CIS 600 Final Project - Digit Recognition

Gafei, Greg Shin, Ashraf Elnashar

Nov. 11th, 2019

Introduction

The ultimate goal in creating a stock portfolio is to gather a group of stocks that will increase in value overtime. However, it is often difficult to pick which stocks to invest in from all the ones that are available.Therefore, we intend to explore the use of clustering analysis in identifying groups of similarly performingstocks.

Goal

To use clustering techniques with historical stock prices to group different stocks together, in orderto form an investment strategy.

Load Libraries

```
#Function that loads libraries
EnsurePackage <- function(x) {
  x <- as.character(x)
  if (!require(x,character.only = T))
    install.packages(x,repos = "https://cran.rstudio.com/")
  require(x,character.only = T)
}

EnsurePackage("caret") # set of functions that attempt to streamline the process for creating predictiv
```

```
## Loading required package: caret
## Loading required package: lattice
## Loading required package: ggplot2
```

```
EnsurePackage("rpart") #Recursive Partitioning And Regression Trees
```

```
## Loading required package: rpart
```

```
EnsurePackage("DMwR") #Smote
```

```
## Loading required package: DMwR
## Loading required package: grid
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
EnsurePackage("rattle") #graphical user interface to many other R packages that provide functionality f
```

```
## Loading required package: rattle
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
EnsurePackage("tidyverse") #Manipulating dataset
```

```
## Loading required package: tidyverse

## -- Attaching packages -------------------------------------------------------------------------

## v tibble  2.1.3     v purrr   0.3.2
## v tidyr   1.0.0     v dplyr   0.8.3
## v readr   1.3.1     v stringr 1.4.0
## v tibble  2.1.3     v forcats 0.4.0

## -- Conflicts -------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```r
EnsurePackage("ggplot2")
EnsurePackage("readr")
EnsurePackage("dplyr") #selecting data
EnsurePackage("magrittr") #using pipe operators
```

```
## Loading required package: magrittr

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
EnsurePackage("corrplot")
```

```
## Loading required package: corrplot

## corrplot 0.84 loaded
```

```r
EnsurePackage("knitr")
```

```
## Loading required package: knitr
```

```r
EnsurePackage("sm")
```

```
## Loading required package: sm

## Package 'sm', version 2.2-5.6: type help(sm) for summary information

##
## Attaching package: 'sm'

## The following object is masked from 'package:rattle':
##
##     binning
```

```r
EnsurePackage("gmodels")
```

```
## Loading required package: gmodels
```

```r
EnsurePackage("rpart") #Recursive Partitioning and Regression Trees
EnsurePackage("rpart.plot")
```

```
## Loading required package: rpart.plot
```

```r
EnsurePackage("plotly")
```

```
## Loading required package: plotly

##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```r
EnsurePackage("e1071") #deals with Probability group theory functions
```

```
## Loading required package: e1071
```

```r
EnsurePackage("RColorBrewer")  #coloring of graphs
```

```
## Loading required package: RColorBrewer
```

```r
EnsurePackage("plotly")
EnsurePackage("cluster")    # clustering algorithms
```

```
## Loading required package: cluster
```

```r
EnsurePackage("dendextend") # for comparing two dendrograms
```

```
## Loading required package: dendextend

##
## ---------------------
## Welcome to dendextend version 1.13.2
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------

##
## Attaching package: 'dendextend'

## The following object is masked from 'package:rpart':
##
```

```
##     prune
## The following object is masked from 'package:stats':
##
##     cutree
```

```
EnsurePackage("stats")
EnsurePackage("pacman")
```

```
## Loading required package: pacman
```

```
EnsurePackage("factoextra")
```

```
## Loading required package: factoextra
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
EnsurePackage("reshape2")
```

```
## Loading required package: reshape2
##
## Attaching package: 'reshape2'
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
EnsurePackage("tidyr")
EnsurePackage("textshape")
```

```
## Loading required package: textshape
##
## Attaching package: 'textshape'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:purrr':
##
##     flatten
## The following object is masked from 'package:tibble':
##
##     column_to_rownames
```

Load Data

```
nRowsRead = 1000 # specify 'None' if want to read whole file
# dataset_summary.csv has 7091 rows in reality, but we are only loading/previewing the first 1000 row

path <- 'dataset_summary.csv'
dataSetReader_Summary <-read.csv(path, nrows = nRowsRead)
#Summary of Stock Prices.
str(dataSetReader_Summary)
```

```
## 'data.frame':    1000 obs. of  7 variables:
##  $ symbol           : Factor w/ 1000 levels "A","AA","AAAP",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ total_prices     : int  4962 697 574 5434 1222 3489 1675 5434 5436 1476 ...
```

```
##  $ stock_from_date   : Factor w/ 593 levels "1998-01-02","1998-01-20",..: 37 458 416 1 365 145 299 1
##  $ stock_to_date     : Factor w/ 71 levels "2018-01-30","2018-02-12",..: 69 69 16 67 69 67 69 67 69 6
##  $ total_earnings    : int  42 11 0 14 21 23 23 39 41 24 ...
##  $ earnings_from_date: Factor w/ 351 levels "2009-04-16","2009-04-23",..: 13 281 351 166 216 166 147
##  $ earnings_to_date  : Factor w/ 149 levels "2015-05-11","2015-07-16",..: 143 123 149 7 83 129 35 39
```

```r
#Head of Summary of Stock.
head(dataSetReader_Summary, 5)
```

```
##   symbol total_prices stock_from_date stock_to_date total_earnings
## 1      A         4962      1999-11-18    2019-08-09             42
## 2     AA          697      2016-11-01    2019-08-09             11
## 3   AAAP          574      2015-11-11    2018-07-18              0
## 4   AABA         5434      1998-01-02    2019-08-07             14
## 5    AAC         1222      2014-10-02    2019-08-09             21
##   earnings_from_date earnings_to_date
## 1         2009-05-14       2019-08-14
## 2         2017-01-24       2019-07-17
## 3               NULL             NULL
## 4         2014-01-28       2017-04-18
## 5         2014-11-05       2019-04-16
```

```r
#Divid ends of Stock Prices.
path <- 'dividends_latest.csv'
dataSetReader_Dividends <-read.csv(path, nrows = nRowsRead)
#Dividends of Stock Prices.
str(dataSetReader_Dividends)
```

```
## 'data.frame':    1000 obs. of  3 variables:
##  $ symbol  : Factor w/ 33 levels "AAL","AAME","AAON",..: 33 33 33 33 33 33 33 33 33 33 ...
##  $ date    : Factor w/ 704 levels "2000-01-10","2000-02-28",..: 633 376 264 368 401 385 178 516 542 1
##  $ dividend: num  0.39 0.16 0.11 0.16 0.2 0.16 0.09 0.31 0.31 0.13 ...
```

```r
#Head of Divid ends of Stock.
head(dataSetReader_Dividends, 5)
```

```
##   symbol       date dividend
## 1   MSFT 2016-11-15     0.39
## 2   MSFT 2011-05-17     0.16
## 3   MSFT 2008-05-13     0.11
## 4   MSFT 2011-02-15     0.16
## 5   MSFT 2012-02-14     0.20
```

```r
#Earnings of Stock Prices.
path <- 'earnings_latest.csv'
dataSetReader_Earnings <-read.csv(path, nrows = nRowsRead)
#Dividends of Stock Prices.
str(dataSetReader_Earnings)
```

```
## 'data.frame':    1000 obs. of  6 variables:
##  $ symbol      : Factor w/ 32 levels "A","AA","AABA",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ date        : Factor w/ 667 levels "2009-05-05","2009-05-06",..: 5 17 31 40 59 72 84 93 111 128 .
##  $ qtr         : Factor w/ 88 levels "01/2010","01/2011",..: 22 44 65 1 23 45 66 2 24 46 ...
##  $ eps_est     : Factor w/ 253 levels "-0.0200","-0.0300",..: 253 253 253 253 253 253 253 253 253 25
##  $ eps         : Factor w/ 261 levels "-0.0100","-0.0200",..: 261 261 261 261 261 261 261 261 261 26
##  $ release_time: Factor w/ 3 levels "NULL","post",..: 2 2 3 3 2 2 3 2 1 2 ...
```

```r
#Head of Earnings of Stock.
head(dataSetReader_Earnings, 5)
```

```
##   symbol       date     qtr eps_est  eps release_time
## 1      A 2009-05-14 04/2009    NULL NULL         post
## 2      A 2009-08-17 07/2009    NULL NULL         post
## 3      A 2009-11-13 10/2009    NULL NULL          pre
## 4      A 2010-02-12 01/2010    NULL NULL          pre
## 5      A 2010-05-17 04/2010    NULL NULL         post
```

```r
#Stock Prices.
path <- 'stock_prices_latest_Simplified.csv'
dataSetReader_Prices <-read.csv(path, nrows = nRowsRead)
#Stock Prices.
str(dataSetReader_Prices)
```

```
## 'data.frame':    1000 obs. of  9 variables:
##  $ symbol           : Factor w/ 924 levels "AABA","AAME",..: 855 789 630 174 544 84 577 871 311 873
##  $ date             : Factor w/ 912 levels "1/10/2012","1/11/2005",..: 697 425 651 91 153 35 154 298
##  $ open             : num  0.51 6.37 9.6 19 27.12 ...
##  $ high             : num  0.51 6.37 9.95 19 27.41 ...
##  $ low              : num  0.51 6.37 9.52 18.73 26.99 ...
##  $ close            : num  0.51 6.37 9.9 18.74 27.35 ...
##  $ close_adjusted   : num  0.17 5.16 11404.8 8.89 27.15 ...
##  $ volume           : num  0 0 147735 5400 1028741 ...
##  $ split_coefficient: int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
#Head of Stock Prices.
head(dataSetReader_Prices, 5)
```

```
##   symbol       date  open   high   low close close_adjusted  volume
## 1   TXMD  7/10/2009  0.51  0.510  0.51  0.51         0.1700       0
## 2    SPA   3/3/1999  6.37  6.370  6.37  6.37         5.1619       0
## 3   NURO  6/25/2007  9.60  9.950  9.52  9.90     11404.8000  147735
## 4    CEA 10/14/2004 19.00 19.000 18.73 18.74         8.8906    5400
## 5    MDU 10/31/2017 27.12 27.405 26.99 27.35        27.1524 1028741
##   split_coefficient
## 1                 1
## 2                 1
## 3                 1
## 4                 1
## 5                 1
```

Data preprocessing

stock dataset summary

```r
#find NA across all
missing = dataSetReader_Summary[, sapply(dataSetReader_Summary, anyNA), drop = FALSE]

cat("Missing data found in ",ncol(missing),"Columns, which is",
  ncol(missing)/ncol(dataSetReader_Summary)*100, "% of features")
```

```
## Missing data found in  0 Columns, which is 0 % of features
```

```
#The missing columns and how many missing value it has
missingData <- sapply(dataSetReader_Summary,function(x) {sum(is.na(x))})
Position(function(x) x > 0, missingData)
```

```
## [1] NA
```

```
MissingNames <- names(dataSetReader_Summary[, sapply(dataSetReader_Summary, anyNA), drop = FALSE])

for (i in MissingNames){
  dataSetReader_Summary[is.na(dataSetReader_Summary[,i]),i] <- median(dataSetReader_Summary[,i],na.rm

#Find number of missing values/check ranges
sum(is.na(dataSetReader_Summary))
```

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

```
# Check Duplicate Data Record
nrow(dataSetReader_Summary)
```

```
## [1] 1000
```

```
nrow(dataSetReader_Summary[!duplicated(dataSetReader_Summary),])
```

```
## [1] 1000
```

stock dataset Divid ends

```
#find NA across all
missing = dataSetReader_Dividends[, sapply(dataSetReader_Dividends, anyNA), drop = FALSE]

cat("Missing data found in ",ncol(missing),"Columns, which is",
  ncol(missing)/ncol(dataSetReader_Dividends)*100, "% of features")
```

```
## Missing data found in  0 Columns, which is 0 % of features
```

```
#The missing columns and how many missing value it has
missingData <- sapply(dataSetReader_Dividends,function(x) {sum(is.na(x))})
Position(function(x) x > 0, missingData)
```

```
## [1] NA
```

```
MissingNames <- names(dataSetReader_Dividends[, sapply(dataSetReader_Dividends, anyNA), drop = FALSE])

for (i in MissingNames){
  dataSetReader_Dividends[is.na(dataSetReader_Dividends[,i]),i] <- median(dataSetReader_Dividends[,i]

#Find number of missing values/check ranges
sum(is.na(dataSetReader_Dividends))
```

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

```
# Check Duplicate Data Record
nrow(dataSetReader_Dividends)
```

```
## [1] 1000
```

```
nrow(dataSetReader_Dividends[!duplicated(dataSetReader_Dividends),])
```

```
## [1] 1000
```

stock dataset Earnings

```r
#find NA across all
missing = dataSetReader_Earnings[, sapply(dataSetReader_Earnings, anyNA), drop = FALSE]

cat("Missing data found in ",ncol(missing),"Columns, which is",
  ncol(missing)/ncol(dataSetReader_Earnings)*100, "% of features")
```

```
## Missing data found in  0 Columns, which is 0 % of features
```

```r
#The missing columns and how many missing value it has
missingData <- sapply(dataSetReader_Earnings,function(x) {sum(is.na(x))})
Position(function(x) x > 0, missingData)
```

```
## [1] NA
```

```r
MissingNames <- names(dataSetReader_Earnings[, sapply(dataSetReader_Earnings, anyNA), drop = FALSE])

for (i in MissingNames){
  dataSetReader_Earnings[is.na(dataSetReader_Earnings[,i]),i] <- median(dataSetReader_Earnings[,i],na

#Find number of missing values/check ranges
sum(is.na(dataSetReader_Earnings))
```

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

```r
# Check Duplicate Data Record
nrow(dataSetReader_Earnings)
```

```
## [1] 1000
```

```r
nrow(dataSetReader_Earnings[!duplicated(dataSetReader_Earnings),])
```

```
## [1] 1000
```

stock dataset Stock Prices.

```r
#find NA across all
missing = dataSetReader_Prices[, sapply(dataSetReader_Prices, anyNA), drop = FALSE]

cat("Missing data found in ",ncol(missing),"Columns, which is",
  ncol(missing)/ncol(dataSetReader_Prices)*100, "% of features")
```

```
## Missing data found in  0 Columns, which is 0 % of features
```

```r
#The missing columns and how many missing value it has
missingData <- sapply(dataSetReader_Prices,function(x) {sum(is.na(x))})
Position(function(x) x > 0, missingData)
```

```
## [1] NA
```

```r
MissingNames <- names(dataSetReader_Prices[, sapply(dataSetReader_Prices, anyNA), drop = FALSE])

for (i in MissingNames){
  dataSetReader_Prices[is.na(dataSetReader_Prices[,i]),i] <- median(dataSetReader_Prices[,i],na.rm = 

#Find number of missing values/check ranges
sum(is.na(dataSetReader_Prices))
```

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

```r
# Check Duplicate Data Record
nrow(dataSetReader_Prices)
```

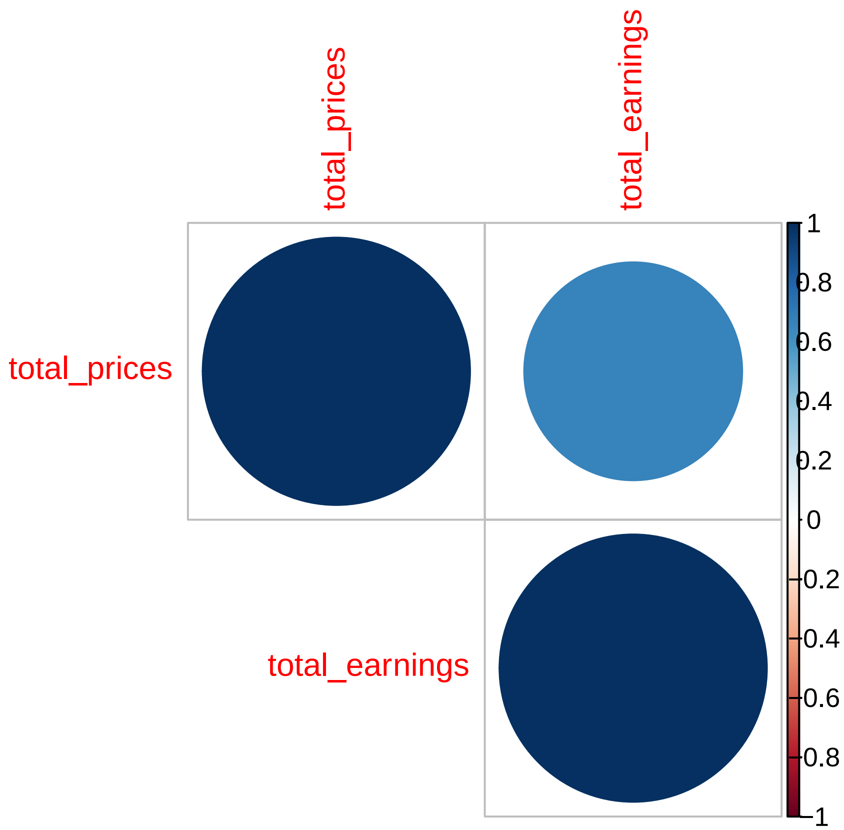
```
## [1] 1000
```

```r
nrow(dataSetReader_Prices[!duplicated(dataSetReader_Prices),])
```

```
## [1] 1000
```

