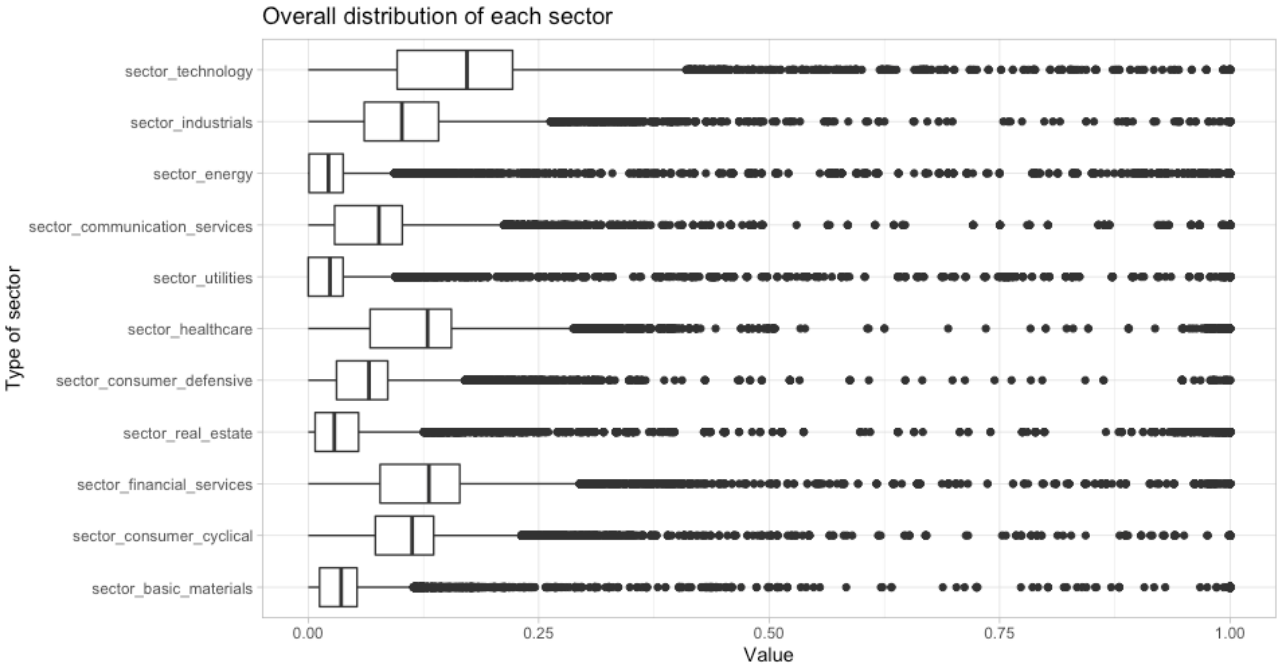


Figure 8: Comparison of overall distribution of each sector

We will combine the data visualization results with the parameters of our models below to make recommendations later.

3. Modeling

3.1 Stepwise AIC method

This part we will use stepwise AIC method to modeling the Mutual Funds data. Actually the stepwise AIC regression model is not good and not bad for mutual funds. But when we continue to use the same method to make the stepwise AIC regression model for ETFs, the R-squared is 0.9111 and adjusted R-squared is 0.9043 which is a little bit low. Also we found the performance of the correlation of stepwise AIC regression models of ETFs is horrible. So we tried a new method called stepwise VIF selection which we will cover in section 3.2. In section 3.2, we got much better results than the model using the AIC method of each fund. Thus we will mainly based on the stepwise VIF selection models of each fund to make recommendations, but we still decide to show the process of modeling using the Stepwise AIC method.

3.1.1 AIC Model for Mutual Funds

In order to create a dataset used for creating the model, we only select the numerical columns in mutual_funds_data and add the investment_type and size_type columns. Then we list the number of NAs of each column using for-loop. Figure 9 shows the results of top 50 columns that contain most NAs.

```
> head(sort(y, decreasing = TRUE, 20), 50)
          credit_us_government                      credit_aaa                       credit_aa
                         14789                           14789                           14789
                      credit_a                      credit_bbb                       credit_bb
                         14789                           14789                           14789
                      credit_b                  credit_below_b             credit_other_ratings
                         14789                           14789                           14789
             fund_return_2010              fund_return_2010_q4              fund_return_2010_q3
                         12397                           12395                           12183
           fund_return_2010_q2             category_return_2010              fund_return_2010_q1
                         11886                           11686                           11558
             fund_return_2011              fund_return_2011_q4              fund_return_2011_q3
                         11327                           11317                           11100
           fund_return_2011_q2             category_return_2011              fund_return_2011_q1
                         10834                           10736                           10596
             fund_return_2012              fund_return_2012_q4              fund_return_2012_q3
                         10216                           10214                            9991
                bond_maturity              fund_return_2012_q2             category_return_2012
                          9768                            9612                            9366
           fund_return_2012_q1                 fund_return_2013              fund_return_2013_q4
                          9225                            8778                            8772
                bond_duration              fund_return_2013_q3              fund_return_2013_q2
                          8631                            8545                            8230
          category_return_2013              fund_return_2013_q1              fund_return_10years
                          7900                            7897                            7551
           fund_alpha_10years                fund_beta_10years fund_mean_annual_return_10years
                          7527                            7527                            7527
      fund_r_squared_10years fund_standard_deviation_10years       fund_sharpe_ratio_10years
                          7527                            7527                            7527
             fund_return_2014              fund_return_2014_q4              fund_return_2014_q3
                          7498                            7494                            7191
          price_earnings_ratio              fund_return_2014_q2             category_return_2014
                          6965                            6852                            6509
          price_cashflow_ratio              fund_return_2014_q1
                          6503                            6503
```

Figure 9: Top 50 columns that contain most NAs

Then we drop some useless columns that contain most NAs, and create a new data set called mutual_reg_data_new for creating the model. Figure 10 shows the piece code of how we select and drop meaningless columns.

```
154  ## Drop the columns with more NAs and create a new mutual_reg_data_new.
155  mutual_reg_data_new = mutual_funds_data %>%
156    select(where(is.numeric)) %>%
157    mutate(investment_type = mutual_funds_data$investment_type,
158           size_type = mutual_funds_data$size_type) %>%
159    select(-starts_with("credit"),-starts_with("category"), -ends_with("q1"), -ends_with("q2"),
160           -ends_with("q3"), -ends_with("q4"), -ends_with("ytd")) %>%
161    na.omit()
```

Figure 10: Piece code for select and drop useless columns

We install the library called Olsrr and use lm() function to fit a linear model called full_model for our data set. Then use ols_step_both_aic() function to build a regression model from the candidate predictor variables of full_model by entering and removing predictors based on akaike information criteria, in a stepwise manner until there is no variable left to enter or remove any more. So in this process, the ols_step_both_aic() function automatically optimizes the fitted parameters to the model. Figure 11 shows the predictors that ols_step_both_aic() function selected for us. So we will use these predictors to create a new model called mutual_model. Since we are going to compare the coefficients of each predictors, we use lm.beta() function from the package(lm.beta) to standardize the regression model and get the standardized coefficients from it. Figure 12 shows the process of creating and normalizing the model.

```
> step_aic_both$predictors
 [1] "fund_mean_annual_return_10years" "asset_stocks"                    "fund_mean_annual_return_3years"
 [4] "fund_return_2018"                "fund_return_1year"               "fund_return_1month"
 [7] "fund_return_5years"              "fund_return_3years"              "fund_standard_deviation_3years"
[10] "asset_stocks"                    "fund_return_2017"                "fund_return_2016"
[13] "fund_return_2014"                "investment_type"                 "fund_return_3months"
[16] "fund_return_2015"                "fund_return_2011"                "fund_return_2013"
[19] "fund_return_2012"                "fund_return_2010"                "fund_return_1month"
[22] "fund_mean_annual_return_5years"  "fund_alpha_10years"              "fund_standard_deviation_10years"
[25] "size_type"                       "fund_beta_10years"               "fund_sharpe_ratio_3years"
[28] "fund_sharpe_ratio_5years"        "fund_treynor_ratio_5years"       "sector_financial_services"
[31] "sector_utilities"                "fund_r_squared_10years"          "fund_r_squared_5years"
[34] "sector_consumer_cyclical"        "fund_standard_deviation_5years"  "fund_return_5years"
[37] "sector_real_estate"              "sector_basic_materials"          "sector_technology"
[40] "median_market_cap"               "fund_net_annual_expense_ratio"   "price_cashflow_ratio"
[43] "price_book_ratio"                "quarters_down"                   "sector_industrials"
[46] "net_asset_value"                 "fund_sharpe_ratio_10years"       "fund_yield"
[49] "asset_bonds"                     "bond_duration"                   "bond_maturity"
[52] "asset_others"                    "asset_convertable"               "price_earnings_ratio"
[55] "quarters_up"                     "sector_consumer_defensive"       "price_sales_ratio"
[58] "asset_preferred"                 "sector_communication_services"
```

Figure 11: Selected predictors by ols_step_both_aic() function

```
mutual_model = lm(fund_return_2019 ~ fund_mean_annual_return_10years + fund_mean_annual_return_3years +
                    fund_return_2018 + fund_return_1year + fund_return_3years + fund_standard_deviation_3years +
                    fund_return_2017 + fund_return_2016 + fund_return_2014 + investment_type +
                    fund_return_3months + fund_return_2015 + fund_return_2011 + fund_return_2013 +
                    fund_return_2012 + fund_return_2010 + fund_mean_annual_return_5years +
                    fund_alpha_10years + fund_standard_deviation_10years + size_type +
                    fund_beta_10years + fund_sharpe_ratio_3years + fund_sharpe_ratio_5years +
                    fund_treynor_ratio_5years + sector_financial_services + sector_utilities +
                    fund_r_squared_10years + fund_r_squared_5years + sector_consumer_cyclical +
                    fund_standard_deviation_5years + sector_real_estate + sector_basic_materials +
                    sector_technology + median_market_cap + fund_net_annual_expense_ratio +
                    price_cashflow_ratio + price_book_ratio + quarters_down + sector_industrials +
                    net_asset_value + fund_sharpe_ratio_10years + fund_yield + asset_bonds +
                    bond_duration + bond_maturity + asset_others + asset_convertable +
                    price_earnings_ratio + quarters_up + sector_consumer_defensive +
                    price_sales_ratio + asset_preferred + sector_communication_services,
                    data = mutual_reg_data_new)

library(lm.beta)
lm.beta(mutual_model)
standardized_mutual_beta = lm.beta(mutual_model)
```

Figure 12: Piecewise code for creating and normalizing the AIC model for Mutual Funds

Then we list the top 20 highest parameter coefficients and top 20 lowestest parameter coefficients.

```
> ## Top 30 highest parameter coefficients
> standardized_mutual_beta$standardized.coefficients %>%
+   unlist() %>% sort(decreasing = TRUE) %>% head(20)
             fund_return_3years fund_mean_annual_return_10years  fund_mean_annual_return_5years
                    2.515086074                     1.463094939                     1.125843615
 fund_standard_deviation_5years        fund_sharpe_ratio_3years  fund_standard_deviation_3years
                    0.319090052                     0.187512987                     0.173386432
          fund_r_squared_5years           price_cashflow_ratio           investment_typeValue
                    0.072988292                     0.052533798                     0.041545533
             median_market_cap                 bond_maturity       fund_treynor_ratio_5years
                    0.028458104                     0.021244661                     0.019990263
            sector_industrials               price_sales_ratio                      asset_bonds
                    0.017929206                     0.014524679                     0.014521376
      sector_financial_services               sector_utilities             fund_alpha_10years
                    0.013574467                     0.012913790                     0.010866341
                     fund_yield                net_asset_value
                    0.009749498                     0.009296426
> ## Top 30 lowest parameter coefficients
> standardized_mutual_beta$standardized.coefficients %>%
+   unlist() %>% sort(decreasing = FALSE) %>% head(20)
 fund_mean_annual_return_3years               fund_return_1year                    fund_return_2018
                    -1.66982295                     -1.52069241                     -0.90188120
               fund_return_2017                fund_return_2016                    fund_return_2013
                    -0.65636722                     -0.54223975                     -0.46324021
               fund_return_2014                fund_return_2011 fund_standard_deviation_10years
                    -0.33308233                     -0.31128964                     -0.27091903
               fund_return_2015                fund_return_2012             fund_return_3months
                    -0.23209782                     -0.14235632                     -0.12539338
         fund_r_squared_10years         fund_sharpe_ratio_5years                price_book_ratio
                    -0.09748720                     -0.09133016                     -0.05557756
               fund_return_2010        fund_sharpe_ratio_10years                   quarters_down
                    -0.05372746                     -0.04689449                     -0.04383727
             sector_real_estate                   bond_duration
                    -0.04063133                     -0.03204539
```

Figure 13: Top 20 highest and lowest parameter coefficients

Using summary() function, we find that the R-squared is 0.9568 and adjusted R-squared is 0.9564, which is ok. Next, we want to check the correlations of each predictors. The correlation plot of our stepwise AIC regression model is shown in Figure 14. The redder or bluer each square is, the more correlated the two corresponding predictors are. So it can be concluded that there are still many predictors correlated with each other, especially positively correlated, even after using ols_step_both_aic() function to exclude some of predictors from all of them.

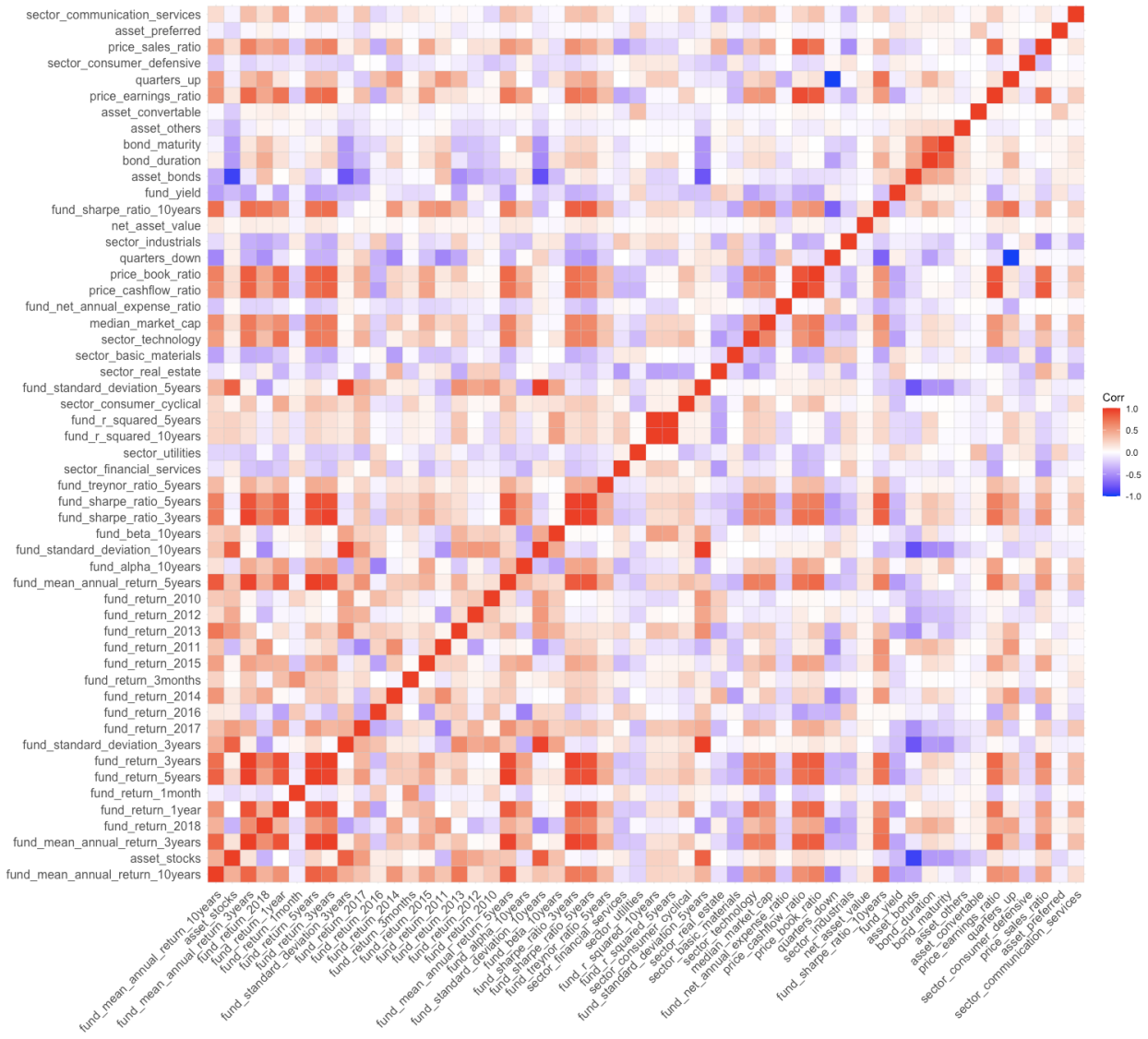


Figure 14: The correlation of stepwise AIC regression model of Mutual Funds

3.1.2 AIC Model for ETFs

Basically we can do the same process as section 3.11. We still need to select the numerical columns in ETFs and add the investment_type and size_type columns. Then list the number of NAs of each column and drop the columns with most NAs to get a new data frame named etfs_reg_data_new. We can use the stepwise AIC ols_step_both_aic() function again to select the

most valuable predictors. Finally we create the AIC model for ETFs by using the lm() function based on the selected predictors. Figure 15 shows the codes for creating the model named etfs_model. And the top 20 highest and lowest parameter coefficients using stepwise AIC method and the correlation plot are shown in Figure 16 and Figure 17. Though summary() function gives us the multiple R-squared 0.9111and adjusted R-squared 0.904, the performance of correlation plot actually is horrible.

```
## Pick the parameter from the results and create the model
etfs_model = lm(fund_return_2019 ~ fund_return_1year + fund_return_2018 + fund_return_5years +
                  fund_treynor_ratio_3years + fund_return_2017 + fund_return_2016 +
                  fund_return_2015 + fund_yield + fund_sharpe_ratio_5years + fund_return_2013 +
                  fund_alpha_5years + fund_alpha_10years + fund_sharpe_ratio_10years +
                  sector_energy + sector_technology + fund_return_2011 + sector_consumer_defensive +
                  fund_treynor_ratio_5years + fund_treynor_ratio_10years + sector_utilities +
                  fund_standard_deviation_3years + fund_return_10years + price_earnings_ratio +
                  fund_beta_3years + fund_mean_annual_return_5years + fund_return_1month +
                  fund_beta_5years + fund_return_3years + fund_standard_deviation_5years +
                  fund_alpha_3years, data = etfs_reg_data_new)
```

Figure 15: Piecewise code for AIC model of ETFs

```
> standardized_etfs_beta$standardized.coefficients %>% sort(decreasing = TRUE) %>% head(20)
             fund_alpha_5years              fund_return_5years      fund_treynor_ratio_3years
                    2.33801513                      2.12429371                     0.77905696
fund_standard_deviation_3years               fund_beta_5years             fund_return_3years
                    0.68578368                      0.66367832                     0.59880427
     fund_sharpe_ratio_10years     fund_treynor_ratio_10years            fund_return_10years
                    0.37803564                      0.36147917                     0.30497846
             fund_return_2013               fund_return_2011                   sector_energy
                    0.25503648                      0.11785204                     0.09936263
           sector_technology               sector_utilities                      fund_yield
                    0.08011199                      0.06237506                     0.06057347
                  (Intercept)            fund_return_1month          price_earnings_ratio
                    0.00000000                     -0.05086079                    -0.05770843
     sector_consumer_defensive               fund_return_2015
                   -0.06201631                     -0.16784535
> standardized_etfs_beta$standardized.coefficients %>% sort(decreasing = FALSE) %>% head(20)
             fund_return_1year             fund_alpha_10years fund_mean_annual_return_5years
                   -1.62830331                     -1.09875533                    -1.01012688
             fund_return_2018      fund_treynor_ratio_5years               fund_return_2016
                   -0.94371689                     -0.88818556                    -0.78509231
             fund_return_2017               fund_beta_3years       fund_sharpe_ratio_5years
                   -0.66051295                     -0.60018735                    -0.57573100
fund_standard_deviation_5years              fund_alpha_3years               fund_return_2015
                   -0.35315977                     -0.31147247                    -0.16784535
     sector_consumer_defensive          price_earnings_ratio             fund_return_1month
                   -0.06201631                     -0.05770843                    -0.05086079
                  (Intercept)                      fund_yield               sector_utilities
                    0.00000000                      0.06057347                     0.06237506
           sector_technology                    sector_energy
                    0.08011199                      0.09936263
```