Stock Summary Data Exploration

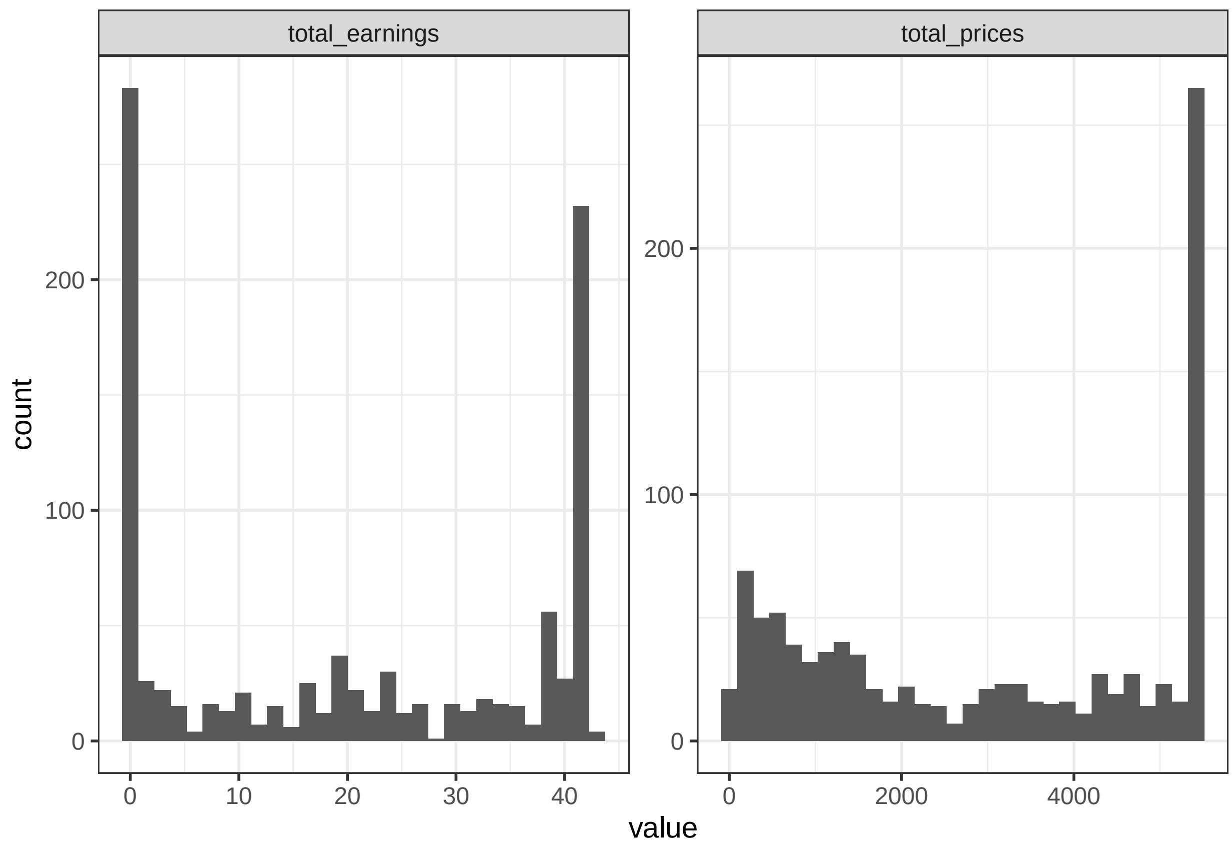
```r
# Removing outliers
dataSetReader_Summary$total_earnings[dataSetReader_Summary$total_earnings %in% boxplot.stats(dataSetRea

# Removing outliers
dataSetReader_Summary$total_prices[dataSetReader_Summary$total_prices %in% boxplot.stats(dataSetReader_S


#Correlation between total_prices and total_earnings variables
cor_matrix <- cor(dataSetReader_Summary[complete.cases(dataSetReader_Summary), sapply(dataSetReader_Summ
corrplot(cor_matrix, type = "upper")
```




```r
# a graphical way of representing the relationship between total_prices and total_earnings field.
theme_set(theme_bw())

# ggplot(dataSetReader_Summary, aes(x = total_earnings, y = total_prices, group = 2)) +
```

```r
#   geom_boxplot() +
#   theme(panel.grid.major.x = element_blank())

dataSetReader_Summary %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
d <- dataSetReader_Summary
d$vs <- factor(d$total_earnings)
d$am <- factor(d$total_prices)

d %>% str()
```

```
## 'data.frame':    1000 obs. of  9 variables:
##  $ symbol            : Factor w/ 1000 levels "A","AA","AAAP",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ total_prices      : num  4962 697 574 5434 1222 ...
##  $ stock_from_date   : Factor w/ 593 levels "1998-01-02","1998-01-20",..: 37 458 416 1 365 145 299 1
##  $ stock_to_date     : Factor w/ 71 levels "2018-01-30","2018-02-12",..: 69 69 16 67 69 67 69 67 69 6
##  $ total_earnings    : num  42 11 0 14 21 23 23 39 41 24 ...
##  $ earnings_from_date: Factor w/ 351 levels "2009-04-16","2009-04-23",..: 13 281 351 166 216 166 147
##  $ earnings_to_date  : Factor w/ 149 levels "2015-05-11","2015-07-16",..: 143 123 149 7 83 129 35 39
##  $ vs                : Factor w/ 44 levels "0","1","2","3",..: 43 12 1 15 22 24 24 40 42 25 ...
```

```
##  $ am                : Factor w/ 672 levels "16","32","33",..: 619 165 144 670 258 490 332 670 672 3
```

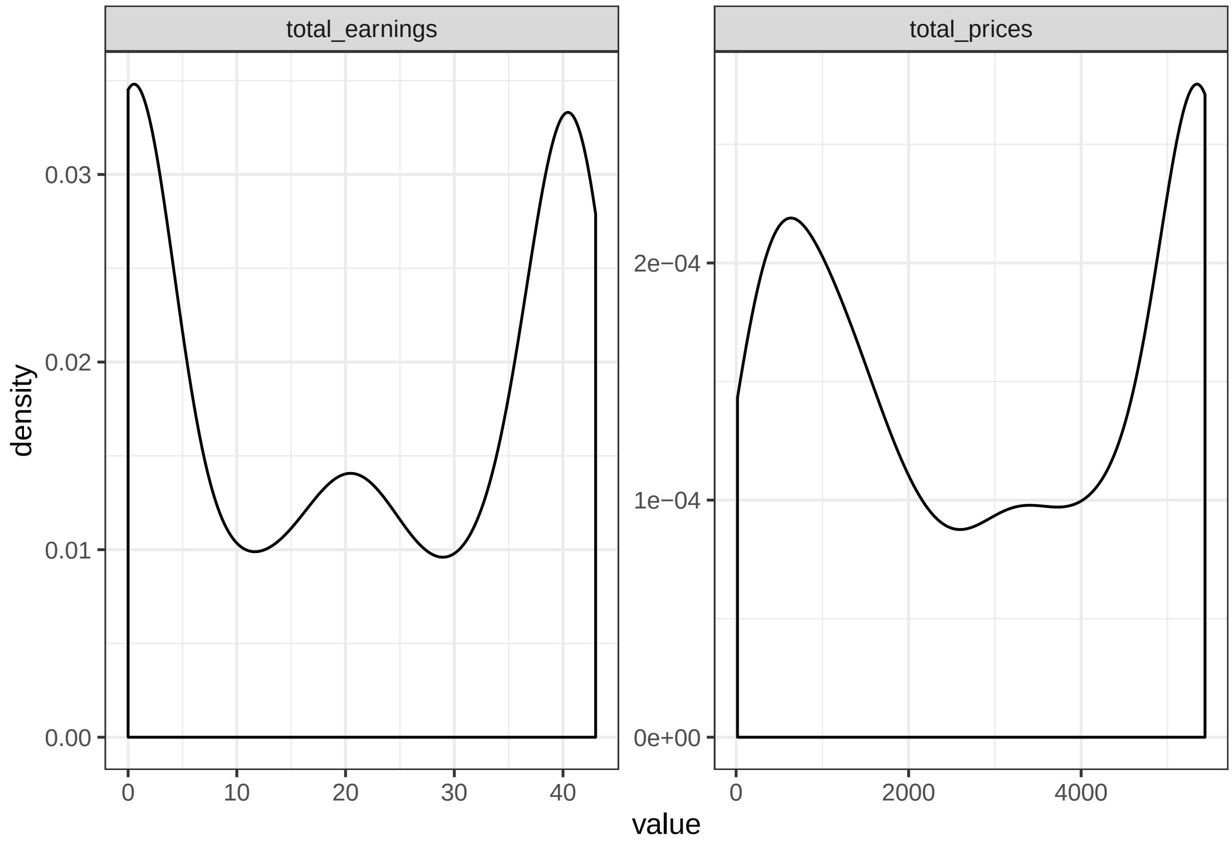
```r
library(purrr)
d %>% keep(is.numeric) %>% head()
```

```
##   total_prices total_earnings
## 1         4962             42
## 2          697             11
## 3          574              0
## 4         5434             14
## 5         1222             21
## 6         3489             23
```

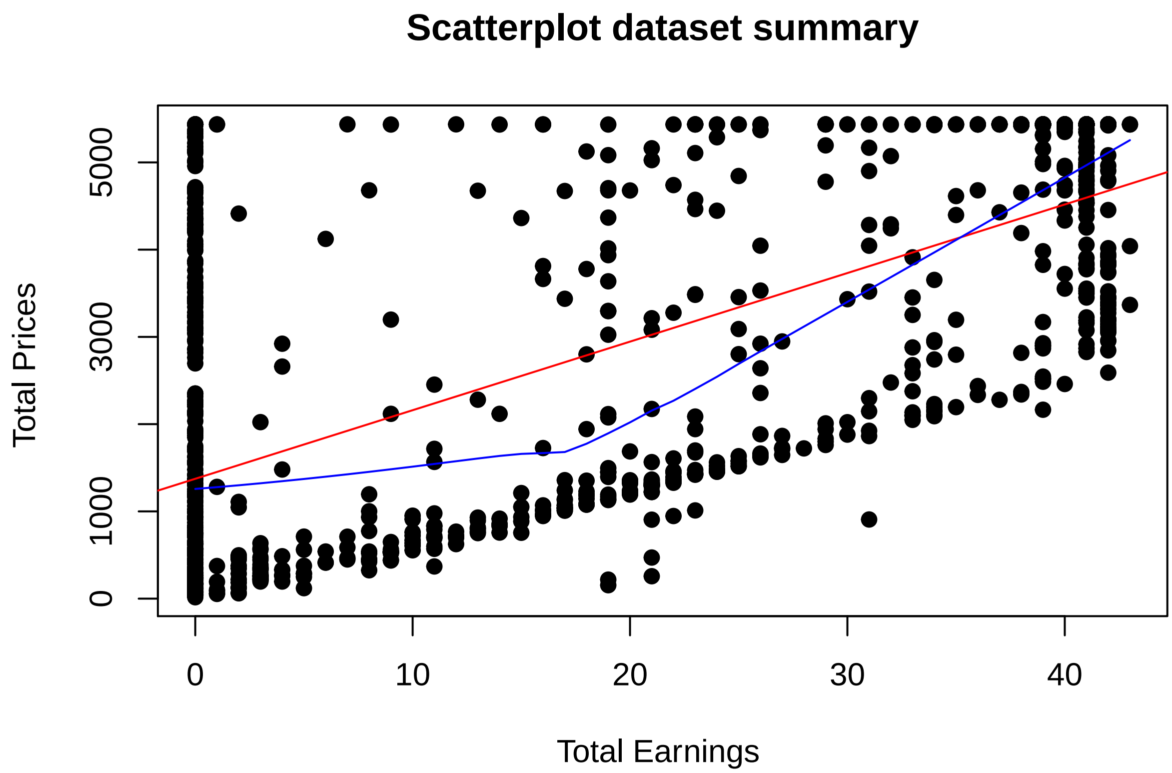
```r
library(tidyr)
d %>%
  keep(is.numeric) %>%
  gather() %>%
  head()
```

```
##            key value
## 1 total_prices  4962
## 2 total_prices   697
## 3 total_prices   574
## 4 total_prices  5434
## 5 total_prices  1222
## 6 total_prices  3489
```

```r
library(ggplot2)
d %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density()                         # as density
```



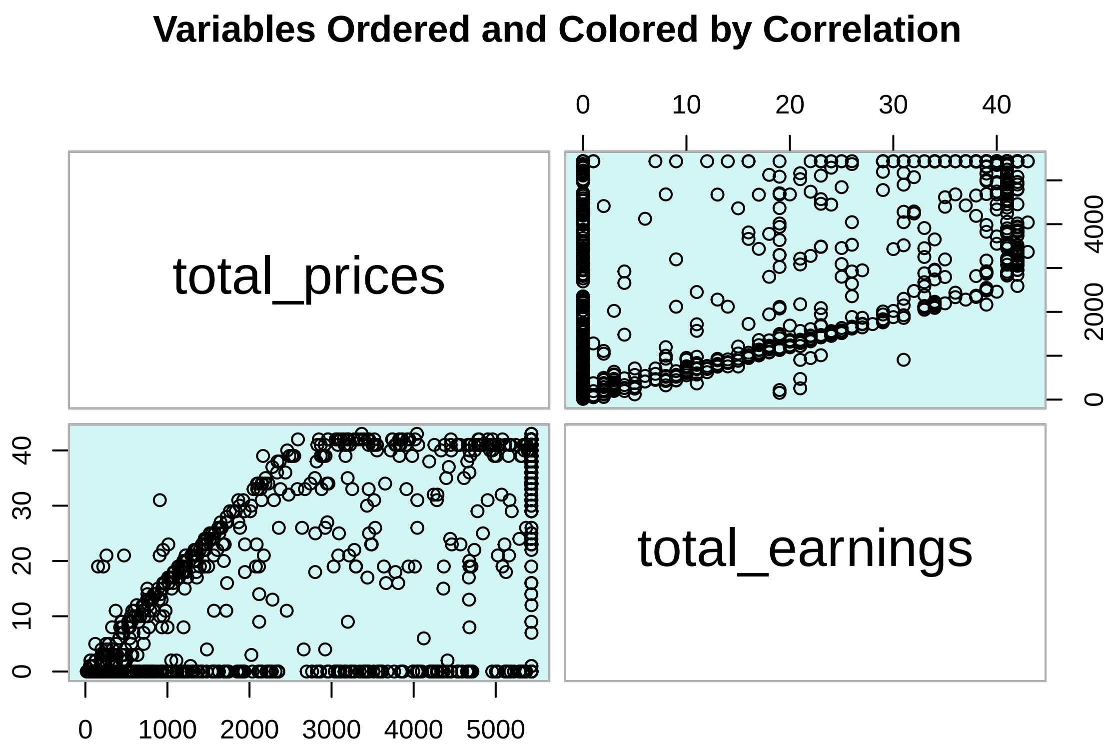

```
plot(dataSetReader_Summary$total_earnings, dataSetReader_Summary$total_prices, main="Scatterplot dataset
   xlab="Total Earnings ", ylab="Total Prices ", pch=19)


# Add fit lines
abline(lm(dataSetReader_Summary$total_prices~dataSetReader_Summary$total_earnings), col="red") # regres
lines(lowess(dataSetReader_Summary$total_earnings,dataSetReader_Summary$total_prices), col="blue") # lo
```



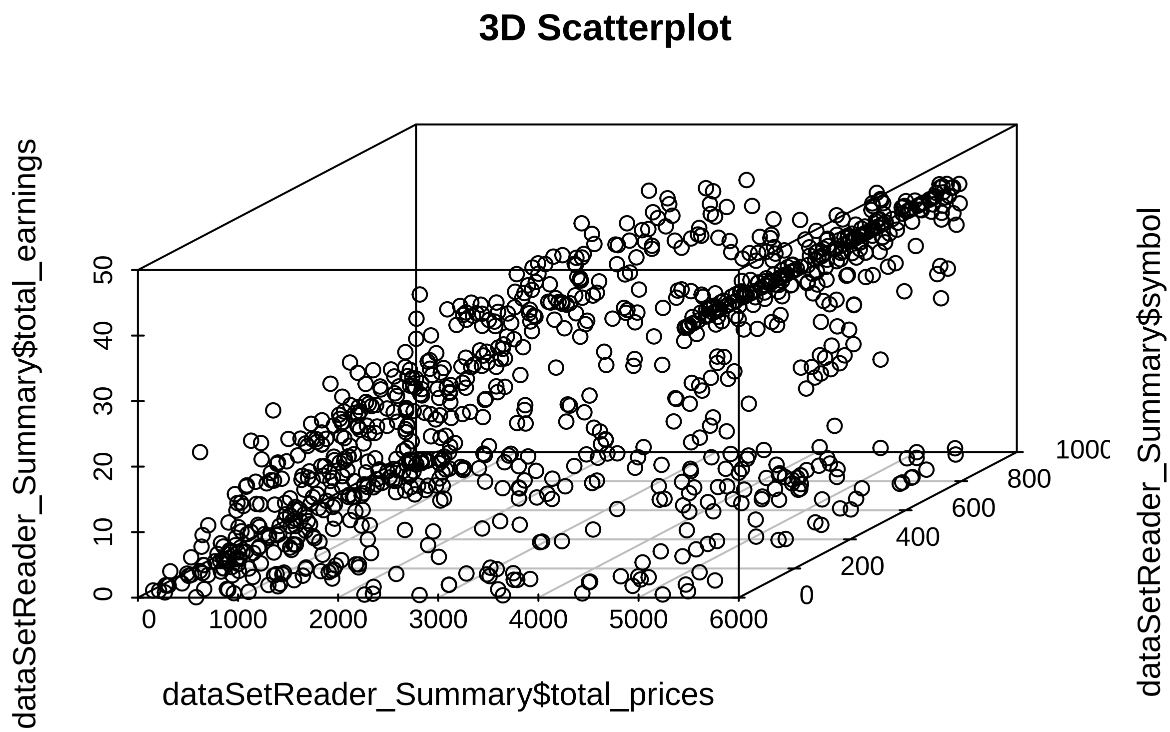

```
# Scatterplot Matrices from the glus Package
library(gclus)
```

```
##
## Attaching package: 'gclus'

## The following object is masked from 'package:dendextend':
##
##     order.hclust
```

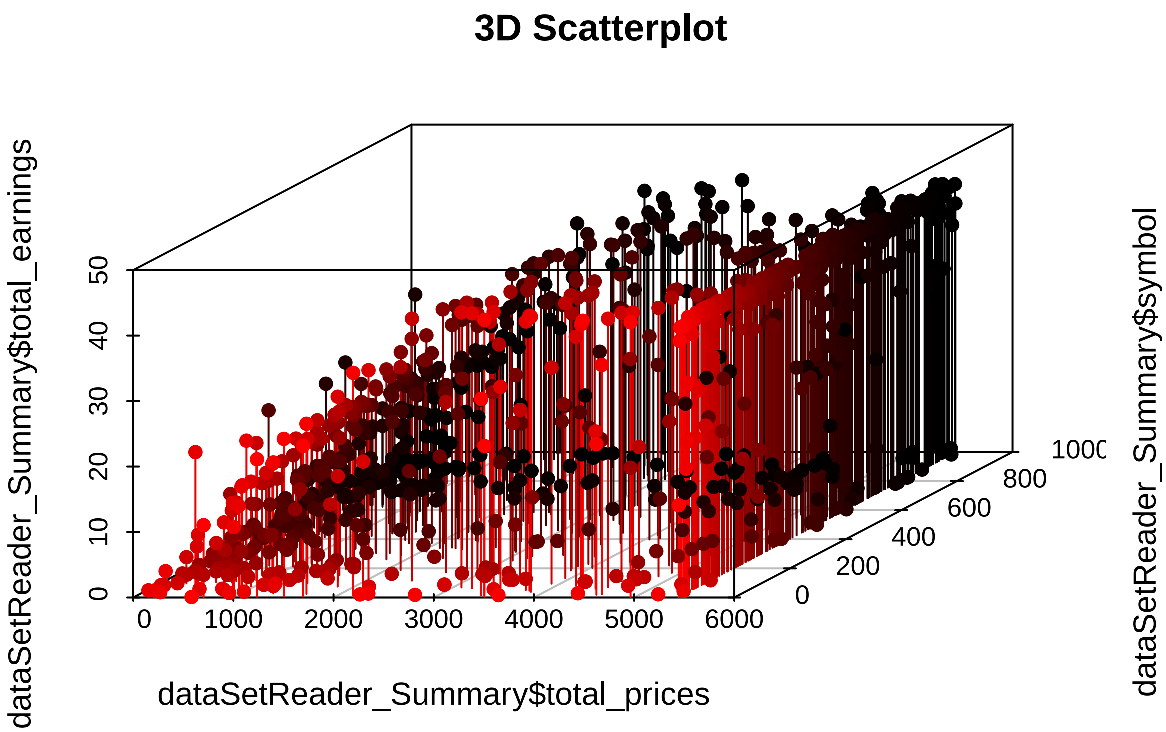
```
dta <- dataSetReader_Summary[c(2,5)] # get data
dta.r <- abs(cor(dta)) # get correlations
dta.col <- dmat.color(dta.r) # get colors
# reorder variables so those with highest correlation
# are closest to the diagonal
dta.o <- order.single(dta.r)
cpairs(dta, dta.o, panel.colors=dta.col, gap=.5,
main="Variables Ordered and Colored by Correlation" )
```




```
# 3D Scatterplot
library(scatterplot3d)
scatterplot3d(dataSetReader_Summary$total_prices,dataSetReader_Summary$symbol,dataSetReader_Summary$tota
```

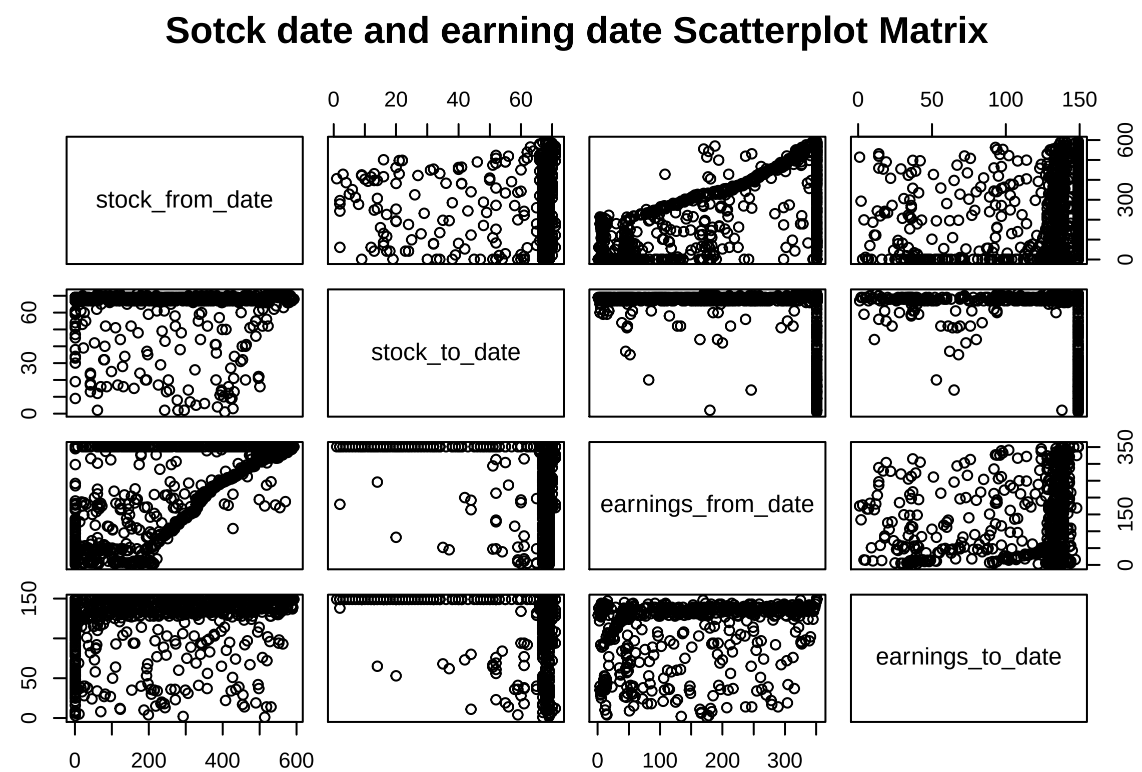



```
# 3D Scatterplot with Coloring and Vertical Drop Lines
library(scatterplot3d)
scatterplot3d(dataSetReader_Summary$total_prices,dataSetReader_Summary$symbol,dataSetReader_Summary$tota
  type="h", main="3D Scatterplot")
```



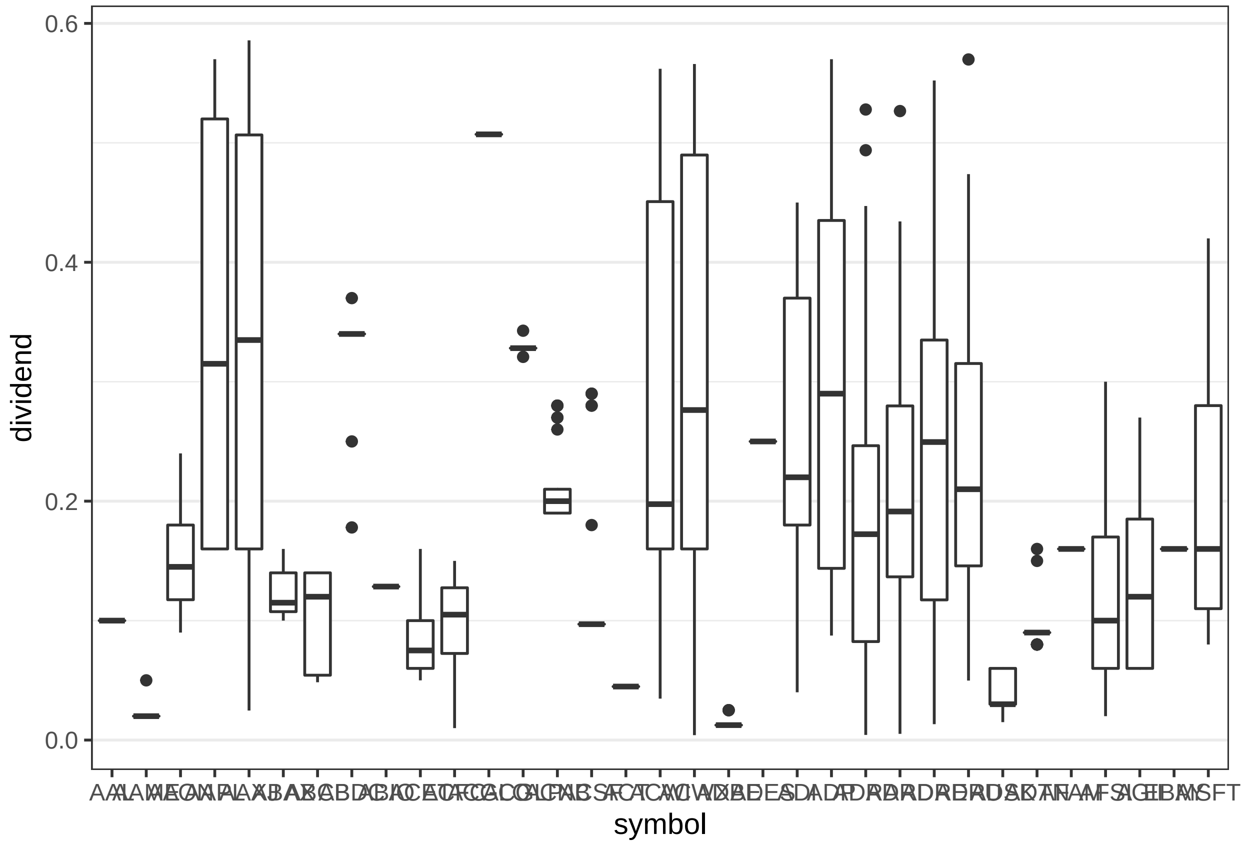

```
# Spinning 3d Scatterplot
library(rgl)

plot3d(dataSetReader_Summary$total_prices,dataSetReader_Summary$symbol,dataSetReader_Summary$total_earn
```

```
# Basic Scatterplot Matrix
pairs(~stock_from_date+stock_to_date+earnings_from_date+earnings_to_date,data=dataSetReader_Summary,
   main="Sotck date and earning date Scatterplot Matrix")
```



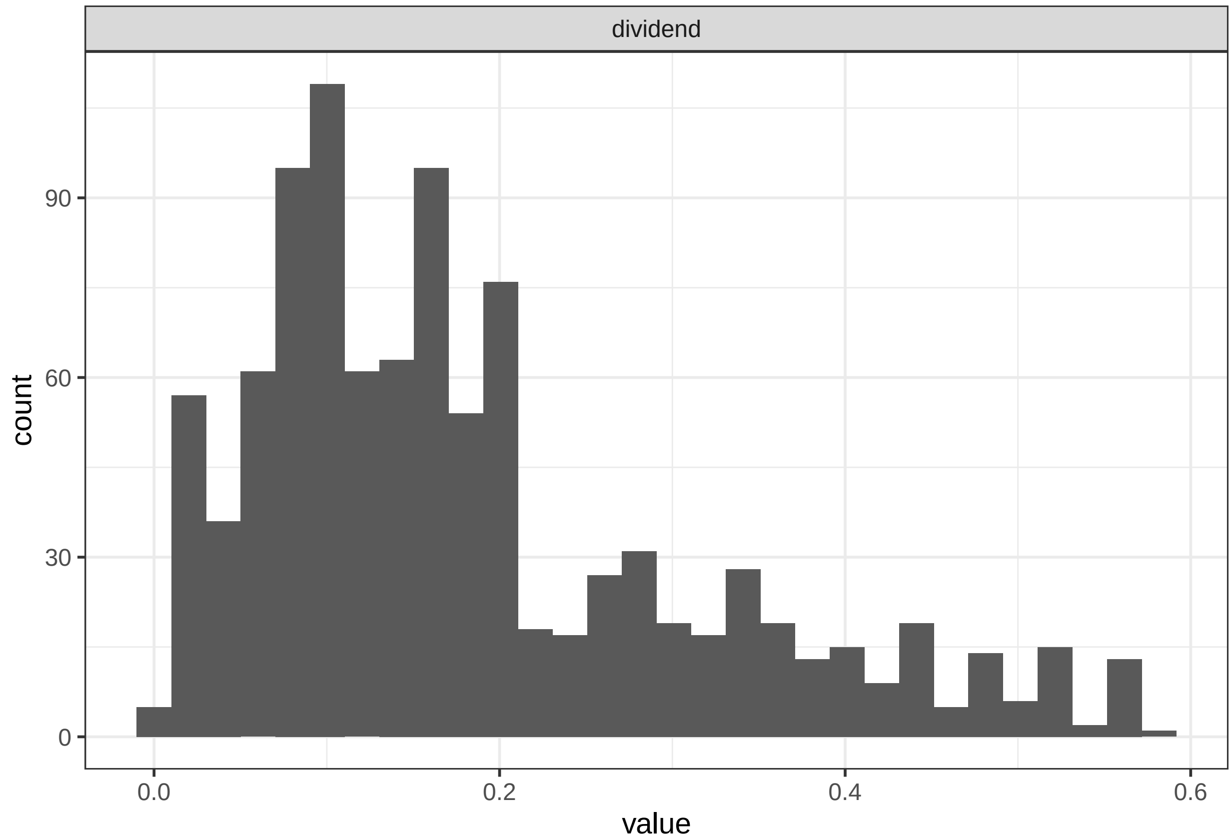
Stock Dividends Data Exploration

```
# Removing outliers
dataSetReader_Dividends$dividend[dataSetReader_Dividends$dividend %in% boxplot.stats(dataSetReader_Divic

ggplot(dataSetReader_Dividends, aes(x = symbol, y = dividend)) +
  geom_boxplot() +
  theme(panel.grid.major.x = element_blank())
```




```
dataSetReader_Dividends %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
d <- dataSetReader_Dividends
d$vs <- factor(d$symbol)
d$am <- factor(d$dividend)

d %>% str()
```

```
## 'data.frame':    1000 obs. of  5 variables:
##  $ symbol  : Factor w/ 33 levels "AAL","AAME","AAON",..: 33 33 33 33 33 33 33 33 33 33 ...
##  $ date    : Factor w/ 704 levels "2000-01-10","2000-02-28",..: 633 376 264 368 401 385 178 516 542
##  $ dividend: num  0.39 0.16 0.11 0.16 0.2 0.16 0.09 0.31 0.31 0.13 ...
##  $ vs      : Factor w/ 33 levels "AAL","AAME","AAON",..: 33 33 33 33 33 33 33 33 33 33 ...
##  $ am      : Factor w/ 331 levels "0.0041","0.0043",..: 269 121 73 121 158 121 62 233 233 92 ...
```

```r
library(purrr)
d %>% keep(is.numeric) %>% head()
```

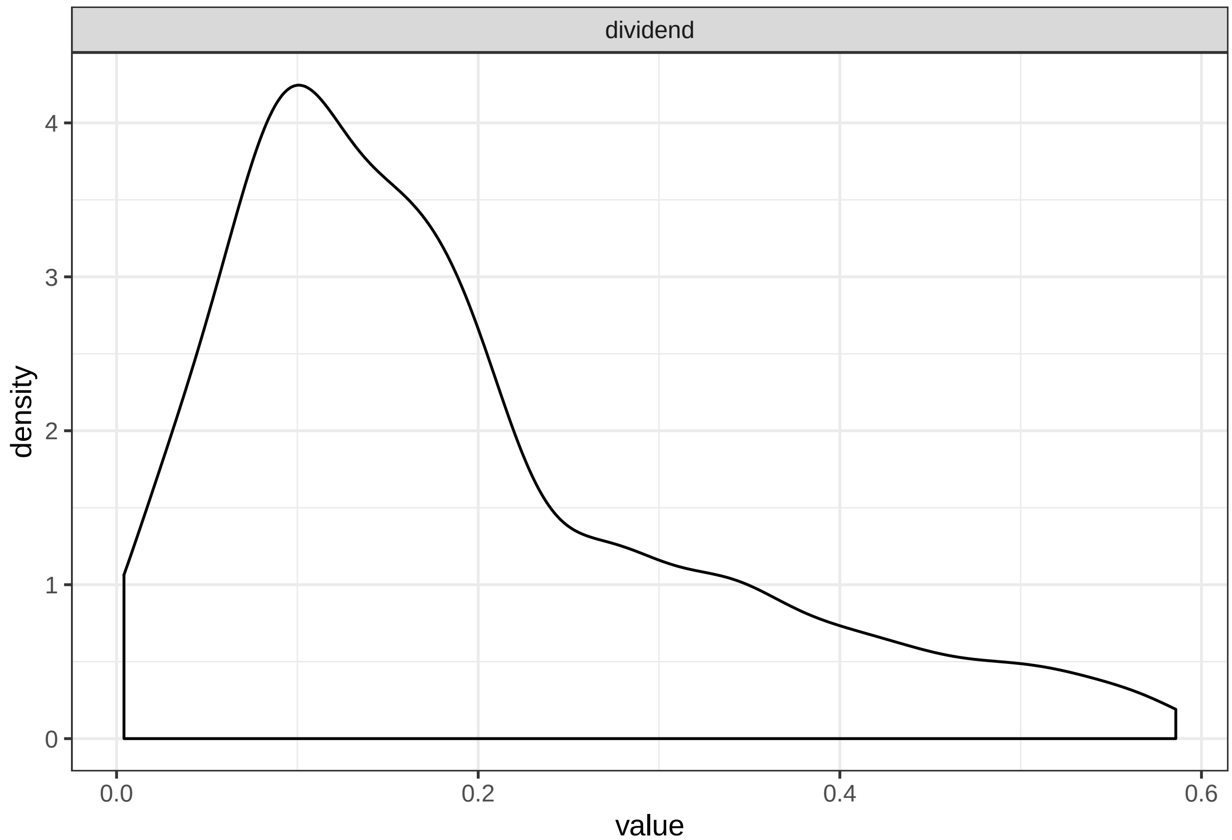
```
##   dividend
## 1     0.39
## 2     0.16
## 3     0.11
## 4     0.16
## 5     0.20
## 6     0.16
```

```r
library(tidyr)
d %>%
  keep(is.numeric) %>%
  gather() %>%
```

```
  head()
```

```
##        key value
## 1 dividend  0.39
## 2 dividend  0.16
## 3 dividend  0.11
## 4 dividend  0.16
## 5 dividend  0.20
## 6 dividend  0.16
```

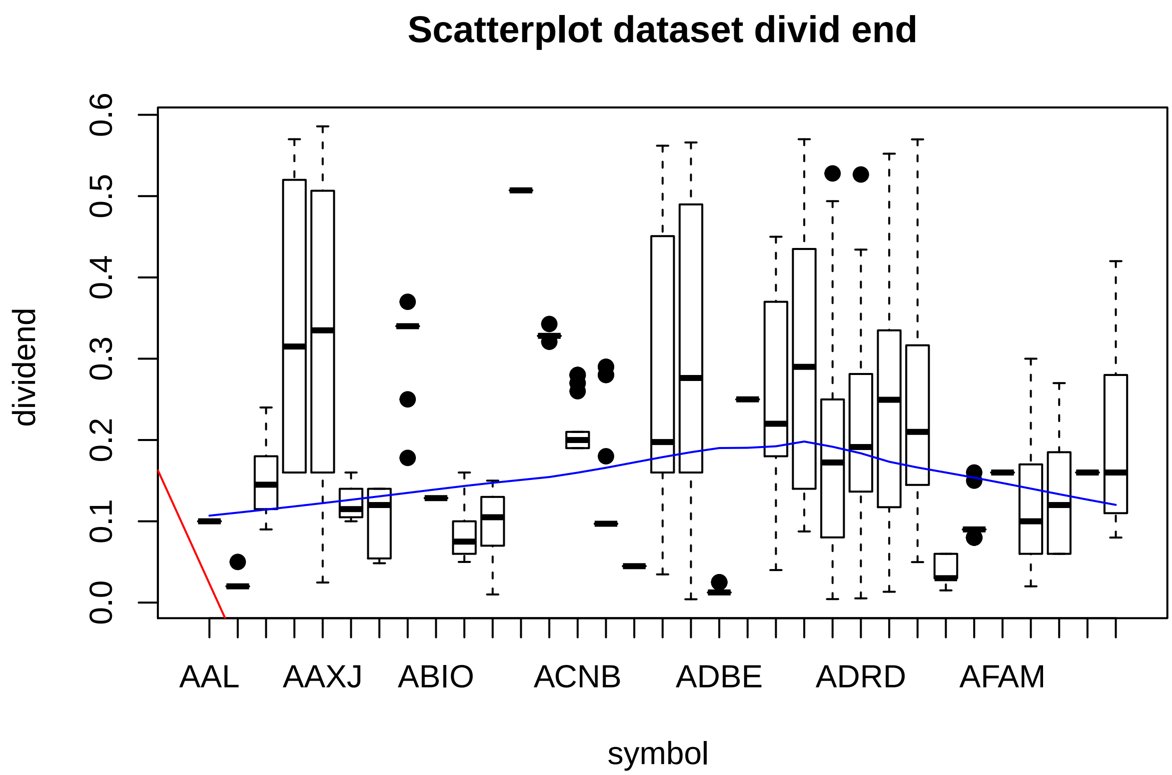
```
library(ggplot2)
d %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density()                         # as density
```



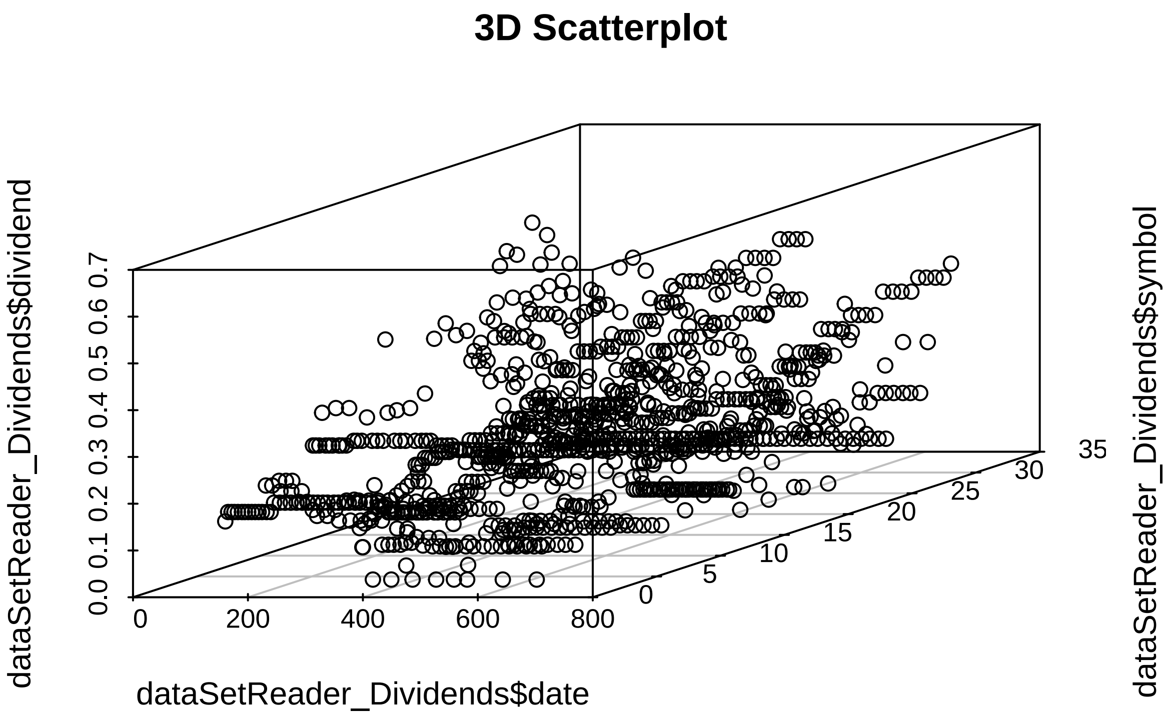

```
plot(dataSetReader_Dividends$symbol, dataSetReader_Dividends$dividend, main="Scatterplot dataset divid
   xlab="symbol ", ylab="dividend ", pch=19)

# Add fit lines
abline(lm(dataSetReader_Dividends$dividend~dataSetReader_Dividends$symbol), col="red") # regression lin
```

```
## Warning in abline(lm(dataSetReader_Dividends$dividend ~
## dataSetReader_Dividends$symbol), : only using the first two of 33
## regression coefficients
```