Figure 16: Top 20 highest and lowest parameter coefficients using stepwise AIC method

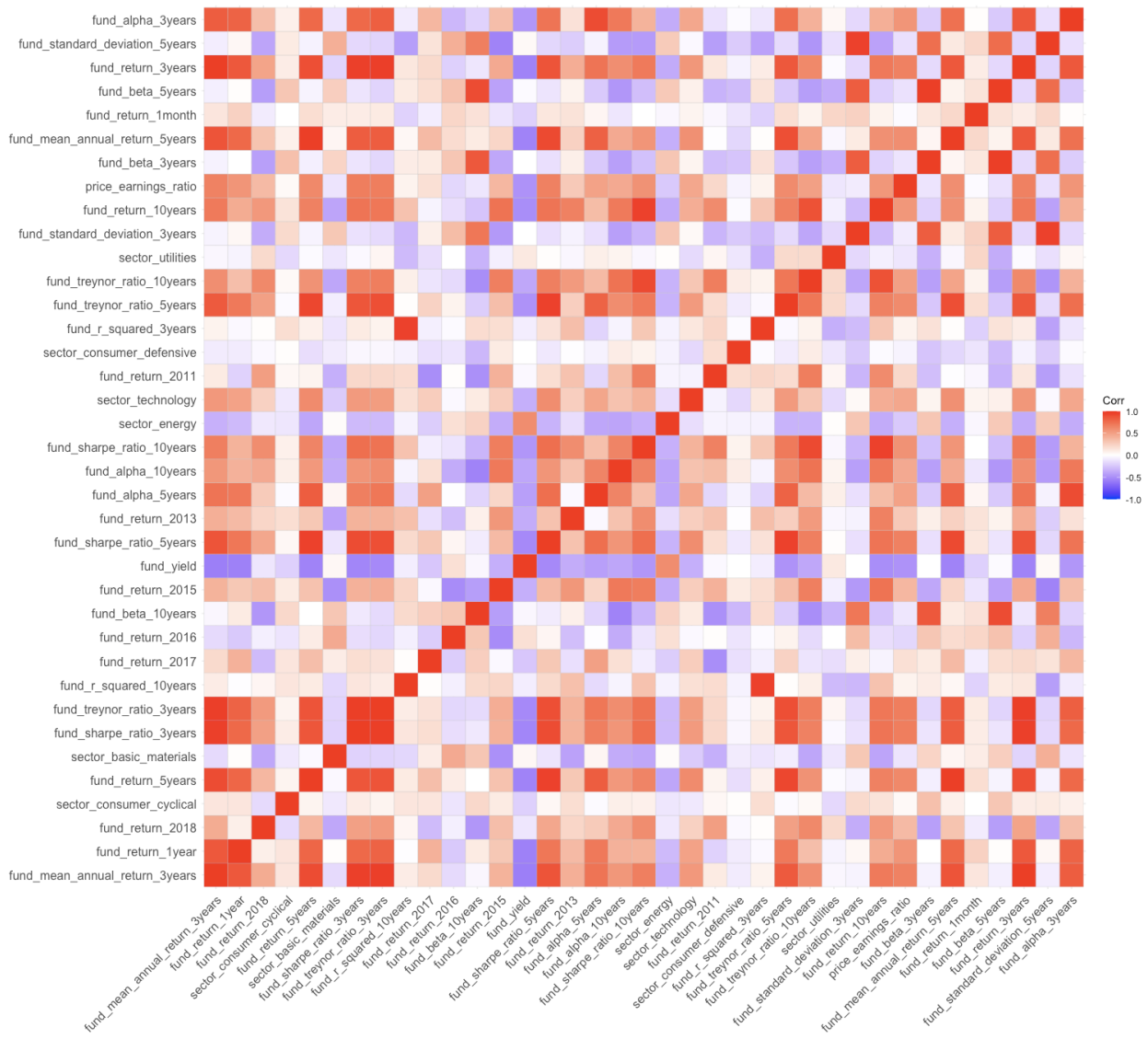


Figure 17: The correlation of stepwise AIC regression model of ETFs

3.2 Stepwise VIF selection

3.2.1 Linear Model for ETFs

Since we want to use beta values to estimate each explanatory variables' impact on the main response and find out those who have the most significant influence, multicollinearity must be avoided as much as possible. Otherwise, the beta values of those correlated variables will tend to be much smaller and can not truly represent their corresponding explanatory variables' impact to the main response. So another method must be deployed to further exclude some predictors so as to control the multicollinearity at an acceptable level. A simple approach to identify collinearity among explanatory variables is the use of variance inflation factors (VIF). VIF calculations are straightforward and easily comprehensible; the higher the value, the higher the collinearity. A VIF for a single explanatory variable is obtained using the r-squared value of the regression of that variable against all other explanatory variables. So instinctively, we can calculate the VIF of each explanatory variable and remove those who have high VIF values. But removing individual variables with high VIF values is insufficient in the initial comparison using the full set of explanatory variables. The VIF will change accordingly after each variable is removed. Referring to the forward and backward AIC method, we can also use a stepwise approach to dynamically calculate VIF values and remove the variable with the highest value each time until all VIF values are below a certain value. So here we will use a custom function to select variables by stepwise approach based on VIF values. The custom function named vif_func is defined as in Figure 18.

```r
#VIF
vif_func<-function(in_frame,thresh=10,trace=T,...){
  library(fmsb)
  if(any(!'data.frame' %in% class(in_frame))) in_frame<-data.frame(in_frame)
  #get initial vif value for all comparisons of variables
  vif_init<-NULL
  var_names <- names(in_frame)
  for(val in var_names){
    regressors <- var_names[-which(var_names == val)]
    form <- paste(regressors, collapse = '+')
    form_in <- formula(paste(val, '~', form))
    vif_init<-rbind(vif_init, c(val, VIF(lm(form_in, data = in_frame, ...))))
  }
  vif_max<-max(as.numeric(vif_init[,2]), na.rm = TRUE)
  if(vif_max < thresh){
    if(trace==T){ #print output of each iteration
      prmatrix(vif_init,collab=c('var','vif'),rowlab=rep('',nrow(vif_init)),quote=F)
      cat('\n')
      cat(paste('All variables have VIF < ', thresh,', max VIF ',round(vif_max,2), sep=''),'\n\n')
    }
    return(var_names)
  }
  else{
    in_dat<-in_frame
    #backwards selection of explanatory variables, stops when all VIF values are below 'thresh'
    while(vif_max >= thresh){
      vif_vals<-NULL
      var_names <- names(in_dat)
      for(val in var_names){
        regressors <- var_names[-which(var_names == val)]
        form <- paste(regressors, collapse = '+')
        form_in <- formula(paste(val, '~', form))
        vif_add<-VIF(lm(form_in, data = in_dat, ...))
        vif_vals<-rbind(vif_vals,c(val,vif_add))
      }
      max_row<-which(vif_vals[,2] == max(as.numeric(vif_vals[,2]), na.rm = TRUE))[1]
      vif_max<-as.numeric(vif_vals[max_row,2])
      if(vif_max<thresh) break
      if(trace==T){ #print output of each iteration
        prmatrix(vif_vals,collab=c('var','vif'),rowlab=rep('',nrow(vif_vals)),quote=F)
        cat('\n')
        cat('removed: ',vif_vals[max_row,1],vif_max,'\n\n')
        flush.console()
      }
      in_dat<-in_dat[,!names(in_dat) %in% vif_vals[max_row,1]]
    }
    return(names(in_dat))
  }
}
```

Figure 18. Custom function vif_func

Firstly, we create a tibble called etfs_reg_data_new_vif_without_y that drops some useless columns and main response which is fund_return_2019 here. Also we create a similar tibble but

keep that fund_return_2019 column and name this tibble as etfs_reg_data_new_vif_with_y. Then we use tibble etfs_reg_data_new_vif_without_y as input of vif_func and set VIF threshold value as 10. The vif_func will automatically drop variables until all VIF values are below 10. Now we get the new selected variables that are less correlated with each other and use them to regress the new model of fund_return_2019 using tibble etfs_reg_data_new_vif_with_y, and name it etfs_vif_model. We can obtain the adj Rsq value which is 0.6946 by calling summary(etfs_vif_model) function. Though its adj Rsq is much lower than the one by using the AIC method, the multicollinearity problem has been alleviated to a great extent according to the correlation plot in Figure 19.