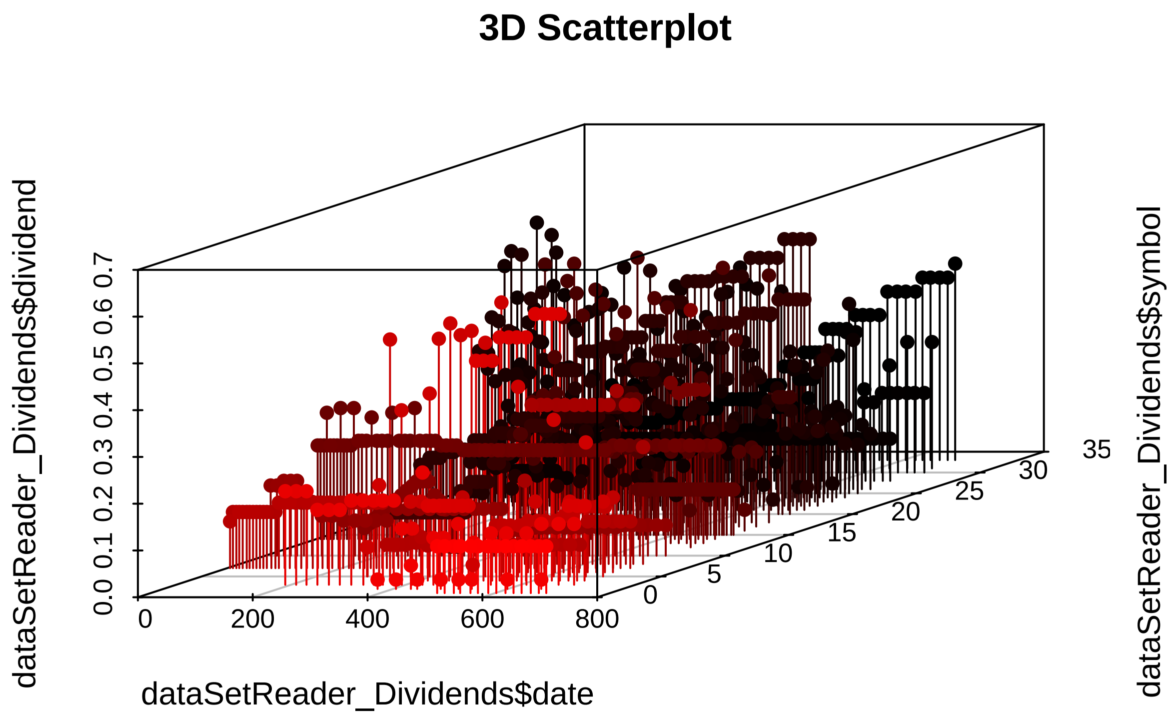
```
lines(lowess(dataSetReader_Dividends$symbol,dataSetReader_Dividends$dividend), col="blue") # lowess lin
```




```
# 3D Scatterplot
library(scatterplot3d)
scatterplot3d(dataSetReader_Dividends$date,dataSetReader_Dividends$symbol,dataSetReader_Dividends$divid
```




```r
# 3D Scatterplot with Coloring and Vertical Drop Lines
library(scatterplot3d)
scatterplot3d(dataSetReader_Dividends$date,dataSetReader_Dividends$symbol,dataSetReader_Dividends$divide
  type="h", main="3D Scatterplot")
```

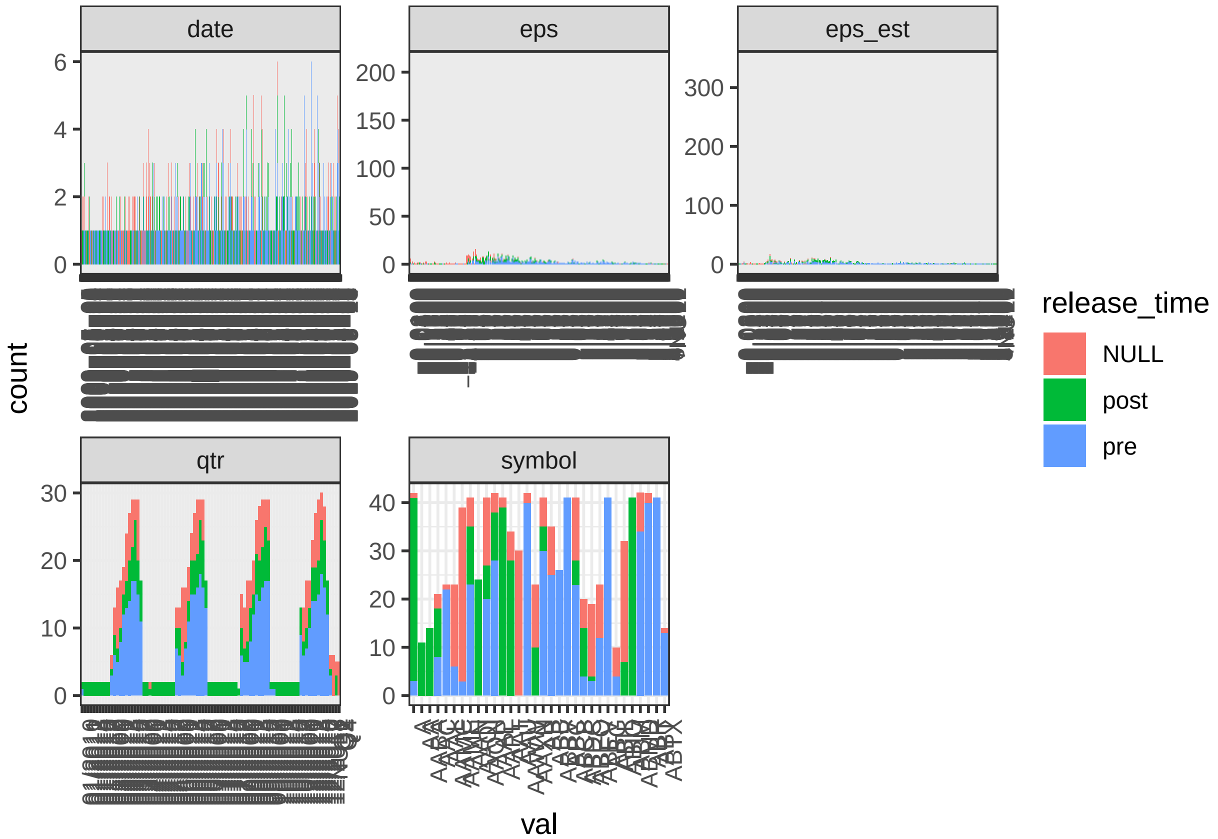



```r
# Spinning 3d Scatterplot
library(rgl)

plot3d(dataSetReader_Dividends$date,dataSetReader_Dividends$symbol,dataSetReader_Dividends$dividend, col
```

Stock Earnings Data Exploration

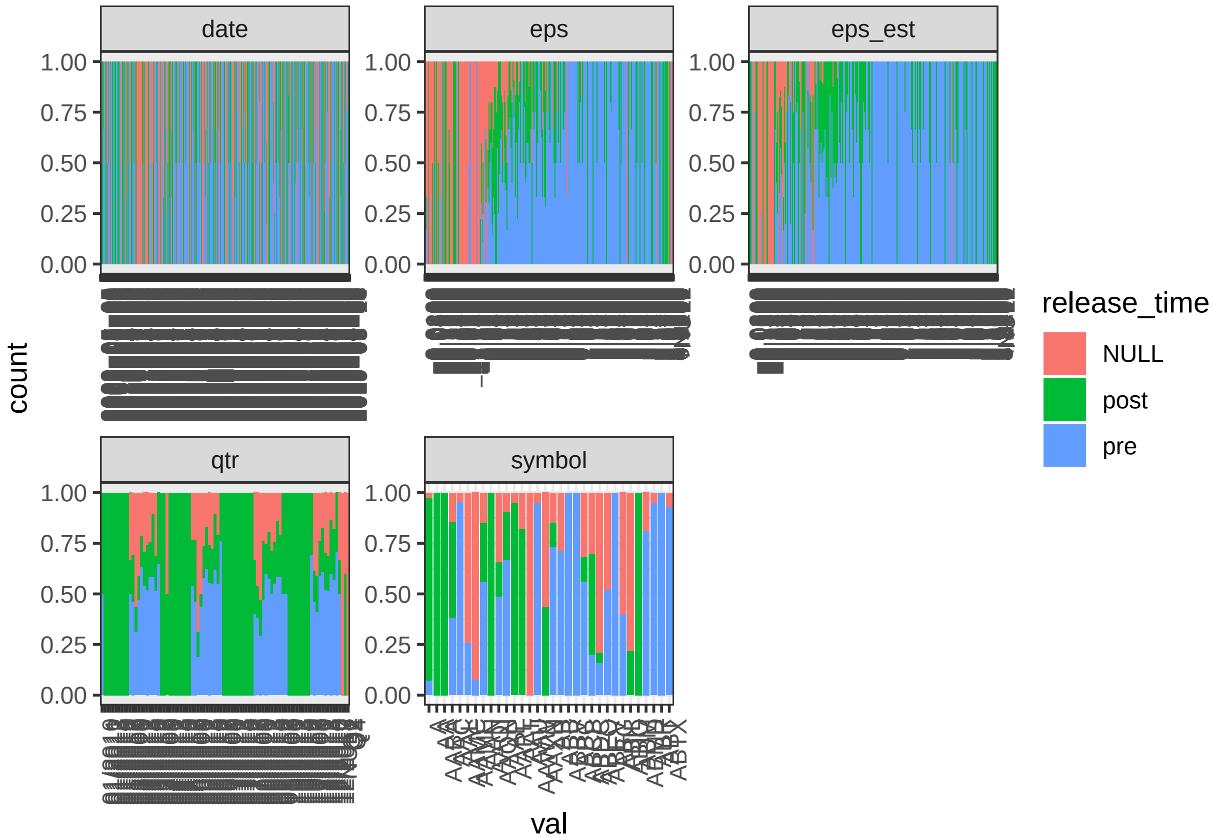
```r
dataSetReader_EarningsFactor <- dataSetReader_Earnings %>% select_if(is.factor)

# Exploration of all factor variables
# absolute bar chart
dataSetReader_EarningsFactor %>%gather("key","val",setdiff(names(.), "release_time")) %>%
  ggplot(aes(val,fill=release_time)) +
    facet_wrap(~ key, scales = "free") +
    geom_bar(stat = 'count',position = "stack") + theme(axis.text.x = element_text(angle = 90, hjust = 
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

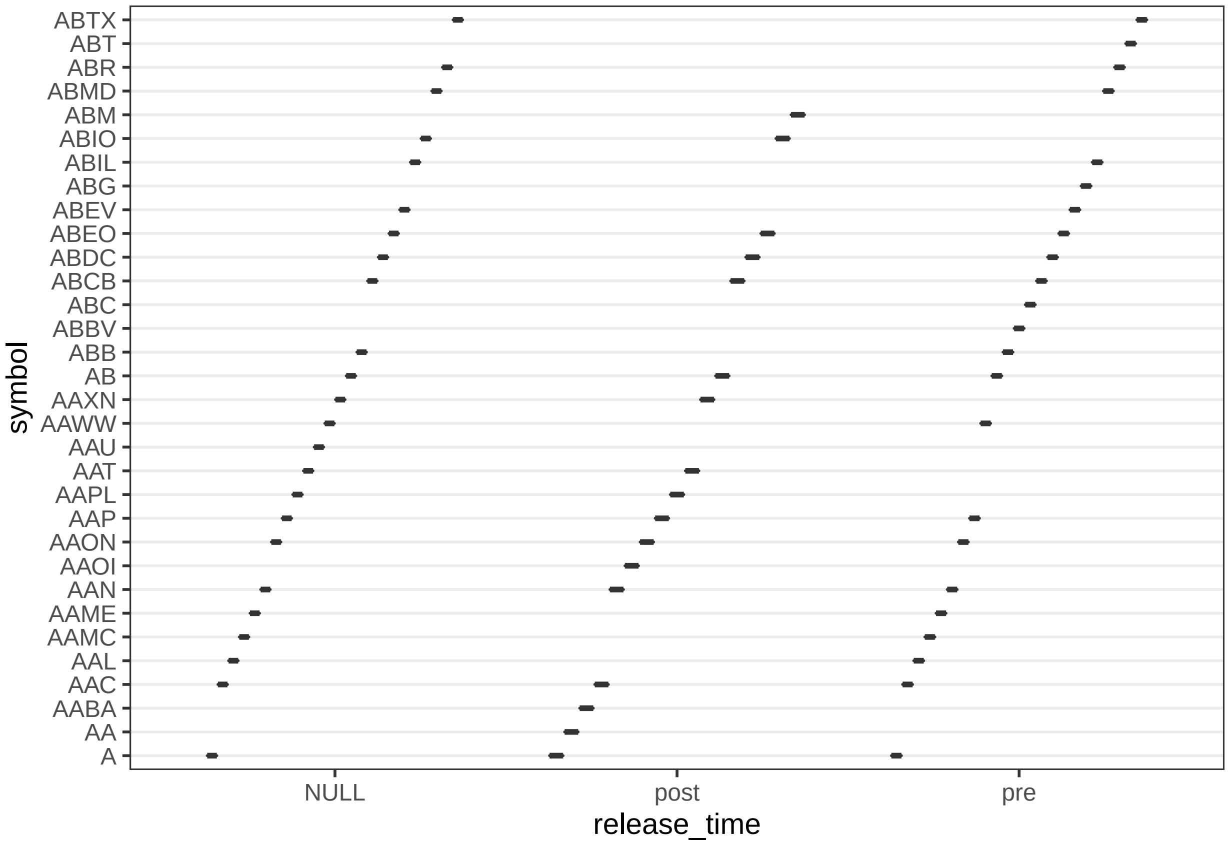



```
# Exploration of all factor variables
# Relative bar chart
dataSetReader_EarningsFactor %>%gather("key","val",setdiff(names(.), "release_time")) %>%
  ggplot(aes(val,fill=release_time)) +
    facet_wrap(~ key, scales = "free") +
    geom_bar(stat = 'count',position = "fill") + theme(axis.text.x = element_text(angle = 90, hjust = 1)
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

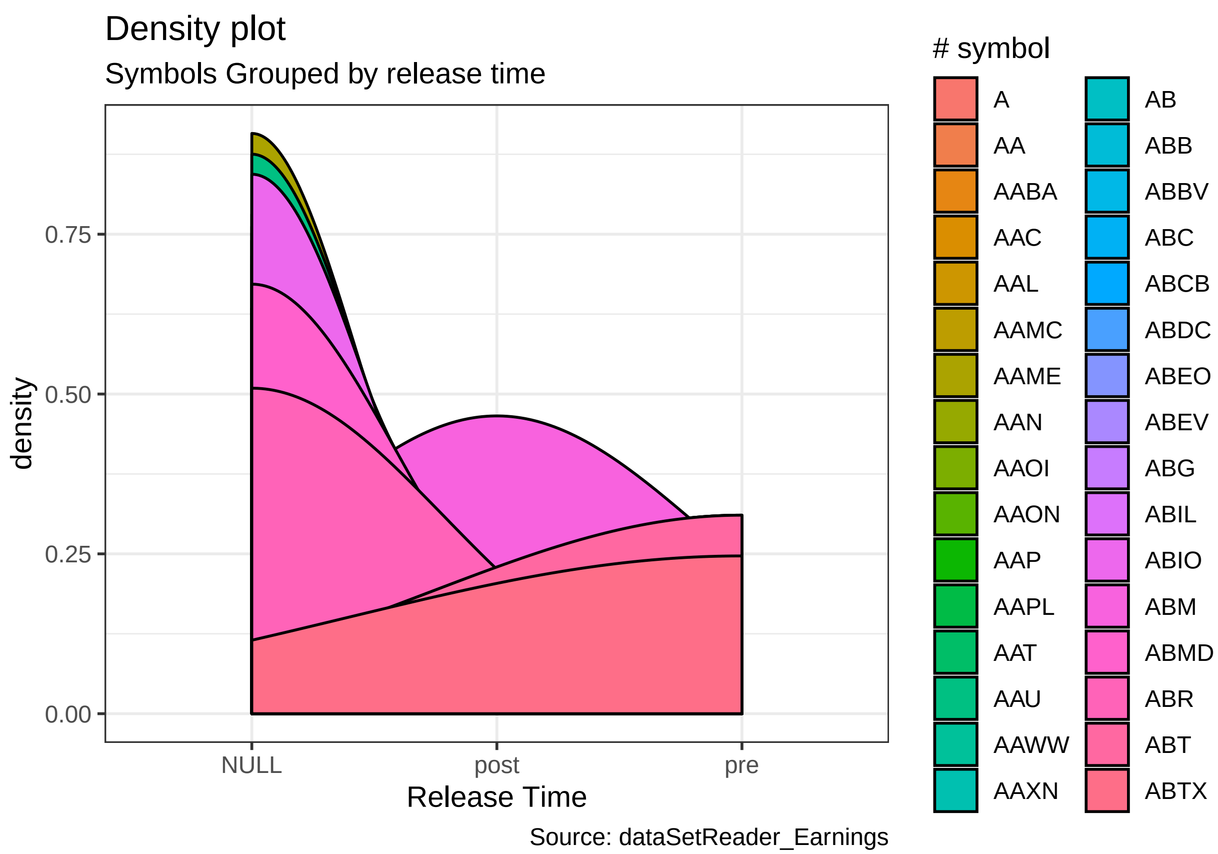



```r
# a graphical way of representing the Min, 1st Qu,  Median, Mean 3rd Qu, and  Max relationship between
theme_set(theme_bw())

ggplot(dataSetReader_Earnings, aes(x = release_time, y = symbol)) +
  geom_boxplot() +
  theme(panel.grid.major.x = element_blank())
```




```r
# The density plot is a basic tool in the data science toolkit.
# density plots are usually a much more effective way to view the distribution of a variable. Create th

ggplot(dataSetReader_Earnings, aes(release_time)) +
  geom_density(aes(fill=factor(symbol))) +
    labs(title="Density plot",
         subtitle="Symbols Grouped by release time",
         caption="Source: dataSetReader_Earnings",
         x="Release Time",
         fill="# symbol")
```

```
## Warning: Groups with fewer than two data points have been dropped.

## Warning: Groups with fewer than two data points have been dropped.

## Warning: Groups with fewer than two data points have been dropped.

## Warning: Groups with fewer than two data points have been dropped.
```




```r
# Categorical variable(release time) vs Categorical variable(symbol)


# compare two categorical variable education field and attrition.
# as we see in the graph the technical people and marketing are the most people that they leave the com
# this is an important attribute for prediction based on the p-value result (p=0.008471793 < 0.05).
xtabs(~symbol+release_time,dataSetReader_Earnings)
```

```
##        release_time
## symbol NULL post pre
##   A       1   38   3
##   AA      0   11   0
##   AABA    0   14   0
##   AAC     3   10   8
##   AAL     1    0  22
##   AAMC   17    0   6
##   AAME   36    0   3
##   AAN     6   12  23
##   AAOI    0   24   0
##   AAON   14    7  20
##   AAP     4   10  28
##   AAPL    2   39   0
##   AAT     6   28   0
##   AAU    30    0   0
##   AAWW    2    0  40
##   AAXN   13   10   0
```

```
##   AB      6    5  30
##   ABB    10    0  25
##   ABBV    0    0  26
##   ABC     0    0  41
##   ABCB   13    5  23
##   ABDC    6   10   4
##   ABEO   15    1   3
##   ABEV   11    0  12
##   ABG     0    0  41
##   ABIL    6    0   4
##   ABIO   25    7   0
##   ABM     0   41   0
##   ABMD    8    0  34
##   ABR     2    0  40
##   ABT     0    0  41
##   ABTX    1    0  13
```

```r
# convert eps and eps_est to numeric

dataSetReader_Earnings$eps <- as.numeric(as.character(dataSetReader_Earnings$eps))
```

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

```r
dataSetReader_Earnings$eps_est <- as.numeric(as.character(dataSetReader_Earnings$eps_est))
```

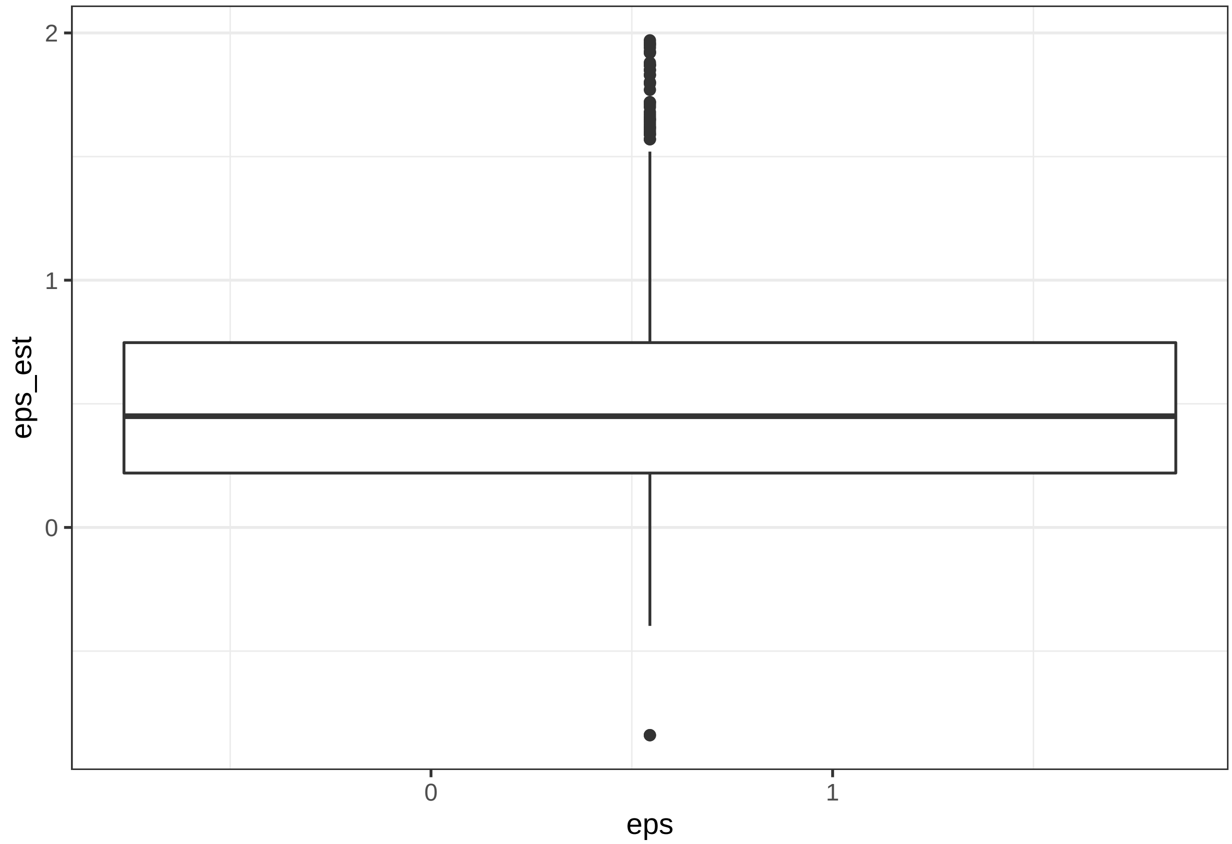
```
## Warning: NAs introduced by coercion
```

```r
# Removing outliers
dataSetReader_Earnings$eps[dataSetReader_Earnings$eps %in% boxplot.stats(dataSetReader_Earnings$eps)$ou

dataSetReader_Earnings$eps_est[dataSetReader_Earnings$eps_est %in% boxplot.stats(dataSetReader_Earnings
```

```r
# a graphical way of representing the relationship between eps and eps_est field.
theme_set(theme_bw())

ggplot(dataSetReader_Earnings, aes(x = eps, y = eps_est, group = 2)) +
  geom_boxplot() +
  theme(panel.grid.major.x = element_blank())
```

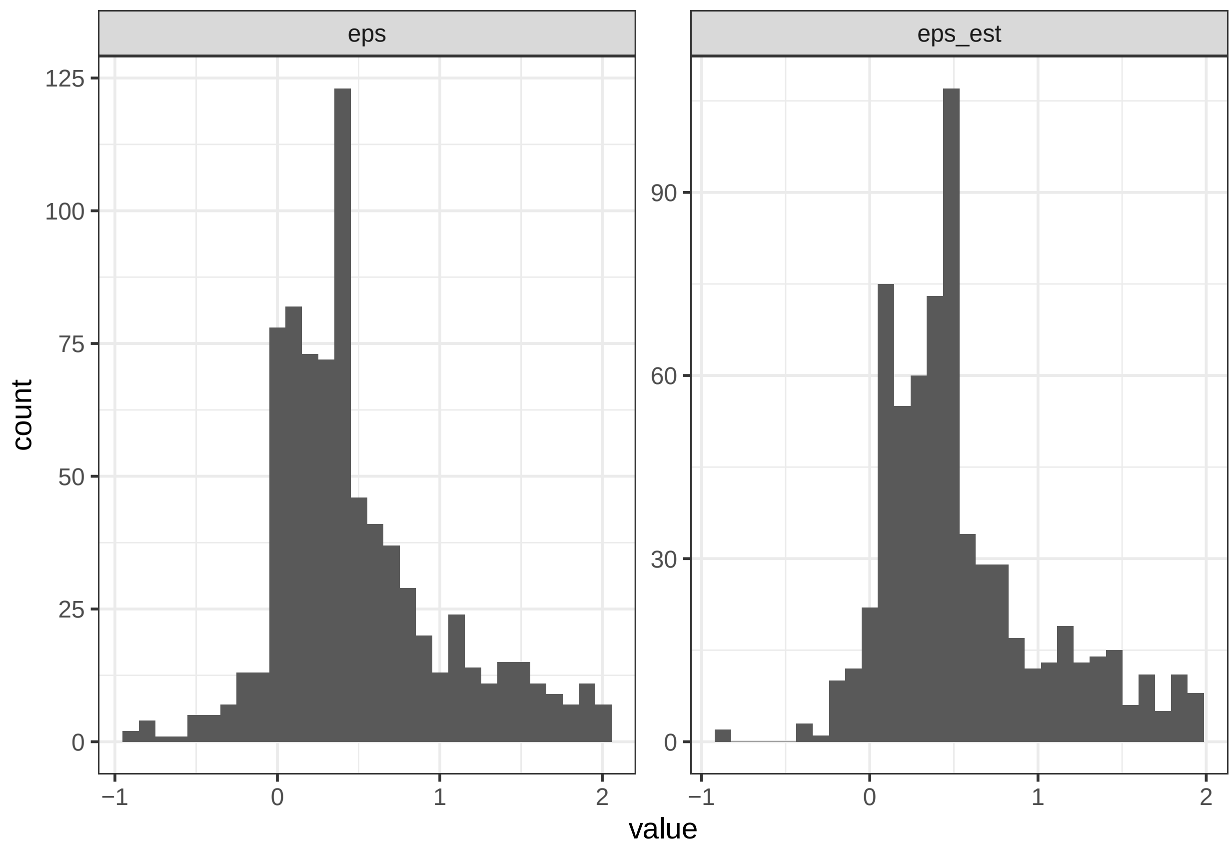
```
## Warning: Removed 211 rows containing missing values (stat_boxplot).
```

```
## Warning: Removed 135 rows containing non-finite values (stat_boxplot).
```




```
dataSetReader_Earnings %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 555 rows containing non-finite values (stat_bin).
```




```
d <- dataSetReader_Earnings
d$vs <- factor(d$eps)
d$am <- factor(d$eps_est)

d %>% str()
```

```
## 'data.frame':    1000 obs. of  8 variables:
##  $ symbol      : Factor w/ 32 levels "A","AA","AABA",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ date        : Factor w/ 667 levels "2009-05-05","2009-05-06",..: 5 17 31 40 59 72 84 93 111 128 .
##  $ qtr         : Factor w/ 88 levels "01/2010","01/2011",..: 22 44 65 1 23 45 66 2 24 46 ...
##  $ eps_est     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ eps         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ release_time: Factor w/ 3 levels "NULL","post",..: 2 2 3 3 2 2 3 2 1 2 ...
##  $ vs          : Factor w/ 209 levels "-0.91","-0.85",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ am          : Factor w/ 215 levels "-0.84","-0.398",..: NA NA NA NA NA NA NA NA NA NA ...
```

```
library(purrr)
d %>% keep(is.numeric) %>% head()
```

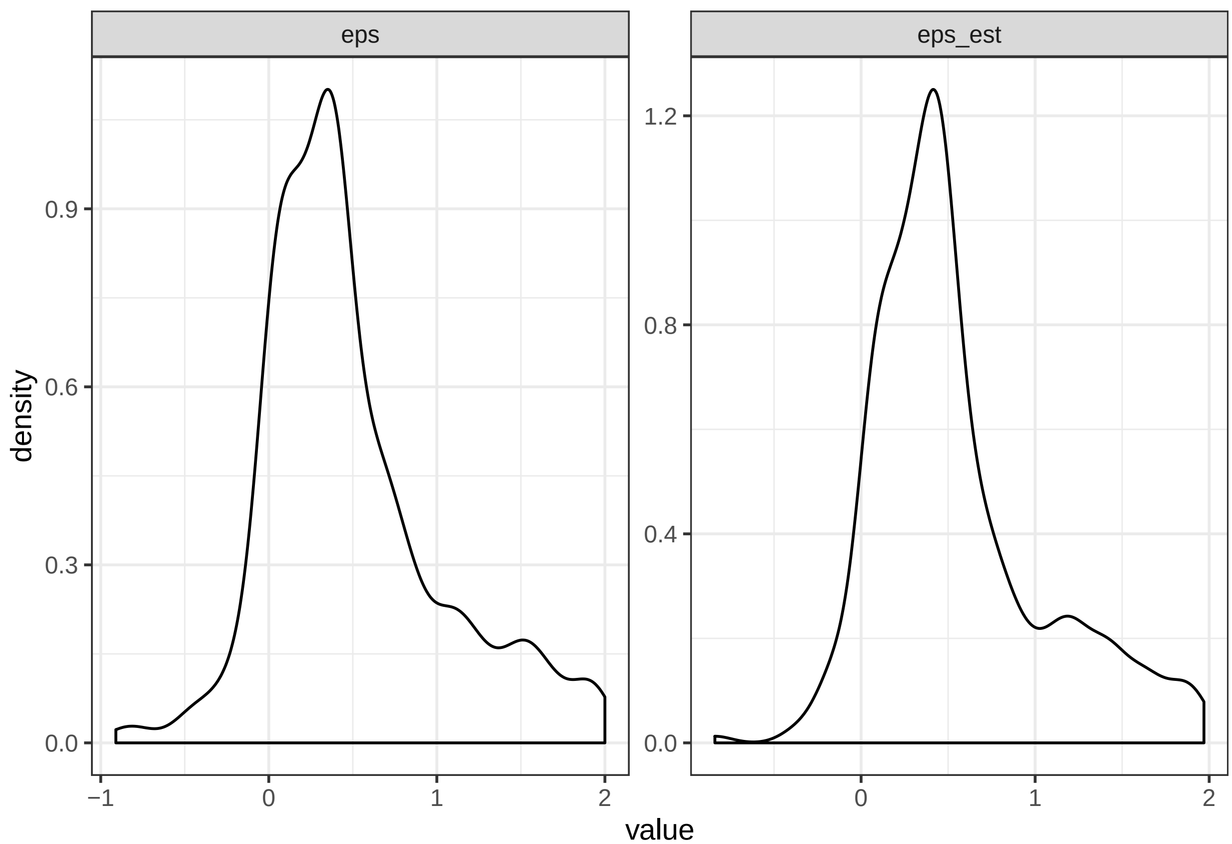
```
##   eps_est eps
## 1      NA  NA
## 2      NA  NA
## 3      NA  NA
## 4      NA  NA
## 5      NA  NA
## 6      NA  NA
```

```r
library(tidyr)
d %>%
  keep(is.numeric) %>%
  gather() %>%
  head()
```

```
##       key value
## 1 eps_est    NA
## 2 eps_est    NA
## 3 eps_est    NA
## 4 eps_est    NA
## 5 eps_est    NA
## 6 eps_est    NA
```

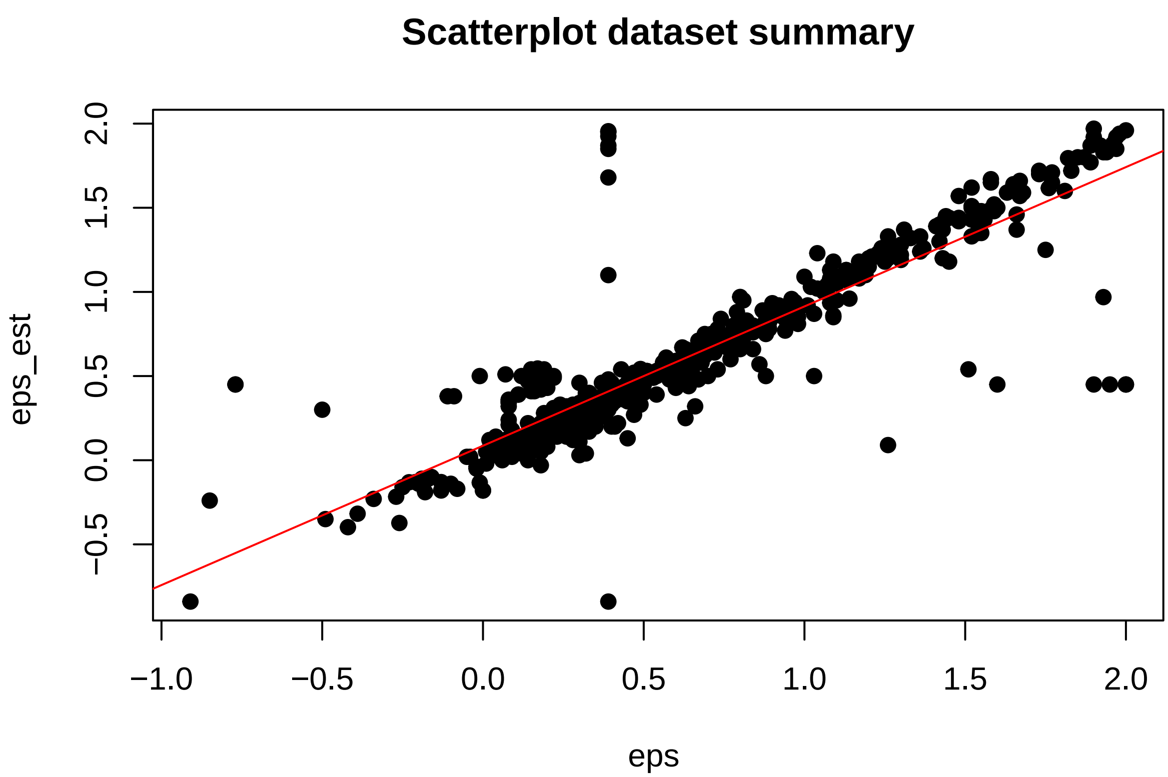
```r
library(ggplot2)
d %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density()                         # as density
```

```
## Warning: Removed 555 rows containing non-finite values (stat_density).
```

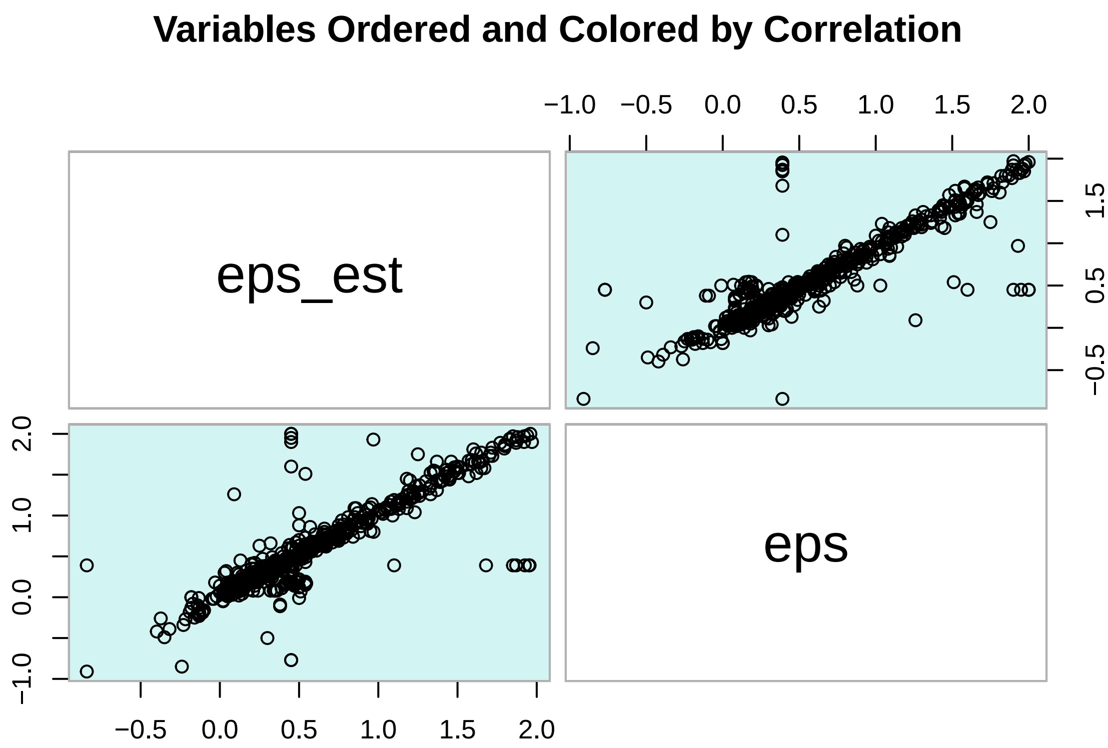



```r
plot(dataSetReader_Earnings$eps, dataSetReader_Earnings$eps_est, main="Scatterplot dataset summary",
   xlab="eps ", ylab="eps_est ", pch=19)
```