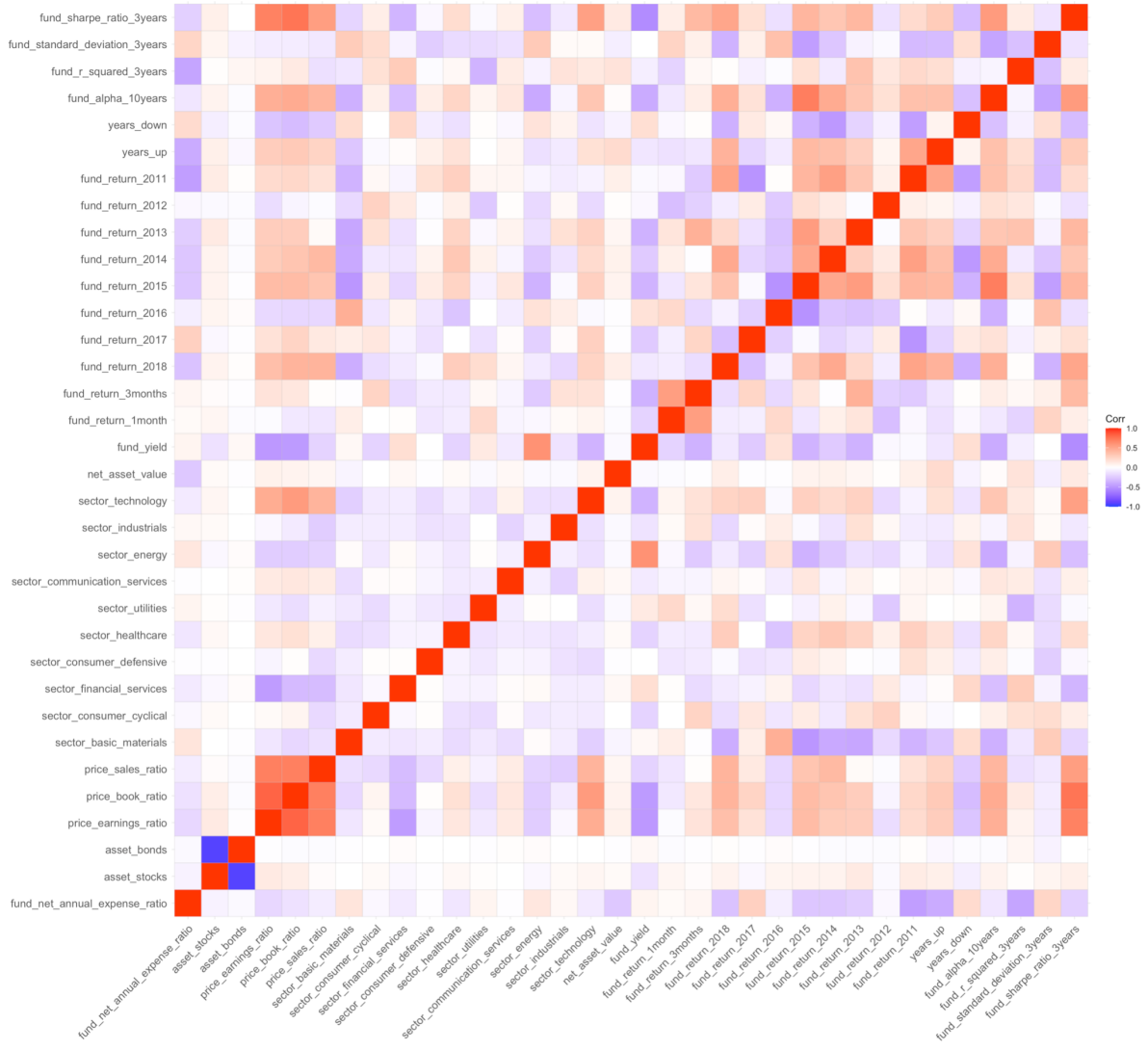


Figure 19: The correlation of stepwise VIF regression model of ETFs

By calling anova function, significant levels of each explanatory variable can be intuitively seen, as the result shown in Figure 20. The stars and dots sign represent significant values for each variable, therefore it can be concluded that most of the variables obtained by stepwise VIF method are of high significance, which is a desired outcome of selecting explanatory variables. So the stepwise VIF method is a much better method in terms of reducing multicollinearity.

```
> anova(etfs_vif_model)
Analysis of Variance Table

Response: fund_return_2019
                               Df  Sum Sq Mean Sq  F value    Pr(>F)
fund_net_annual_expense_ratio   1  2315.7  2315.7  71.8360 4.931e-16 ***
asset_stocks                    1   285.5   285.5   8.8561 0.0031041 **
asset_bonds                     1   318.3   318.3   9.8730 0.0018060 **
price_earnings_ratio            1  7706.3  7706.3 239.0644 < 2.2e-16 ***
price_book_ratio                1  1126.8  1126.8  34.9551 7.401e-09 ***
price_sales_ratio               1   324.3   324.3  10.0613 0.0016347 **
sector_basic_materials          1   213.1   213.1   6.6113 0.0105052 *
sector_consumer_cyclical        1   111.9   111.9   3.4716 0.0631874 .
sector_financial_services       1    77.9    77.9   2.4166 0.1208699
sector_consumer_defensive       1   217.8   217.8   6.7579 0.0096898 **
sector_healthcare               1   566.7   566.7  17.5798 3.416e-05 ***
sector_utilities                1     5.2     5.2   0.1615 0.6880002
sector_communication_services   1  1072.6  1072.6  33.2731 1.638e-08 ***
sector_energy                   1  1679.8  1679.8  52.1090 2.788e-12 ***
sector_industrials              1    12.8    12.8   0.3986 0.5282042
sector_technology               1  3216.5  3216.5  99.7804 < 2.2e-16 ***
net_asset_value                 1     5.3     5.3   0.1632 0.6864393
fund_yield                      1   111.9   111.9   3.4715 0.0631908 .
fund_return_1month              1    22.7    22.7   0.7042 0.4019074
fund_return_3months             1   971.6   971.6  30.1395 7.279e-08 ***
fund_return_2018                1    41.3    41.3   1.2799 0.2586202
fund_return_2017                1   493.3   493.3  15.3022 0.0001081 ***
fund_return_2016                1  1089.7  1089.7  33.8048 1.274e-08 ***
fund_return_2015                1   150.9   150.9   4.6803 0.0311201 *
fund_return_2014                1   205.9   205.9   6.3868 0.0118933 *
fund_return_2013                1   643.3   643.3  19.9574 1.040e-05 ***
fund_return_2012                1    99.7    99.7   3.0940 0.0793700 .
fund_return_2011                1   313.3   313.3   9.7194 0.0019593 **
years_up                        1    29.3    29.3   0.9100 0.3407026
years_down                      1    92.5    92.5   2.8701 0.0910441 .
fund_alpha_10years              1  1696.5  1696.5  52.6278 2.209e-12 ***
fund_r_squared_3years           1   389.7   389.7  12.0888 0.0005647 ***
fund_standard_deviation_3years  1  1175.4  1175.4  36.4634 3.643e-09 ***
fund_sharpe_ratio_3years        1  5255.5  5255.5 163.0365 < 2.2e-16 ***
Residuals                     388 12507.3    32.2
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 20. Anova table of etfs_vif_model

Therefore we can analyze each explanatory variables' impact on fund_return_2019 by comparing corresponding beta values. Figure 21 shows explanatory variables with top 20 positive beta values and explanatory variables with top 20 negative beta variables. We can analyze and summarize some valuable information based on Figure 21. Eventually, some practical recommendations can be proposed based on the information.

```
> standardized_etfs_vif$standardized.coefficients%>% sort(decreasing = TRUE) %>% head(20)
      fund_sharpe_ratio_3years             fund_alpha_10years fund_standard_deviation_3years
                   0.808031807                    0.316624373                    0.243463422
         fund_r_squared_3years              price_sales_ratio                fund_return_2013
                   0.232377340                    0.217590396                    0.135069420
              fund_return_2016              fund_return_2011                     years_down
                   0.096614779                    0.094828117                    0.072792671
            sector_technology              sector_utilities                net_asset_value
                   0.054940337                    0.052520175                    0.001639946
                   (Intercept)                     fund_yield      sector_financial_services
                   0.000000000                   -0.010600938                   -0.011131181
                  asset_stocks  fund_net_annual_expense_ratio                     asset_bonds
                  -0.016887071                   -0.029356205                   -0.038600714
           sector_industrials      sector_consumer_defensive
                  -0.045619694                   -0.055995378
> standardized_etfs_vif$standardized.coefficients%>% sort(decreasing = FALSE) %>% head(20)
              fund_return_2018              price_book_ratio sector_communication_services
                   -0.33233909                   -0.21195246                   -0.20227748
                 sector_energy        sector_basic_materials      sector_consumer_cyclical
                   -0.19261176                   -0.18092929                   -0.16166385
            sector_healthcare            fund_return_1month              fund_return_2017
                   -0.16118819                   -0.13197999                   -0.12778384
              fund_return_2014           fund_return_3months              fund_return_2015
                   -0.08981207                   -0.08555126                   -0.07577106
          price_earnings_ratio                      years_up              fund_return_2012
                   -0.07265842                   -0.06690769                   -0.05760000
     sector_consumer_defensive            sector_industrials                     asset_bonds
                   -0.05599538                   -0.04561969                   -0.03860071
 fund_net_annual_expense_ratio                   asset_stocks
                   -0.02935621                   -0.01688707
```

Figure 21. Top 20 highest and lowest beta values of ETFs using stepwise VIF method

3.2.2 Liner Model for Mutual Funds

Since the stepwise VIF method can significantly control the multicollinearity of selected explanatory variables in a desirable level, we decide to apply this method to Mutual Funds data. Using custom function vif_func with VIF threshold 10 to select explanatory variables out of Mutual Funds data, and building the corresponding linear model named mutual_vif_model by

calling the lm function. Now we can generate a correlation plot Figure 22 to check whether the multinearity has been reduced as well. Compared to the correlation plot of Mutual Funds using stepwise AIC method, the correlations in the variables have also been greatly controlled in a desirable level.

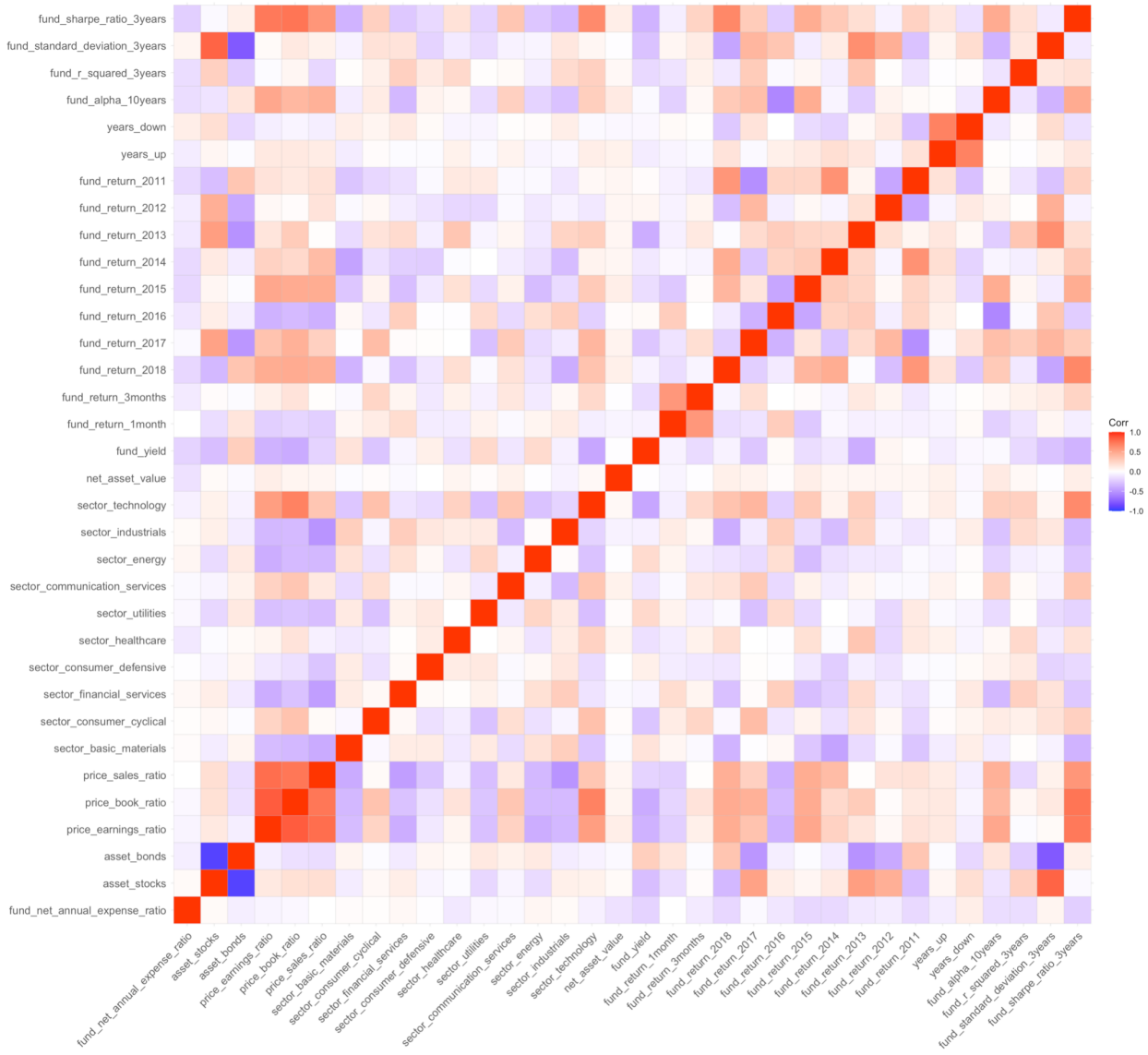


Figure 22. The correlation of stepwise VIF regression model of Mutual Funds

Now we can check each explanatory variable in anova table Figure 23 by calling anova function. It is obvious that most of the variables have high significance levels by checking stars signs after each row, which is much improved over the result of the stepwise AIC method .