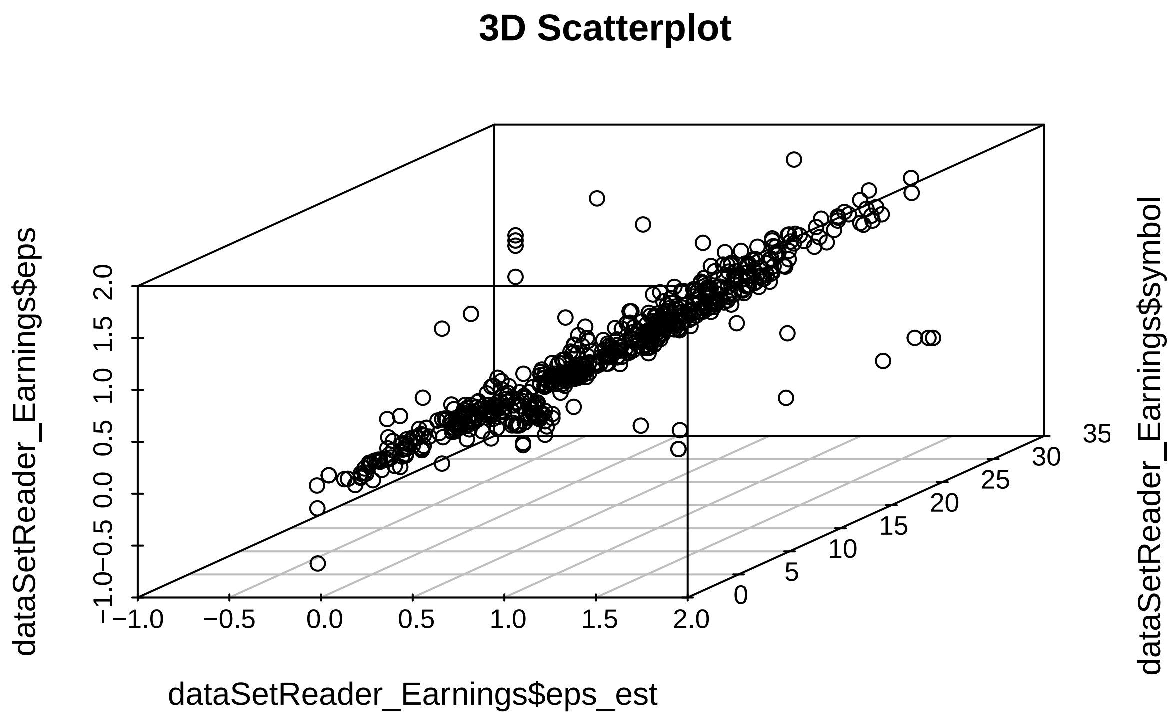
```
# Add fit lines
abline(lm(dataSetReader_Earnings$eps_est~dataSetReader_Earnings$eps), col="red") # regression line (y~x)
```



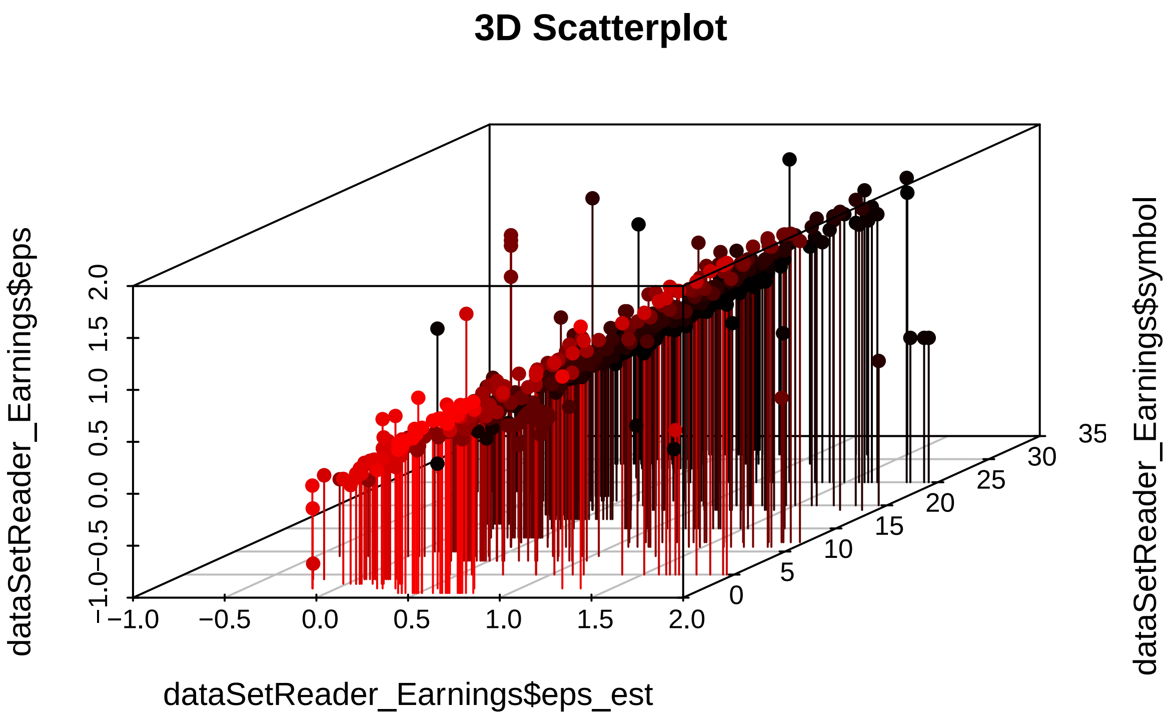

```
# Scatterplot Matrices from the glus Package
library(gclus)
dta <- dataSetReader_Earnings[c(4,5)] # get data
dta.r <- abs(cor(dta)) # get correlations
dta.col <- dmat.color(dta.r) # get colors
# reorder variables so those with highest correlation
# are closest to the diagonal
dta.o <- order.single(dta.r)
cpairs(dta, dta.o, panel.colors=dta.col, gap=.5,
main="Variables Ordered and Colored by Correlation" )
```




```r
# 3D Scatterplot
library(scatterplot3d)
scatterplot3d(dataSetReader_Earnings$eps_est,dataSetReader_Earnings$symbol,dataSetReader_Earnings$eps,
```

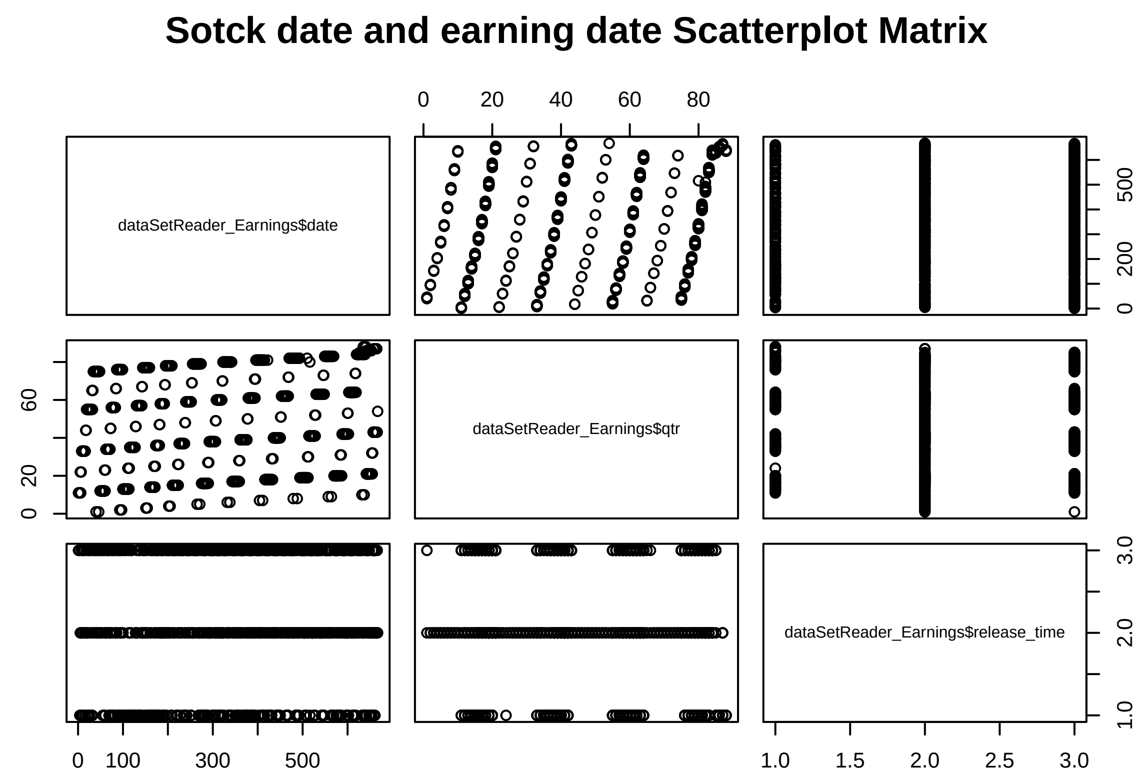



```
# 3D Scatterplot with Coloring and Vertical Drop Lines
library(scatterplot3d)
scatterplot3d(dataSetReader_Earnings$eps_est,dataSetReader_Earnings$symbol,dataSetReader_Earnings$eps, 
  type="h", main="3D Scatterplot")
```




```
# Spinning 3d Scatterplot
library(rgl)

plot3d(dataSetReader_Earnings$eps_est,dataSetReader_Earnings$symbol,dataSetReader_Earnings$eps, col="re
```

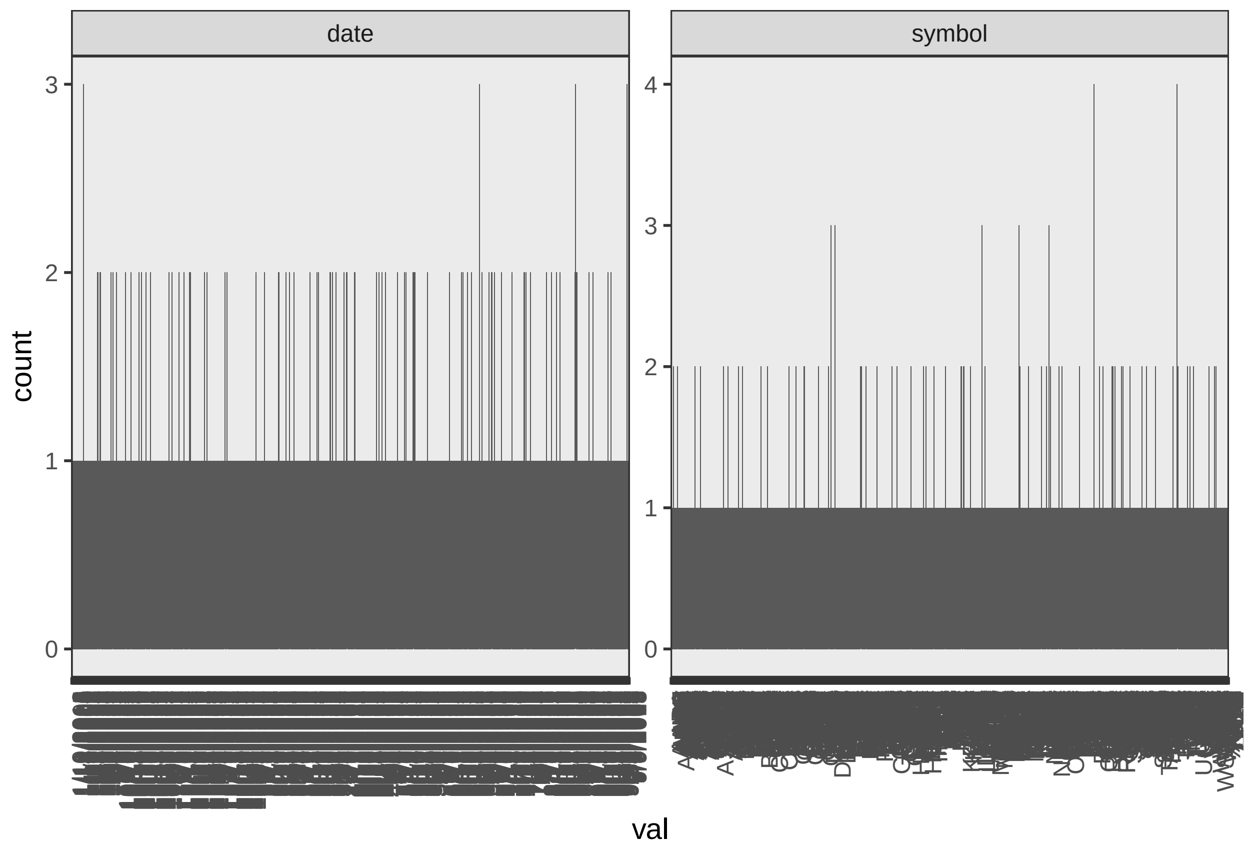
```
# Basic Scatterplot Matrix
pairs(~dataSetReader_Earnings$date+dataSetReader_Earnings$qtr+dataSetReader_Earnings$release_time,data=
   main="Sotck date and earning date Scatterplot Matrix")
```



Stock Prices Data Exploration

```
dataSetReader_PricesFactor <- dataSetReader_Prices %>% select_if(is.factor)

# Exploration of all factor variables
# absolute bar chart
dataSetReader_PricesFactor %>%gather("key","val",setdiff(names(.), "release_time")) %>%
  ggplot(aes(val,fill=dataSetReader_PricesFactor$close_adjusted)) +
    facet_wrap(~ key, scales = "free") +
    geom_bar(stat = 'count',position = "stack") + theme(axis.text.x = element_text(angle = 90, hjust = 
```

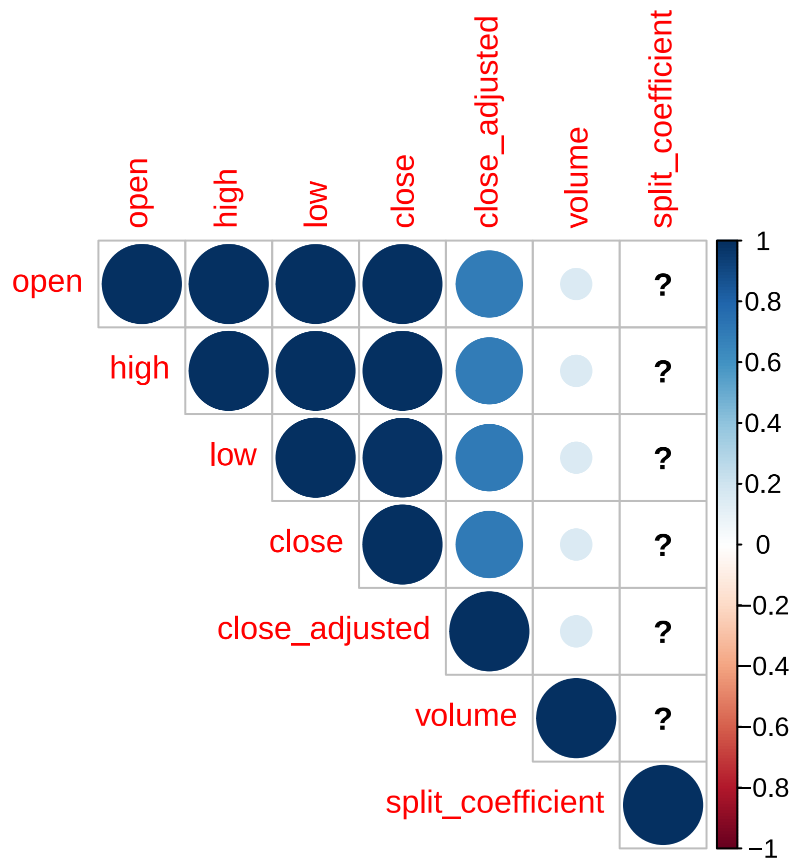
```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```




```r
# Removing outliers
dataSetReader_Prices$high[dataSetReader_Prices$high %in% boxplot.stats(dataSetReader_Prices$high)$out] <
dataSetReader_Prices$low[dataSetReader_Prices$low %in% boxplot.stats(dataSetReader_Prices$low)$out] <- r
dataSetReader_Prices$close[dataSetReader_Prices$close %in% boxplot.stats(dataSetReader_Prices$close)$out
dataSetReader_Prices$open[dataSetReader_Prices$open %in% boxplot.stats(dataSetReader_Prices$open)$out] <
dataSetReader_Prices$close_adjusted[dataSetReader_Prices$close_adjusted %in% boxplot.stats(dataSetReader
dataSetReader_Prices$split_coefficient[dataSetReader_Prices$split_coefficient %in% boxplot.stats(dataSet
dataSetReader_Prices$volume[dataSetReader_Prices$volume %in% boxplot.stats(dataSetReader_Prices$volume)$


#Correlation between total_prices and total_earnings variables
cor_matrix <- cor(dataSetReader_Prices[complete.cases(dataSetReader_Prices), sapply(dataSetReader_Prices
```

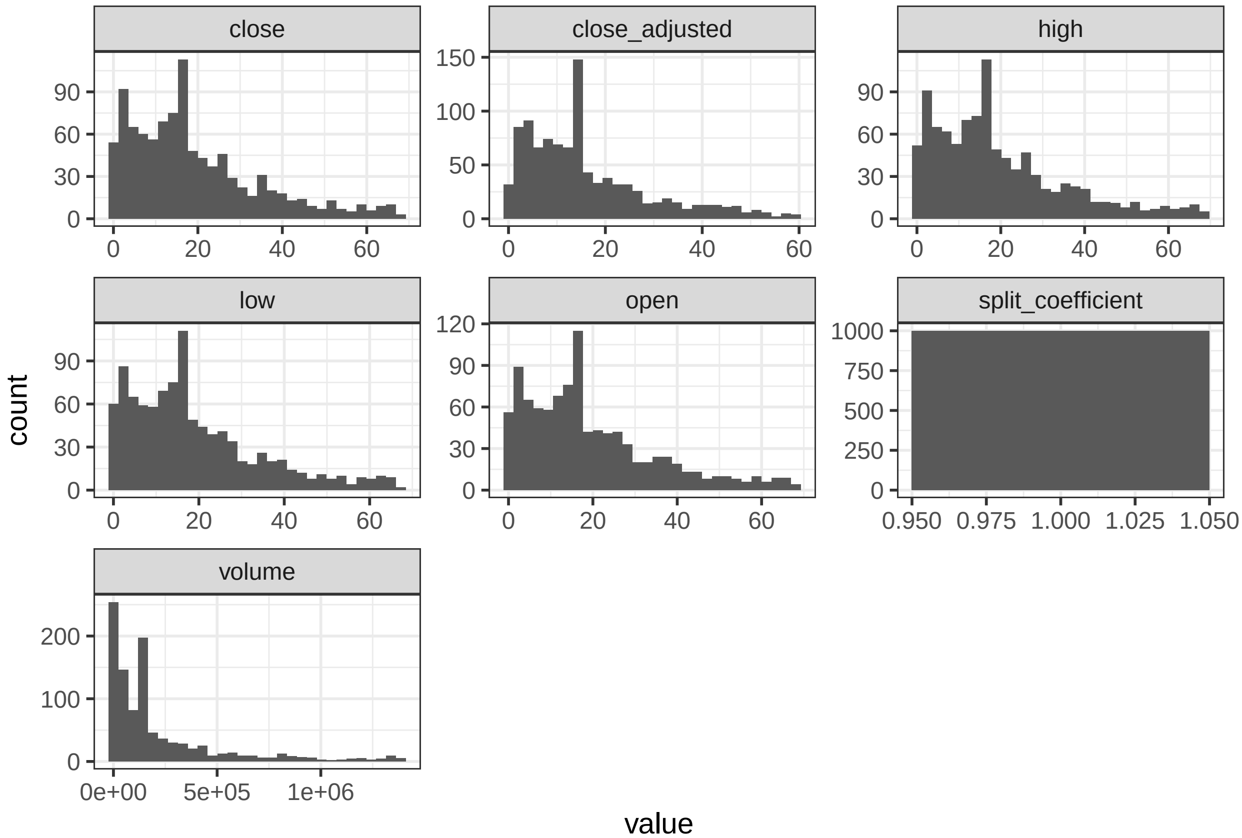
```
## Warning in cor(dataSetReader_Prices[complete.cases(dataSetReader_Prices), :
## the standard deviation is zero
```