```
> anova(mutual_vif_model)
Analysis of Variance Table

Response: fund_return_2019
                               Df Sum Sq Mean Sq   F value    Pr(>F)
rating                          1  16077   16077 1748.0850 < 2.2e-16 ***
return_rating                   1   4738    4738  515.1517 < 2.2e-16 ***
risk_rating                     1   2402    2402  261.2067 < 2.2e-16 ***
fund_net_annual_expense_ratio   1   1542    1542  167.6485 < 2.2e-16 ***
asset_cash                      1  62191   62191 6761.9945 < 2.2e-16 ***
asset_bonds                     1  82183   82183 8935.7622 < 2.2e-16 ***
asset_others                    1   1175    1175  127.7640 < 2.2e-16 ***
asset_preferred                 1    134     134   14.5649 0.0001367 ***
asset_convertable               1    571     571   62.1050 3.808e-15 ***
price_earnings_ratio            1  32728   32728 3558.4973 < 2.2e-16 ***
price_sales_ratio               1    537     537   58.4010 2.456e-14 ***
median_market_cap               1   4441    4441  482.8231 < 2.2e-16 ***
sector_basic_materials          1    831     831   90.3518 < 2.2e-16 ***
sector_consumer_cyclical        1     85      85    9.2286 0.0023924 **
sector_financial_services       1   1109    1109  120.6011 < 2.2e-16 ***
sector_consumer_defensive       1   1385    1385  150.6386 < 2.2e-16 ***
sector_healthcare               1      0       0    0.0405 0.8405200
sector_utilities                1    253     253   27.5120 1.612e-07 ***
sector_communication_services   1     80      80    8.6958 0.0032011 **
sector_energy                   1     17      17    1.8556 0.1731798
sector_industrials              1    949     949  103.1375 < 2.2e-16 ***
sector_technology               1    319     319   34.6312 4.189e-09 ***
bond_maturity                   1    242     242   26.3560 2.924e-07 ***
bond_duration                   1   2324    2324  252.7047 < 2.2e-16 ***
net_asset_value                 1     76      76    8.2570 0.0040731 **
fund_yield                      1   1019    1019  110.7706 < 2.2e-16 ***
fund_return_1month              1    209     209   22.6956 1.940e-06 ***
fund_return_3months             1    754     754   81.9691 < 2.2e-16 ***
fund_return_2018                1    161     161   17.4632 2.969e-05 ***
fund_return_2017                1    680     680   73.9812 < 2.2e-16 ***
fund_return_2016                1   4060    4060  441.4195 < 2.2e-16 ***
fund_return_2015                1   1002    1002  108.9611 < 2.2e-16 ***
fund_return_2014                1   1899    1899  206.4922 < 2.2e-16 ***
fund_return_2013                1   2697    2697  293.2812 < 2.2e-16 ***
fund_return_2012                1    874     874   95.0567 < 2.2e-16 ***
fund_return_2011                1      5       5    0.5033 0.4780653
fund_return_2010                1    601     601   65.3509 7.452e-16 ***
years_up                        1     53      53    5.8051 0.0160080 *
years_down                      1      1       1    0.0618 0.8036953
quarters_up                     1   3107    3107  337.8340 < 2.2e-16 ***
fund_alpha_10years              1     77      77    8.4211 0.0037217 **
fund_beta_3years                1    818     818   88.9576 < 2.2e-16 ***
fund_r_squared_3years           1    483     483   52.5232 4.759e-13 ***
fund_sharpe_ratio_3years        1  15682   15682 1705.1120 < 2.2e-16 ***
fund_treynor_ratio_5years       1    206     206   22.3844 2.280e-06 ***
Residuals                    6317  58098       9
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 23. Anova table of mutual_vif_model

The linear model has been standardized by calling lm.beta function so that we can compare beta values in the model. The explanatory variables with top 20 positive beta coefficients and variables with top 20 negative variables are listed in Figure 24.

```
> standardized_mutual_vif$standardized.coefficients%>% sort(decreasing = TRUE) %>% head(20)
 fund_sharpe_ratio_3years               quarters_up          fund_return_2014
               0.58835617                0.08464501                0.07561459
         fund_return_2017          fund_return_2016          fund_beta_3years
               0.07162378                0.07143278                0.07100251
        price_sales_ratio          fund_return_2013          fund_return_2012
               0.07020324                0.06484921                0.06384600
           bond_maturity         median_market_cap      price_earnings_ratio
               0.06347264                0.06274335                0.05969557
sector_financial_services               risk_rating        sector_industrials
               0.04493363                0.04346821                0.04023650
         sector_utilities          fund_return_2015             return_rating
               0.03883088                0.03860879                0.03715587
fund_treynor_ratio_5years          fund_return_2010
               0.02990051                0.02940403
> standardized_mutual_vif$standardized.coefficients%>% sort(decreasing = FALSE) %>% head(20)
                  asset_bonds              fund_return_2018                    asset_cash
                -0.375685217                -0.247157581                -0.121573492
               bond_duration      sector_consumer_cyclical             asset_convertable
                -0.086620418                -0.085201503                -0.078771692
           sector_technology           fund_return_1month                      rating
                -0.067749662                -0.060284277                -0.046261227
           sector_healthcare           fund_alpha_10years sector_communication_services
                -0.031840108                -0.026967854                -0.025507289
             net_asset_value        fund_r_squared_3years fund_net_annual_expense_ratio
                -0.021545283                -0.020649683                -0.016839524
      sector_basic_materials               sector_energy                    years_up
                -0.014792522                -0.011746051                -0.011611244
   sector_consumer_defensive          fund_return_3months
                -0.009809607                -0.002790912
```