```r
corrplot(cor_matrix, type = "upper")
```




```r
# a graphical way of representing the relationship between total_prices and total_earnings field.
theme_set(theme_bw())

dataSetReader_Prices %>%
  keep(is.numeric) %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
d <- dataSetReader_Prices
d$vs <- factor(d$close)
d$am <- factor(d$open)

d %>% str()
```

```
## 'data.frame':    1000 obs. of  11 variables:
##  $ symbol           : Factor w/ 924 levels "AABA","AAME",..: 855 789 630 174 544 84 577 871 311 873
##  $ date             : Factor w/ 912 levels "1/10/2012","1/11/2005",..: 697 425 651 91 153 35 154 298
##  $ open             : num  0.51 6.37 9.6 19 27.12 ...
##  $ high             : num  0.51 6.37 9.95 19 27.41 ...
##  $ low              : num  0.51 6.37 9.52 18.73 26.99 ...
##  $ close            : num  0.51 6.37 9.9 18.74 27.35 ...
##  $ close_adjusted   : num  0.17 5.16 13.91 8.89 27.15 ...
##  $ volume           : num  0 0 147735 5400 1028741 ...
##  $ split_coefficient: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ vs               : Factor w/ 851 levels "0.007","0.011",..: 24 190 270 483 622 245 442 532 55 834
##  $ am               : Factor w/ 838 levels "0.0068","0.011",..: 24 192 266 479 603 246 432 525 58 82
```

```r
library(purrr)
d %>% keep(is.numeric) %>% head()
```

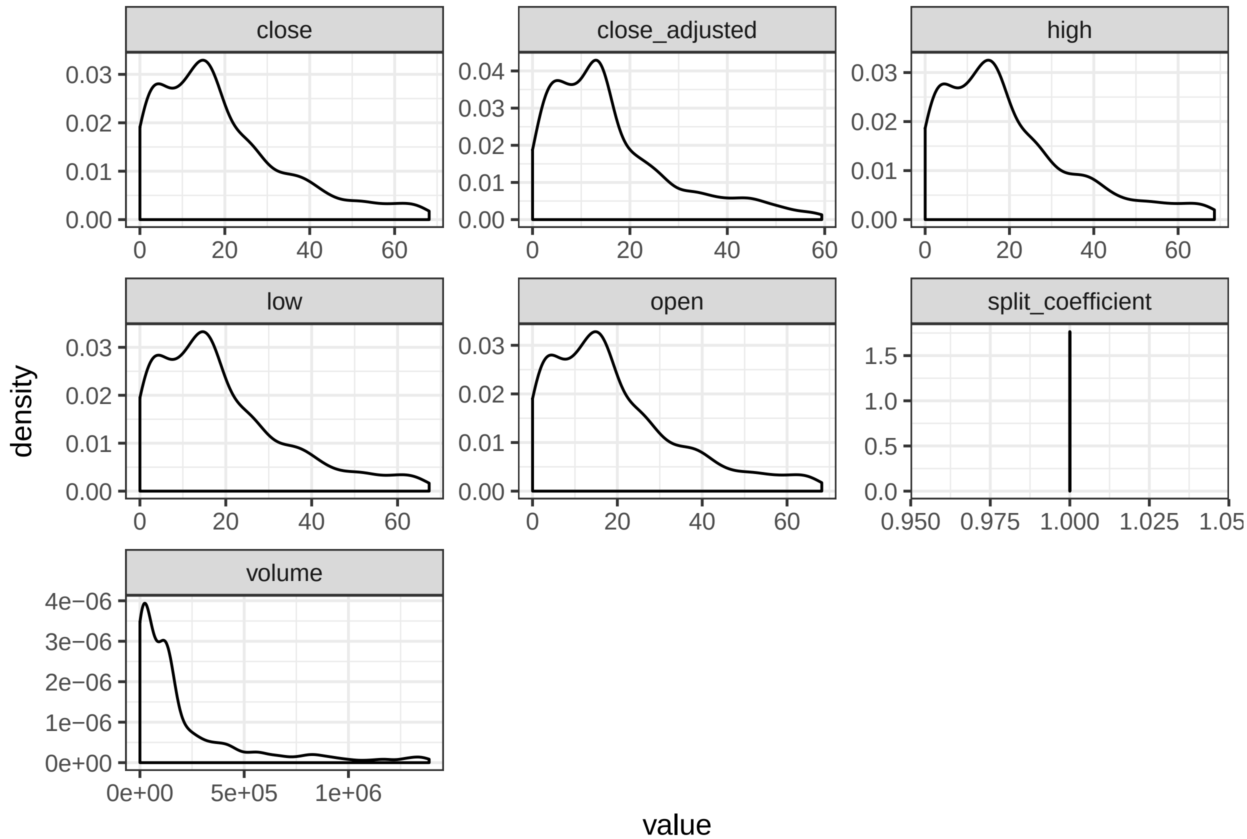
```
##    open   high   low close close_adjusted  volume split_coefficient
## 1  0.51  0.510  0.51  0.51        0.17000       0                 1
## 2  6.37  6.370  6.37  6.37        5.16190       0                 1
## 3  9.60  9.950  9.52  9.90       13.90595  147735                 1
## 4 19.00 19.000 18.73 18.74        8.89060    5400                 1
```

```
## 5 27.12 27.405 26.99 27.35        27.15240 1028741                 1
## 6  8.80  8.800  8.65  8.80         8.23970    1672                 1
```

```
library(tidyr)
d %>%
  keep(is.numeric) %>%
  gather() %>%
  head()
```

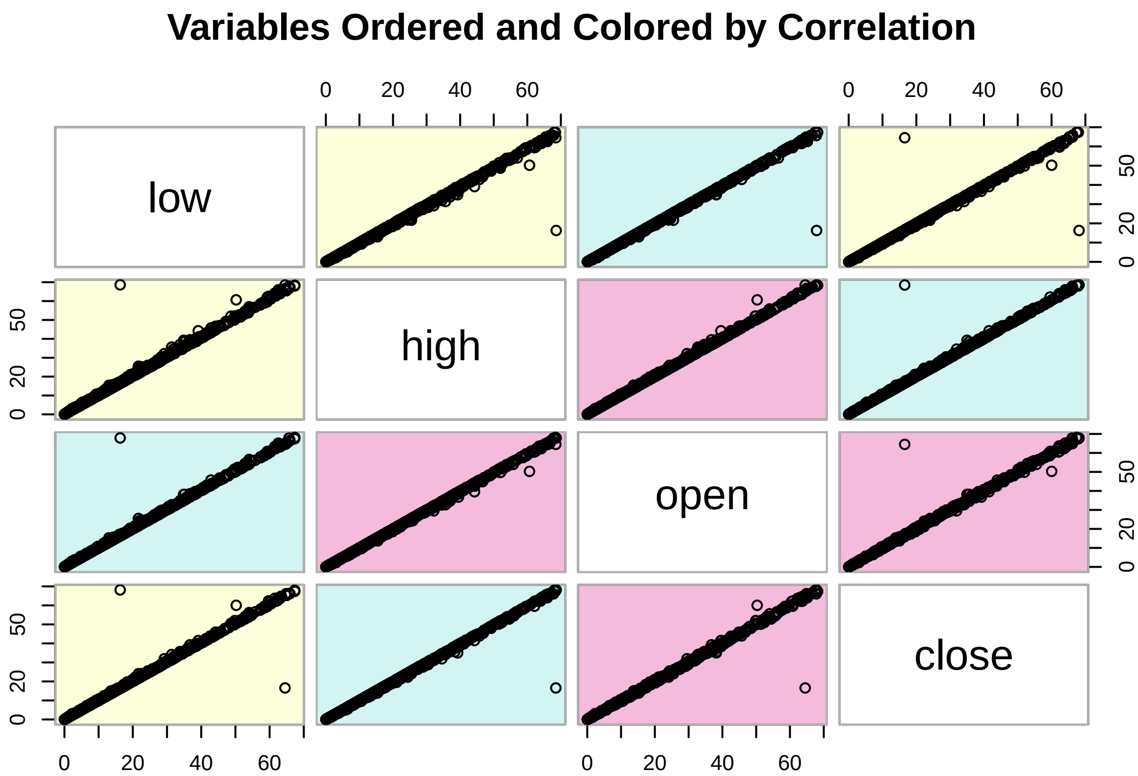
```
##    key value
## 1 open  0.51
## 2 open  6.37
## 3 open  9.60
## 4 open 19.00
## 5 open 27.12
## 6 open  8.80
```

```
library(ggplot2)
d %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density()                         # as density
```




```
# Scatterplot Matrices from the glus Package
library(gclus)
dta <- dataSetReader_Prices[c(3,6,4,5)] # get data
```

```r
dta.r <- abs(cor(dta)) # get correlations
dta.col <- dmat.color(dta.r) # get colors
# reorder variables so those with highest correlation
# are closest to the diagonal
dta.o <- order.single(dta.r)
cpairs(dta, dta.o, panel.colors=dta.col, gap=.5,
main="Variables Ordered and Colored by Correlation" )
```



Attempt our own cleaning up of data here.

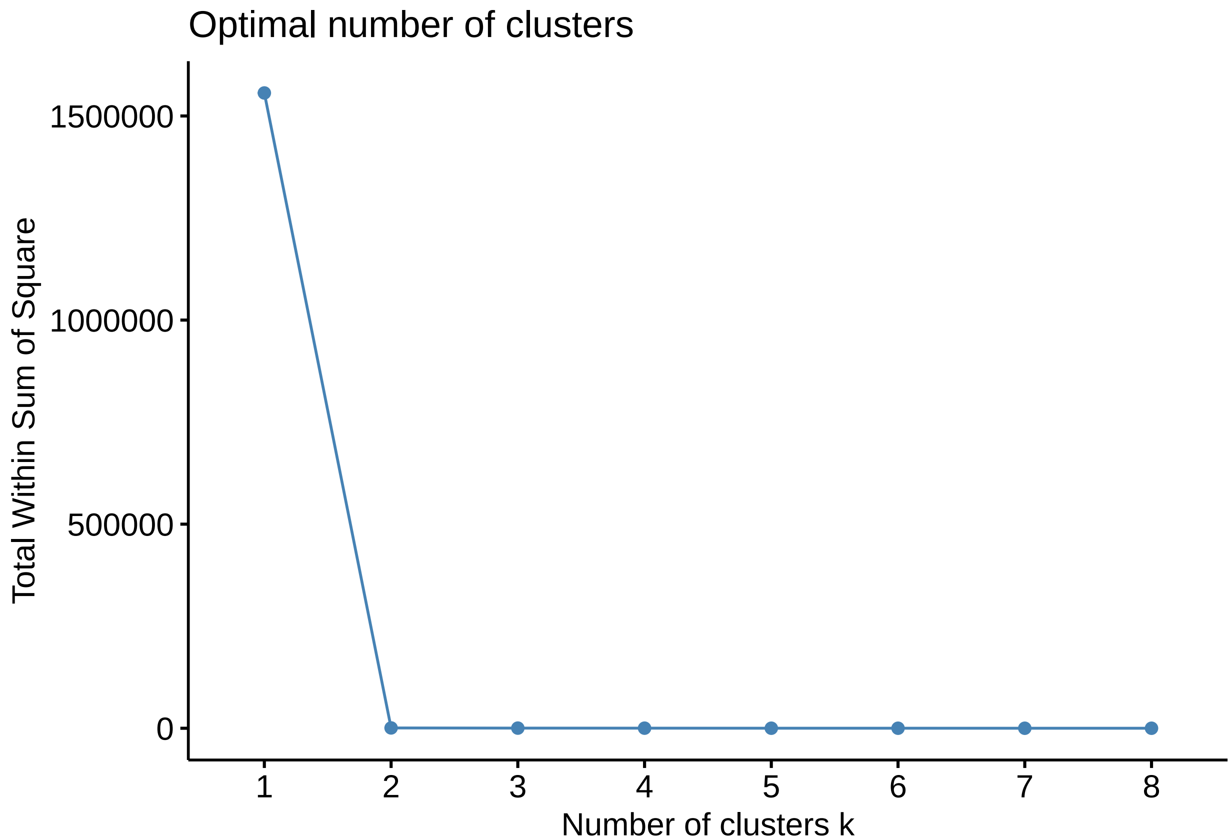
```r
# lets look at the data briefly
str(stockTradingDataKmeans)
```

```
## 'data.frame':    21170358 obs. of  9 variables:
##  $ symbol           : Factor w/ 7091 levels "A","AA","AAAP",..: 4292 4292 4292 4292 4292 4292 4292 42
##  $ date             : Factor w/ 5440 levels "1998-01-02","1998-01-05",..: 4622 1015 932 2470 4148 348
##  $ open             : num  50.8 68.8 53.4 36 41.6 ...
##  $ high             : num  52 69.8 55 36 42.3 ...
##  $ low              : num  50.8 67.8 53.2 34.6 41.5 ...
##  $ close            : num  51.8 67.9 54.3 35 42.2 ...
##  $ close_adjusted   : num  49.7 22.6 18.1 27.2 38.7 ...
##  $ volume           : num  2.00e+07 3.10e+07 4.16e+07 2.88e+08 7.46e+07 ...
##  $ split_coefficient: num  1 1 1 1 1 1 1 1 1 1 ...
```

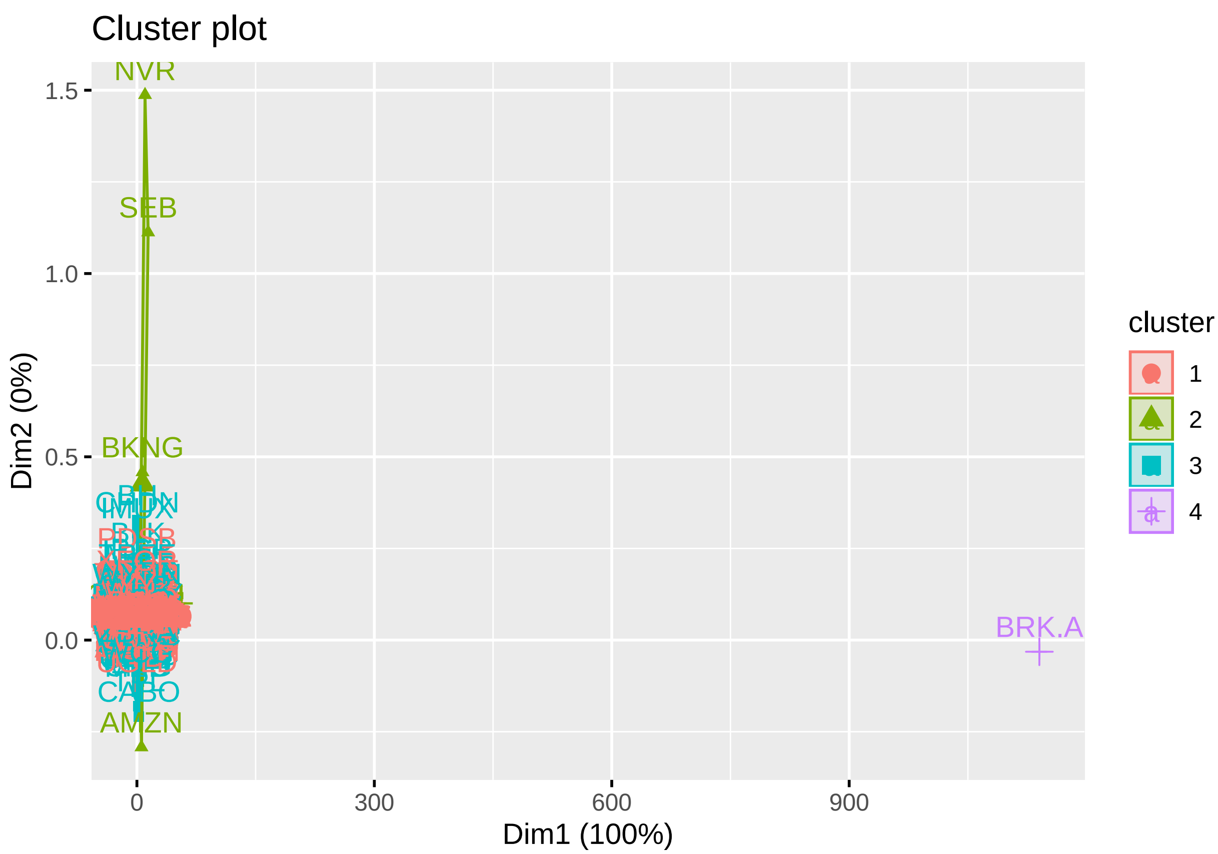
```r
# remove the split coefficient column
stockTradingDataKmeans <-stockTradingDataKmeans[, -c(9)]
```

```r
# change the date column to be in "date" format
stockTradingDataKmeans$date <- as.Date(stockTradingDataKmeans$date)
```

```r
# lets figure out the optimal amount of clusters for testna1 (symbols)
fviz_nbclust(testna1, kmeans, method = "wss", k.max = 8)
```