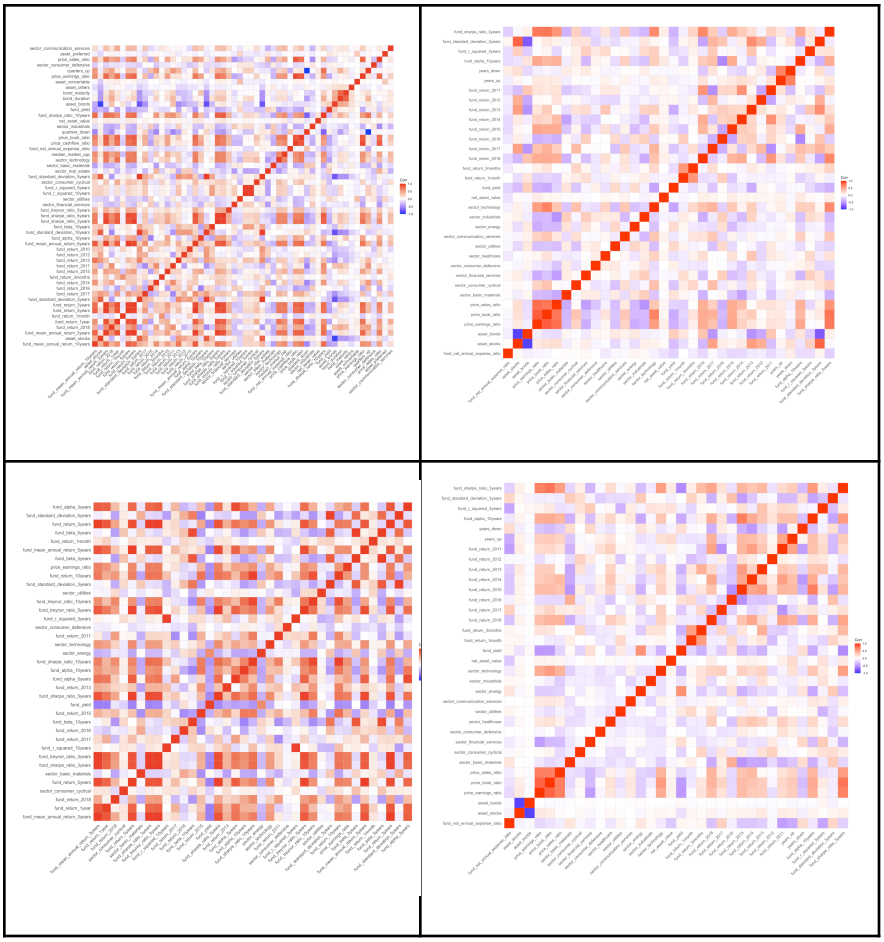
Figure 24. Top 20 highest and lowest beta values of Mutual funds using stepwise VIF method

Similarly, we can analyze and summarize this list and propose some recommendations from Figure 24.

4. Recommendation and Conclusion

Recall the adjusted R-squared of AIC model and INF model of Mutual Funds and ETFs from Section 3 and the correlation plots of each model:

	AIC	VIF
Mutual Funds:	0.9564	0.8106
ETFs:	0.9043	0.6946



Now recall our Question: For mutual funds and ETFs, what are the differences between these two? How should we choose them? Our goal is to compare the parameter of predictors to answer this question. As we can see, both adjusted R-squared values of stepwise VIF method are lower than the values of stepwise AIC method, but the correlation plots of the VIF method model of the

two funds performed significantly better than the ones of the AIC model. We discovered that the low adjusted R-squared of VIF model of ETFs is because of the small data set. Since the adjusted R-squared of the VIF model for Mutual Funds is 0.8106 which is good, we can assume that the VIF model would perform better with more data. That means our recommendation for ETFs has limitations and risks: The premise is that we assume that the VIF model will perform better on a larger data set of ETFs. Since we aim to compare the Standardized Beta of each predictors, we believe the performance of correlation plots of each method would have more impact on the analysis of the value of the Standardized Beta and the recommendation that we will give. Stepwise VIF method can significantly reduce multicollinearity and produce a more accurate beta value. This advantage of the VIF method is more related to our question, so we decided to use stepwise VIF Selection.

The most important positive indicator to select funds: Sharpe Ratio 3 years

According to Figure 21 and Figure 24, it is obvious that the two values of fund_sharpe_ratio_3years rank first on both top 20 positive beta values of two methods and these two values are relatively much greater than any other values on both two lists. So it can be concluded that values of Sharpe Ratio 3 years are of most importance when choosing either Mutual Funds or ETFs. The greater the Sharpe Ratio 3 years values, the more possible to get lucrative returns. The definition of Sharpe Ratio is the average return earned in excess of the risk-free rate per unit of volatility or total risk. So we can further conclude that do not be too afraid to buy funds with high volatility or high risk, because the return of these high volatility may largely exceed the loss caused by volatility.

The deceptive indicator: Fund return 2018

As we can see, the indicator fund_return_2018 ranks the second and the first in the top 20 negative beta values of mutual funds and ETfs respectively. Here is what happened on the 2018 stock market: *"US stocks post worst year in a decade as the S&P 500 falls more than 6% in 2018"* (Fred) This is the news title on CNBC written by Fred Imbert on Dec 3, 2018. The message from this news is that the U.S. stock market was in a very bad place in 2018. But in

2019, the U.S. stock market is in a completely different place than it was in 2018. *"The market boomed in 2019, with major indexes hitting numerous record highs as stocks posted their best annual return in six years, thanks to the U.S. economy's moderate expansion holding steady and renewed trade optimism on Wall Street."*(Sergei) This is a topline of finance news on Forbes. According to these news, 2018 was the year of market collapse, and then 2019 was the year of boom, so the return rate of 2018 and 2019 showed obvious negative correlation, which is why the indicator fund_return_2018 ranks the top two among the top 20 beta values. However, this phenomenon cannot be used as the reference indicator for our recommendation, so we choose to ignore the beta value of fund_return_2018.

Advice on choosing Mutual Funds

We can choose Mutual Funds with high ratios of finance services, industrials and utilities in sectors, because these sectors have positive beta values so will have positive impacts on Funds annual returns. However we can avoid those funds with high ratios of consumer cyclical, technology, healthcare, communication services, basic materials and energy, because these sectors have negative beta values so will have negative impacts on Funds annual return.

Besides, Mutual Funds with high ratios of bonds and cash in their assets should be avoided as much as possible. We can see that asset_bonds and asset_cash are in the first and third place of the top 20 negative beta values on Figure 24. So the more ratios of bonds and cash in total assets of funds, the more likely to lose money, because these sectors have negative beta values so will have negative impacts on Funds annual return.

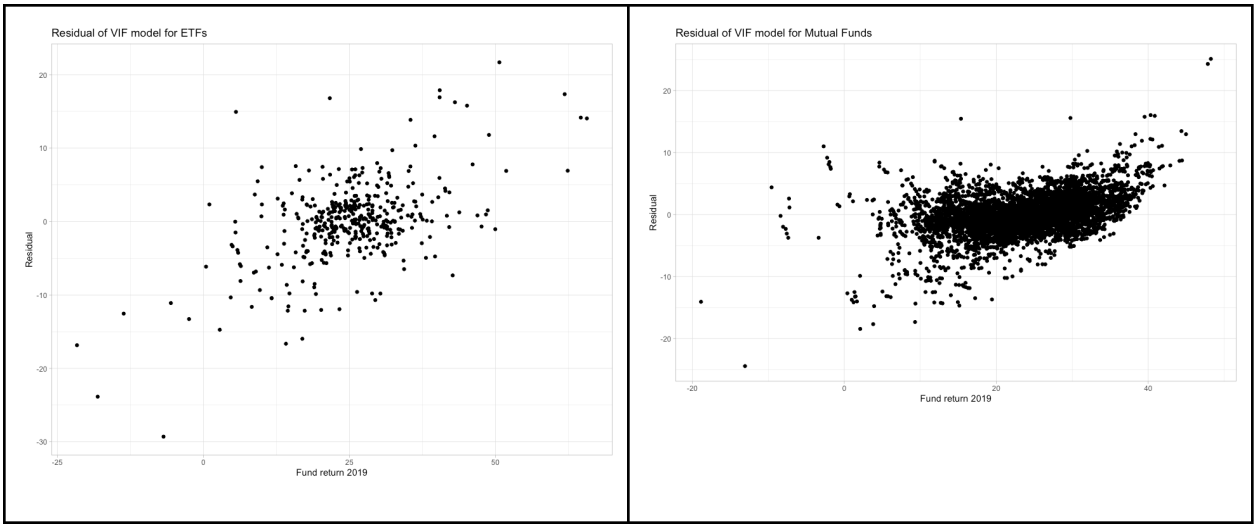
Advice on choosing ETFs

We can choose ETFs with high ratios of technology and utilities, which have positive beta values so will have positive impacts on Funds annual returns. However, we can avoid those funds with

high ratios of communication services, energy, basic materials, consumer cyclical, healthcare, consumer defensive and industrials, because these sectors have negative beta values so will have negative impacts on Funds annual return.

In addition, ETFs with high Price-To-Book ratios can be avoided as much as possible. It is clearly that the price_book_ratio ranks second on the top 20 negative beta values on Figure 21. Price-To-Book ratio is calculated by dividing the company's stock price per share by its book value per share. So the higher the P/B ratio, the more bubbles the company will have. So it really makes sense that we should avoid funds holding too many stocks of companies that have too many bubbles.

Final words to investor



Here are two residual plots for our VIF model of each fund. As we can see, our model is not that accurate based on the residual, so it may lead to inaccuracies and uncertainty in beta values. But remember our primary goal with this project is not to predict total returns of any specific funds, but to explore the similarities and differences between mutual funds and ETFs and make recommendations to potential investors like us. We are more satisfied with the Stepwise VIF method because it's closer to the problem that we started with.

Work Cited

Imbert, Fred. “US Stocks Post Worst Year in a Decade as the S&P 500 Falls More than 6% in 2018,” January 7, 2019. https://www.cnbc.com/2018/12/31/stock-market-wall-street-stocks-eye-us-china-trade-talks.html .

Klebnikov, Sergei. “Here's How 2019 Turned Out To Be A Historic Year For The Stock Market.” Forbes. Forbes Magazine, December 16, 2020. https://www.forbes.com/sites/sergeiklebnikov/2020/01/01/heres-how-2019-turned-out-to-be-a-historic-year-for-the-stock-market/?sh=471739498c47.

```
#                          Appendix                                    #

#######################   Set Up     ################################
library(tidyverse)
library(reshape2)
library(corrplot)
library(ggplot2)
theme_set(theme_light())
mutual_funds_data = read.csv("~/Downloads/Mutual
Funds.csv",na.strings=c("",NA))
etfs_data = read.csv("~/Downloads/ETFs.csv",na.strings=c("",NA))
#mutual_funds_data = read.csv("/Users/david/Downloads/archive-2/Mutual
Funds.csv",na.strings=c("",NA))
#etfs_data = read.csv("/Users/david/Downloads/archive-2/
ETFs.csv",na.strings=c("",NA))

######################## Data Wrangling ##############################

## Select the intersection columns of two data sets and drop the columns
that are not useful for our project
#  and add a column called fund_type
mutual_funds = mutual_funds_data %>%
  select(intersect(names(mutual_funds_data), names(etfs_data))) %>%
  select(-fund_symbol, -fund_extended_name, -fund_family, -inception_date,
         -category, -investment_strategy, -currency, -top10_holdings) %>%
  mutate(fund_type = "mutual funds")

etfs = etfs_data %>%
  select(intersect(names(mutual_funds_data), names(etfs_data)))%>%
  select(-fund_symbol, -fund_extended_name, -fund_family, -inception_date,
         -category, -investment_strategy, -currency, -top10_holdings) %>%
  mutate(fund_type = "etfs")

## Rbind datasets
both_funds = rbind(mutual_funds, etfs)



######################## Data Visualization ##########################
## Barplot of the number of each investment type with different size type
of Mutual Fund
ggplot(mutual_funds %>% filter(!is.na(investment_type)),
       aes(x = investment_type,
           fill = size_type)) + geom_bar()+
  labs(fill = "Size Type" ,
       title="The number of each investment type with different size type
of Mutual Fund",
       y="Total Count",
       x= "Investment Type")+
  theme(
    legend.title = element_text(face = "bold")
  )
```

```
## Barplot of the number of each investment type with different size type
of ETFs
ggplot(etfs %>% filter(!is.na(investment_type)),
       aes(x = investment_type,
           fill = size_type)) + geom_bar()+
  labs(fill = "Size Type" ,
       title="The number of each investment type with different size type
of ETFs",
       y="Total Count",
       x= "Investment Type")+
  theme(
    legend.title = element_text(face = "bold")
  )

## The density plot of each variables with corresponding investment type of
Mutual Funds
melt_mutual = melt(mutual_funds %>% filter(!is.na(investment_type)))
ggplot(melt_mutual, aes(
  x = value, color = investment_type, fill = investment_type)) +
  stat_density() +
  facet_wrap(~variable, scales = "free") +
  scale_fill_brewer(palette="Pastel1")+
  labs(title="Density distribution of each variables with corresponding
investment type of Mutual Funds regarding the values",
       y="Density",
       x= "Value")+ scale_fill_discrete(name='Investment Type')+
  scale_color_discrete(name='Investment Type')+
  theme(
    legend.title = element_text(face = "bold")
  )


## The density plot of each variables with corresponding investment type of
ETFs
melt_etfs = melt(etfs %>% filter(!is.na(investment_type)))
ggplot(melt_etfs, aes(
  x = value, color = investment_type, fill = investment_type)) +
  stat_density() +
  facet_wrap(~variable, scales = "free") +
  scale_fill_brewer(palette="Pastel1")+
  labs(title="Density distribution of each variables with corresponding
investment type of ETFs regarding the values",
       y="Density",
       x= "Value")+ scale_fill_discrete(name='Investment Type')+
  scale_color_discrete(name='Investment Type')+
  theme(
    legend.title = element_text(face = "bold")
  )




## Sector shows how much funds invest in specific areas.
```

```
#### This version provides a clearer comparision of distriibution betweeen
ETFs and Mutual_funds of each sector
mutual_funds_sector = mutual_funds %>% select(starts_with("sector"))
mutual_funds_sector = na.omit(mutual_funds_sector)/100
mutual_funds_sector = mutual_funds_sector %>% mutate(type= "mutual_funds")

etfs_sector = etfs %>% select(starts_with("sector"))
etfs_sector = na.omit(etfs_sector)/100
etfs_sector = etfs_sector %>% mutate(type= "etfs")

both_funds_sector = rbind(mutual_funds_sector, etfs_sector)
both_funds_sector_new= melt(both_funds_sector,id.vars="type")

# Boxplot of comparision of distriibution betweeen ETFs and Mutual_funds of
each sector
compare <- ggplot(data = both_funds_sector_new, aes(x=variable, y=value)) +
  geom_boxplot(aes(fill=type))+
  labs(title="Comparision of distribution betweeen ETFs and Mutual_funds of
each sector",
       y="Value", x="Type of sector")+
  theme(legend.title = element_text(face = "bold"))+
  scale_fill_discrete(name='Investment Type',labels = c("ETFs", "Mutual
Funds"))

comparison_sector = compare + facet_wrap( ~ variable, scales="free")



#### This version provides a clearer comparison of the distribution of EACH
SECTOR
comparison_eachfunds = ggplot(data = both_funds_sector_new, aes(x=variable,
y=value)) +
  geom_boxplot(aes(fill=type))+coord_flip()+
  labs(title="Comparison of the distribution of EACH SECTOR",
       y="Value", x="Type of sector")+
  theme(legend.title = element_text(face = "bold"))+
  scale_fill_discrete(name='Investment Type',labels = c("ETFs", "Mutual
Funds"))


#### This version provides a comparision of overall distribution of each
sector
comparison_overall = ggplot(data = both_funds_sector_new, aes(x=variable,
y=value)) +
  geom_boxplot()+coord_flip()+labs(title="Overall distribution of each
sector",
                                   y="Value", x="Type of sector")


#########################################################################
######################    AIC Method Modeling   ######################
```

```
######################   Model For Mutual Funds (AIC)
######################

## Only select the numerical columns in Mutual Funds Data and add the
investment_type and size_type columns.
mutual_reg_data = mutual_funds_data %>%
  select(where(is.numeric)) %>%
  mutate(investment_type = mutual_funds_data$investment_type,
         size_type = mutual_funds_data$size_type)

## Lists the number of NAs of each columns using for loop
y = c()
for (i in seq_along(names(mutual_reg_data))) {
  x = sum(is.na(mutual_reg_data[,i]))
  y = c(y, x)
}
names(y) = names(mutual_reg_data)
head(sort(y, decreasing = TRUE, 20), 50)


## Drop the columns with more NAs and create a new mutual_reg_data_new.
mutual_reg_data_new = mutual_funds_data %>%
  select(where(is.numeric)) %>%
  mutate(investment_type = mutual_funds_data$investment_type,
         size_type = mutual_funds_data$size_type) %>%
  select(-starts_with("credit"),-starts_with("category"), -ends_with("q1"),
-ends_with("q2"),
         -ends_with("q3"), -ends_with("q4"), -ends_with("ytd")) %>%
  na.omit()


## Select the optimized parameter and automaticaly optimized by
ols_step_both_aic() function.
library(olsrr)
full_model = lm(fund_return_2019~.,data = mutual_reg_data_new)
step_aic_both = ols_step_both_aic(full_model, details = TRUE)
step_aic_both$predictors

## Pick the parameter from the results and create the model
mutual_model = lm(fund_return_2019 ~ fund_mean_annual_return_10years +
fund_mean_annual_return_3years +
                    fund_return_2018 + fund_return_1year +
fund_return_3years + fund_standard_deviation_3years +
                    fund_return_2017 + fund_return_2016 + fund_return_2014
+ investment_type +
                    fund_return_3months + fund_return_2015 +
fund_return_2011 + fund_return_2013 +
                    fund_return_2012 + fund_return_2010 +
fund_mean_annual_return_5years +
                    fund_alpha_10years + fund_standard_deviation_10years +
size_type +
                    fund_beta_10years + fund_sharpe_ratio_3years +
fund_sharpe_ratio_5years +
```

```
                    fund_treynor_ratio_5years + sector_financial_services +
sector_utilities +
                    fund_r_squared_10years + fund_r_squared_5years +
sector_consumer_cyclical +
                    fund_standard_deviation_5years + sector_real_estate +
sector_basic_materials +
                    sector_technology + median_market_cap +
fund_net_annual_expense_ratio +
                    price_cashflow_ratio + price_book_ratio + quarters_down
+ sector_industrials +
                    net_asset_value + fund_sharpe_ratio_10years +
fund_yield + asset_bonds +
                    bond_duration + bond_maturity + asset_others +
asset_convertable +
                    price_earnings_ratio + quarters_up +
sector_consumer_defensive +
                    price_sales_ratio + asset_preferred +
sector_communication_services,
                  data = mutual_reg_data_new)

library(lm.beta)
lm.beta(mutual_model)
standardized_mutual_beta = lm.beta(mutual_model)

## Top 30 highest parameter coefficients
standardized_mutual_beta$standardized.coefficients %>%
  unlist() %>% sort(decreasing = TRUE) %>% head(20)

## Top 30 lowest parameter coefficients
standardized_mutual_beta$standardized.coefficients %>%
  unlist() %>% sort(decreasing = FALSE) %>% head(20)

## Compute analysis of variance tables for the fitted model.
summary(mutual_model)  #R-squared:  0.9568, Adjusted R-squared:  0.9564
anova(mutual_model)


## Correlations of each predictors
cor_mutual = cor(mutual_reg_data_new %>% select(step_aic_both$predictors)
%>% select(-investment_type, -size_type)) %>% as.matrix()
library(ggcorrplot)
ggcorrplot(cor_mutual)


## Add predictions of the model and the abline.
library(modelr)
mutual_reg_data_new_prediction = mutual_reg_data_new %>%
  add_predictions(mutual_model) %>%
  mutate(resid = fund_return_2019 - pred)
ggplot(mutual_reg_data_new_prediction, aes(fund_return_2019, pred)) +
  geom_point()+
  geom_abline(slope = 1, intercept = 0, color = "red")

## Plot the residuals of the model
```

```
ggplot(mutual_reg_data_new_prediction, aes(fund_return_2019, resid)) +
  geom_point()

plot(mutual_model$residuals)

## Nomal QQ Plot of residuals
qqnorm(mutual_model$residuals)
qqline(mutual_model$residuals)



######################   Model For ETFs (AIC) ######################

## Only select the numerical columns in ETFs and add the investment_type
and size_type columns.
etfs_reg_data = etfs_data %>%
  select(where(is.numeric)) %>% mutate(investment_type =
etfs_data$investment_type,
                                       size_type = etfs_data$size_type)

## Lists the number of NAs of each columns using for loop
y = c()
for (i in seq_along(names(etfs_reg_data))) {
  x = sum(is.na(etfs_reg_data[,i]))
  y = c(y, x)
}
names(y) = names(etfs_reg_data)
head(sort(y, decreasing = TRUE, 20), 50)

## Drop the columns with more NAs and create a new etfs_reg_data_new
etfs_reg_data_new = etfs_data %>%
  select(where(is.numeric)) %>%
  mutate(investment_type = etfs_data$investment_type,
         size_type = etfs_data$size_type) %>%
  select(-starts_with("credit"),-starts_with("category"), -
ends_with("2010"),
         -ends_with("ytd")) %>%
  na.omit()

## Select the optimized parameter and automaticaly optimized by
ols_step_both_aic() function.
full_model2 = lm(fund_return_2019~.,data = etfs_reg_data_new)
step_aic_both2 = ols_step_both_aic(full_model2, details = TRUE)
step_aic_both2$predictors

## Pick the parameter from the results and create the model
etfs_model = lm(fund_return_2019 ~ fund_return_1year + fund_return_2018 +
fund_return_5years +
                  fund_treynor_ratio_3years + fund_return_2017 +
fund_return_2016 +
                  fund_return_2015 + fund_yield + fund_sharpe_ratio_5years
+ fund_return_2013 +
                  fund_alpha_5years + fund_alpha_10years +
fund_sharpe_ratio_10years +
```

```
                  sector_energy + sector_technology + fund_return_2011 +
sector_consumer_defensive +
                  fund_treynor_ratio_5years + fund_treynor_ratio_10years +
sector_utilities +
                  fund_standard_deviation_3years + fund_return_10years +
price_earnings_ratio +
                  fund_beta_3years + fund_mean_annual_return_5years +
fund_return_1month +
                  fund_beta_5years + fund_return_3years +
fund_standard_deviation_5years +
                  fund_alpha_3years, data = etfs_reg_data_new)

standardized_etfs_beta = lm.beta(etfs_model)


## Top 30 highest parameter coefficients
standardized_etfs_beta$standardized.coefficients %>% sort(decreasing =
TRUE) %>% head(20)

## Top 30 lowest parameter coefficients
standardized_etfs_beta$standardized.coefficients %>% sort(decreasing =
FALSE) %>% head(20)

## Compute analysis of variance tables for the fitted model.
summary(etfs_model) #Multiple R-squared:  0.9111, Adjusted R-squared:
0.9043
anova(etfs_model)




## Correlations of each predictors
cor_etfs = cor(etfs_reg_data_new %>% select(step_aic_both2$predictors)) %>%
as.matrix()
ggcorrplot(cor_etfs)

## Add predictions of the model and the abline.
etfs_reg_data_new_prediction = etfs_reg_data_new %>%
  add_predictions(etfs_model) %>%
  mutate(resid = fund_return_2019 - pred)

ggplot(etfs_reg_data_new_prediction, aes(fund_return_2019, pred)) +
  geom_point()+
  geom_abline(slope = 1, intercept = 0, color = "red")

## Plot the residuals of the model
ggplot(etfs_reg_data_new_prediction, aes(fund_return_2019, resid))+
  geom_point()

## Nomal QQ Plot of residuals
qqnorm(etfs_model$residuals)
qqline(etfs_model$residuals)
```

```
#########################################################################
######################    VIF Method Modeling   ######################

## VIF function
vif_func<-function(in_frame,thresh=10,trace=T,...){
  library(fmsb)
  if(any(!'data.frame' %in% class(in_frame))) in_frame<-
data.frame(in_frame)
  #get initial vif value for all comparisons of variables
  vif_init<-NULL
  var_names <- names(in_frame)
  for(val in var_names){
    regressors <- var_names[-which(var_names == val)]
    form <- paste(regressors, collapse = '+')
    form_in <- formula(paste(val, '~', form))
    vif_init<-rbind(vif_init, c(val, VIF(lm(form_in, data = in_frame,
...))))
  }
  vif_max<-max(as.numeric(vif_init[,2]), na.rm = TRUE)
  if(vif_max < thresh){
    if(trace==T){ #print output of each iteration
      prmatrix(vif_init,collab=c('var','vif'),rowlab=rep('',nrow(vif_init)),quote=F)
      cat('\n')
      cat(paste('All variables have VIF < ', thresh,', max VIF
',round(vif_max,2), sep=''),'\n\n')
    }
    return(var_names)
  }
  else{
    in_dat<-in_frame
    #backwards selection of explanatory variables, stops when all VIF
values are below 'thresh'
    while(vif_max >= thresh){
      vif_vals<-NULL
      var_names <- names(in_dat)
      for(val in var_names){
        regressors <- var_names[-which(var_names == val)]
        form <- paste(regressors, collapse = '+')
        form_in <- formula(paste(val, '~', form))
        vif_add<-VIF(lm(form_in, data = in_dat, ...))
        vif_vals<-rbind(vif_vals,c(val,vif_add))
      }
      max_row<-which(vif_vals[,2] == max(as.numeric(vif_vals[,2]), na.rm =
TRUE))[1]
      vif_max<-as.numeric(vif_vals[max_row,2])
      if(vif_max<thresh) break
      if(trace==T){ #print output of each iteration
        prmatrix(vif_vals,collab=c('var','vif'),rowlab=rep('',nrow(vif_vals)),quote=F)
        cat('\n')
        cat('removed: ',vif_vals[max_row,1],vif_max,'\n\n')
        flush.console()
      }
      in_dat<-in_dat[,!names(in_dat) %in% vif_vals[max_row,1]]
    }
```

```
    return(names(in_dat))
  }
}


#####################   Model For ETFs (VIF) ######################

etfs_reg_data_new_vif_without_y = etfs_data %>%
  select(where(is.numeric))  %>%
  select(-starts_with("credit"),-starts_with("category"), -
ends_with("2010"),
         -ends_with("ytd")) %>% na.omit() %>% select(-fund_return_2019)

etfs_reg_data_new_vif_with_y = etfs_data %>%
  select(where(is.numeric))  %>%
  select(-starts_with("credit"),-starts_with("category"), -
ends_with("2010"),
         -ends_with("ytd")) %>% na.omit()

keep_dat = vif_func(in_frame = etfs_reg_data_new_vif_without_y, thresh =
10, trace = T)
etfs_vif_model = lm(paste('fund_return_2019
~',paste(keep_dat,collapse='+')),
                    data = etfs_reg_data_new_vif_with_y)

standardized_etfs_vif = lm.beta(etfs_vif_model)
standardized_etfs_vif$standardized.coefficients%>% sort(decreasing = TRUE)
%>% head(20)
standardized_etfs_vif$standardized.coefficients%>% sort(decreasing = FALSE)
%>% head(20)

summary(etfs_vif_model) # Multiple R-squared:  0.7192, Adjusted R-squared:
0.6946
anova(etfs_vif_model)

cor_etfs_vif = cor(etfs_reg_data_new_vif_without_y %>% select(keep_dat))
%>% as.matrix()
ggcorrplot(cor_etfs_vif)

etfs_reg_data_new_vif_with_y_prediction = etfs_reg_data_new_vif_with_y %>%
  add_predictions(etfs_vif_model)%>%
  mutate(resid = fund_return_2019 - pred)

ggplot(etfs_reg_data_new_vif_with_y_prediction, aes(fund_return_2019,
pred)) +
  geom_point()+
  geom_abline(slope = 1, intercept = 0, color = "red")

ggplot(etfs_reg_data_new_vif_with_y_prediction, aes(fund_return_2019,
resid))+
  geom_point()+
  labs(title="Residual of VIF model for ETFs",
       y="Residual",
       x= "Fund return 2019")
```

```
qqnorm(etfs_vif_model$residuals)
qqline(etfs_vif_model$residuals)


######################   Model For Mutual Funds (VIF)
######################
mutual_reg_data_new_vif_with_y = mutual_funds_data %>%
  select(where(is.numeric))  %>%
  select(-starts_with("credit"),-starts_with("category"), -ends_with("q1"),
-ends_with("q2"),
         -ends_with("q3"), -ends_with("q4"), -ends_with("ytd")) %>%
  na.omit()

mutual_reg_data_new_vif_without_y = mutual_funds_data %>%
  select(where(is.numeric)) %>%
  select(-starts_with("credit"),-starts_with("category"), -ends_with("q1"),
-ends_with("q2"),
         -ends_with("q3"), -ends_with("q4"), -ends_with("ytd")) %>%
  na.omit() %>% select(-fund_return_2019)

keep_dat_mutual = vif_func(in_frame = mutual_reg_data_new_vif_without_y,
thresh = 10, trace = T)
mutual_vif_model = lm(paste('fund_return_2019
~',paste(keep_dat_mutual,collapse='+')),
                      data = mutual_reg_data_new_vif_with_y)
standardized_mutual_vif = lm.beta(mutual_vif_model)
standardized_mutual_vif$standardized.coefficients%>% sort(decreasing =
TRUE) %>% head(20)
standardized_mutual_vif$standardized.coefficients%>% sort(decreasing =
FALSE) %>% head(20)

summary(mutual_vif_model) # Multiple R-squared:  0.8119, Adjusted R-
squared:  0.8106
anova(mutual_vif_model)

cor_mutual_vif = cor(mutual_reg_data_new_vif_without_y %>%
select(keep_dat)) %>% as.matrix()
ggcorrplot(cor_mutual_vif)

mutual_reg_data_new_vif_with_y_prediction = mutual_reg_data_new_vif_with_y
%>%
  add_predictions(mutual_vif_model)%>%
  mutate(resid = fund_return_2019 - pred)

ggplot(mutual_reg_data_new_vif_with_y_prediction, aes(fund_return_2019,
pred)) +
  geom_point()+
  geom_abline(slope = 1, intercept = 0, color = "red")

ggplot(mutual_reg_data_new_vif_with_y_prediction, aes(fund_return_2019,
resid))+
  geom_point()+
  labs(title="Residual of VIF model for Mutual Funds",
       y="Residual",
```

```
       x= "Fund return 2019")
qqnorm(etfs_vif_model$residuals)
qqline(etfs_vif_model$residuals)
```