```r
# shows an aggregate of stock movement, based on stock symbol for center of 4
fviz_cluster(kmeansStocks2018_s4, data = testna1)
```



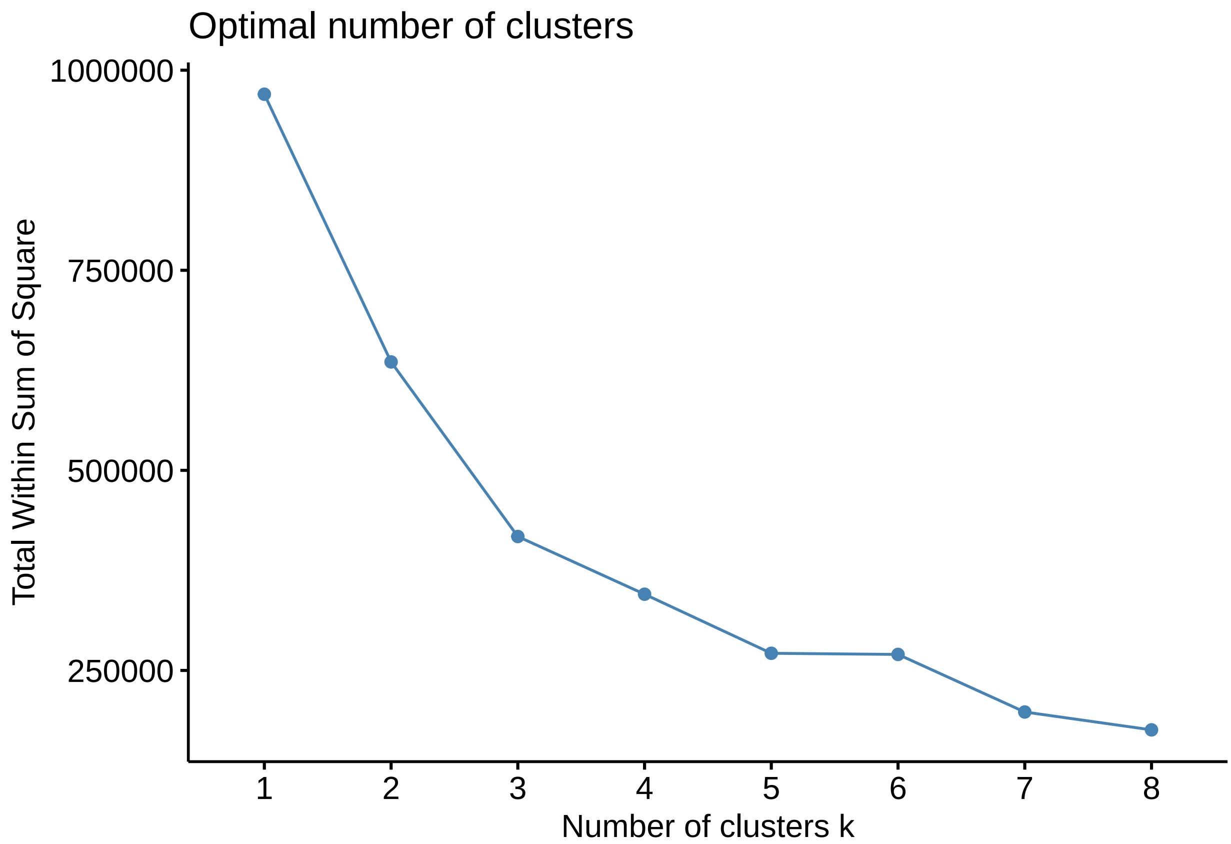
results say, remove brk.a, bkng, nvr, seb since it skews data

testna1_mod<-testna1[-c(687, 614, 3381, 4148),]

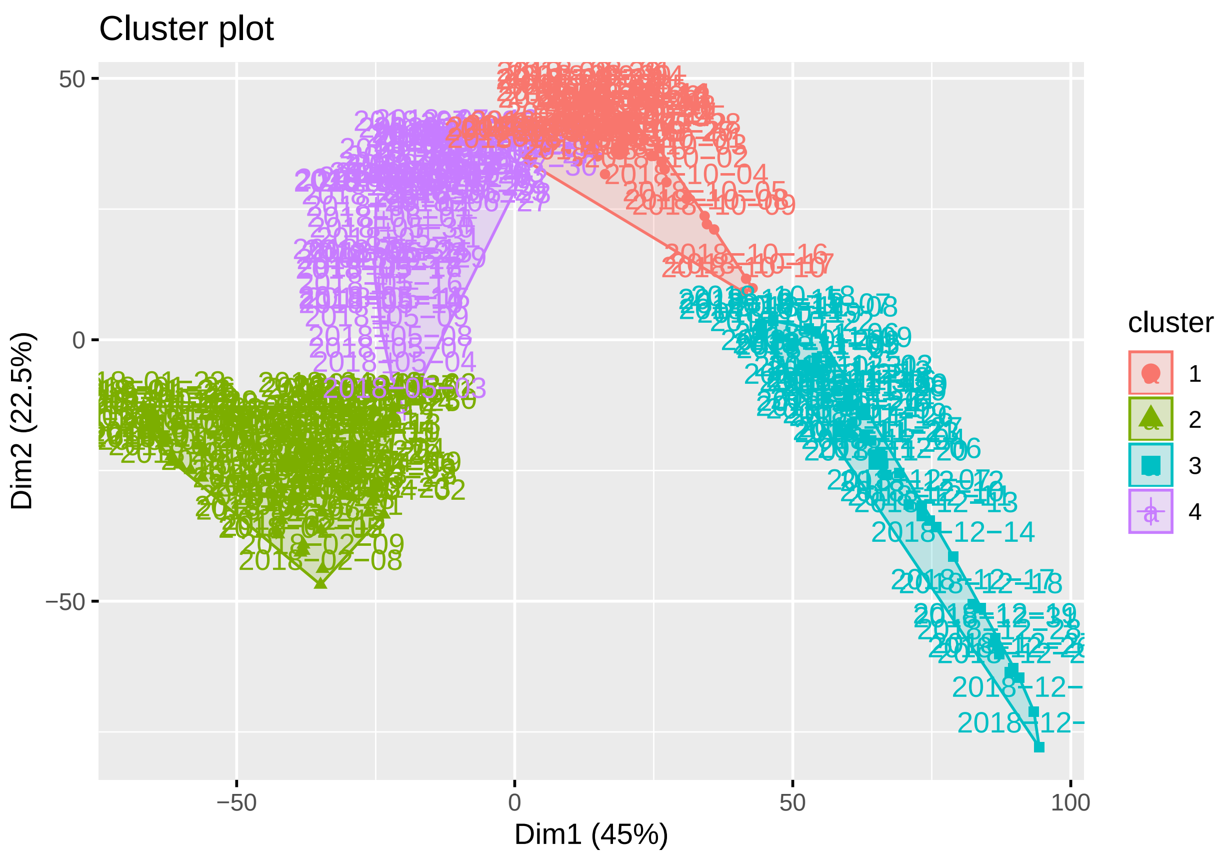
rerun with testna1 modified

testna1_mod<-testna1[-c(687, 614, 3381, 4148),] fviz_nbclust(testna1_mod, kmeans, method = "wss", k.max = 8) kmeansStocks2018_s4 <-kmeans(testna1_mod, centers = 4, iter.max = 10000) fviz_cluster(kmeansStocks2018_s4, data = testna1_mod)

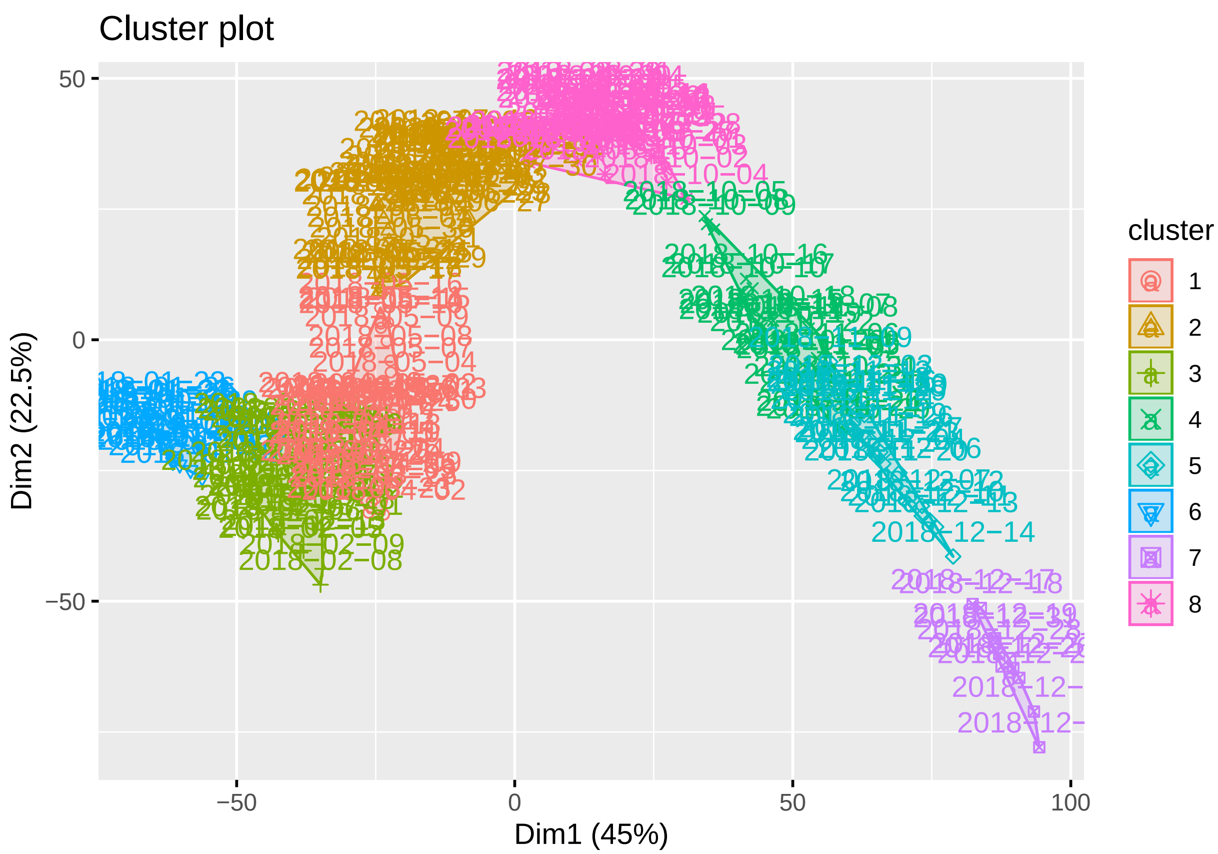
```
# lets figure out the optimal amount of clusters for testna2 (date)
fviz_nbclust(testna2, kmeans, method = "wss", k.max = 8)
```




```
# shows an aggregate of stock movement, based on trading day for center of 4
fviz_cluster(kmeansStocks2018_4, data = testna2)
```

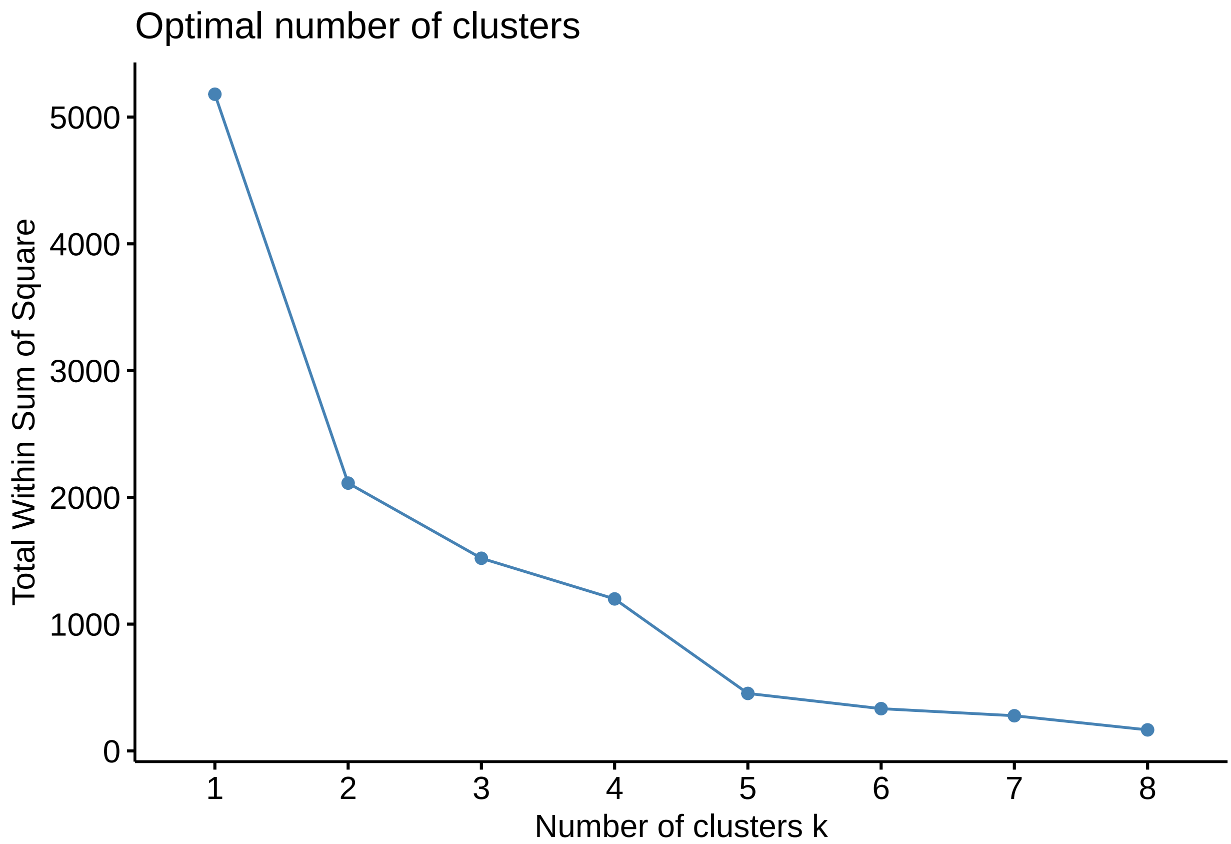



```r
# shows an aggregate of stock movement, based on trading day for center of 8
fviz_cluster(kmeansStocks2018_8, data = testna2)
```



Lets calcuate annual returns here? How should we select stocks for annual returns?

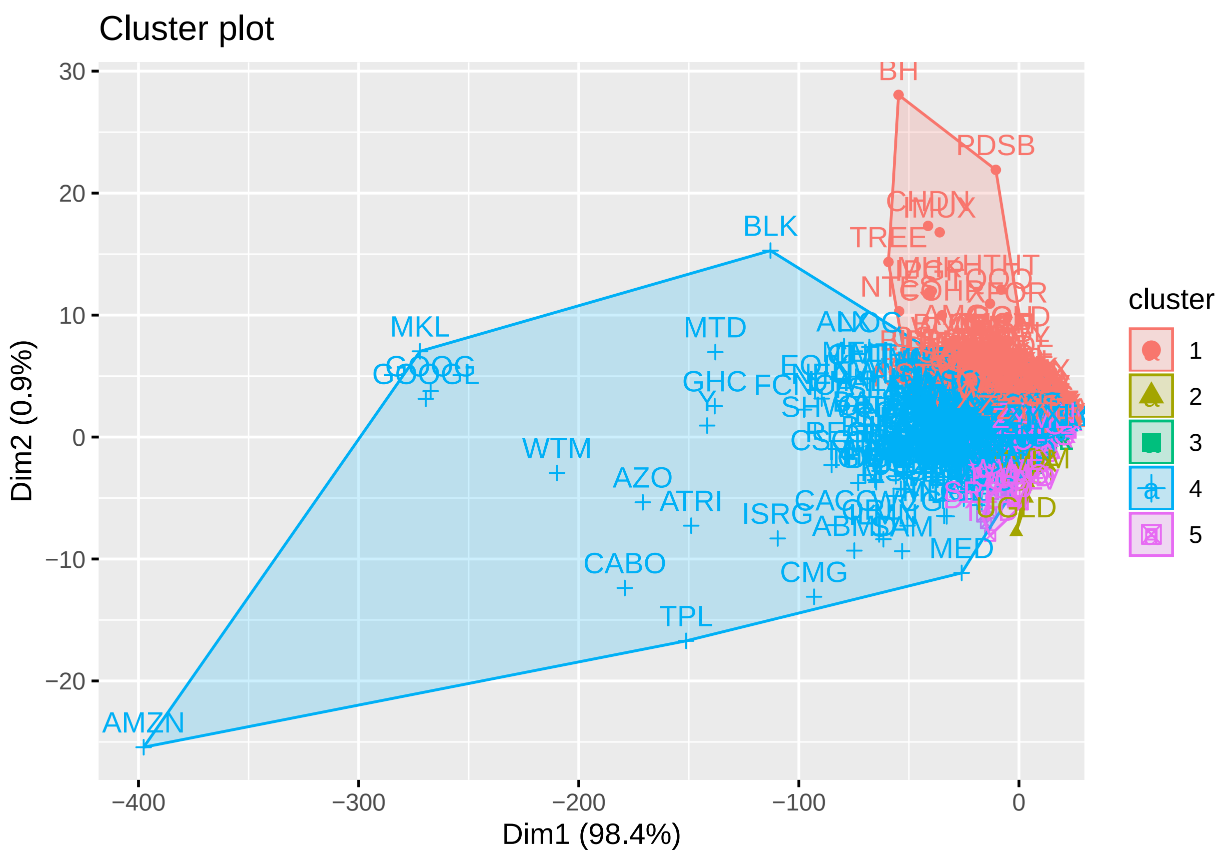
```
# lets figure out the optimal amount of clusters for annual returns
fviz_nbclust(annual_returns_scaled, kmeans, method = "wss", k.max = 8)
```



graph says 5 clusters should be fine.

```
#k means analysis by annaul returns, but omit anything that is na, with center of 5
kmeansannualreturns <-kmeans(annual_returns_scaled, centers = 5, iter.max = 10000)
```

```
# shows an aggregate of stock based on annual returns,
fviz_cluster(kmeansannualreturns, data = reshapeStockssymbolrowNONA)
```



Calculate returns based on earnings data? Generally, there's higher volatility during "earnings" season. Q1, Q2, Q3, Q4?

Data Preprocessing

To prepare the data for cluster analysis, the Earnings data was first restricted to all earnings within the year 2018. From there, the difference between the reported Earnings per Share and the Estimated Earnings per Share was calculated. Outliers were removed, and the data was then scaled and centered.

```r
#import data
initial <- read_csv("../earnings_latest.csv")
```

```
## Parsed with column specification:
## cols(
##   symbol = col_character(),
##   date = col_date(format = ""),
##   qtr = col_character(),
##   eps_est = col_character(),
##   eps = col_character(),
##   release_time = col_character()
## )
```

```r
head(initial)
```

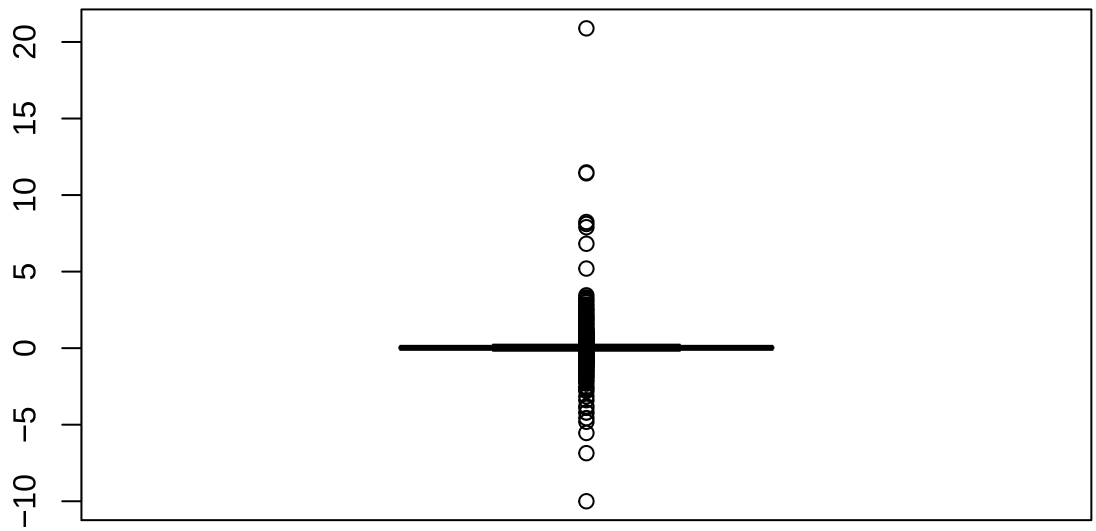
```
## # A tibble: 6 x 6
##   symbol date       qtr     eps_est eps   release_time
##   <chr>  <date>     <chr>   <chr>   <chr> <chr>
## 1 A      2009-05-14 04/2009 NULL    NULL  post
```

```
## 2 A      2009-08-17 07/2009 NULL    NULL  post
## 3 A      2009-11-13 10/2009 NULL    NULL  pre
## 4 A      2010-02-12 01/2010 NULL    NULL  pre
## 5 A      2010-05-17 04/2010 NULL    NULL  post
## 6 A      2010-08-16 07/2010 NULL    NULL  post
```

```r
#restrict data to 2018
stocks2018 <- initial[initial$date >= "2018-01-01" & initial$date <= "2018-12-31",]

#replace NULL values with NA
stocks2018$eps_est <- gsub("NULL", NA, stocks2018$eps_est)

#drop unnecessary columns
stocks2018$release_time <- NULL
stocks2018$qtr <- NULL
stocks2018$date <- NULL

#drop all incomplete cases
stocks2018 <- stocks2018[complete.cases(stocks2018),]

#cast numeric data as.numeric
stocks2018$eps <- as.numeric(stocks2018$eps)
stocks2018$eps_est <- as.numeric(stocks2018$eps_est)

#create column to represent difference between estimate and actual earnings per share
stocks2018$diff <- stocks2018$eps - stocks2018$eps_est

str(stocks2018)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    13048 obs. of  4 variables:
##  $ symbol : chr  "A" "A" "A" "A" ...
##  $ eps_est: num  0.58 0.65 0.63 0.73 1.23 0.6 1.33 0.25 0.04 0.1 ...
##  $ eps    : num  0.66 0.65 0.67 0.81 1.04 0.77 1.52 0.63 0.1 0.13 ...
##  $ diff   : num  0.08 0 0.04 0.08 -0.19 0.17 0.19 0.38 0.06 0.03 ...
```

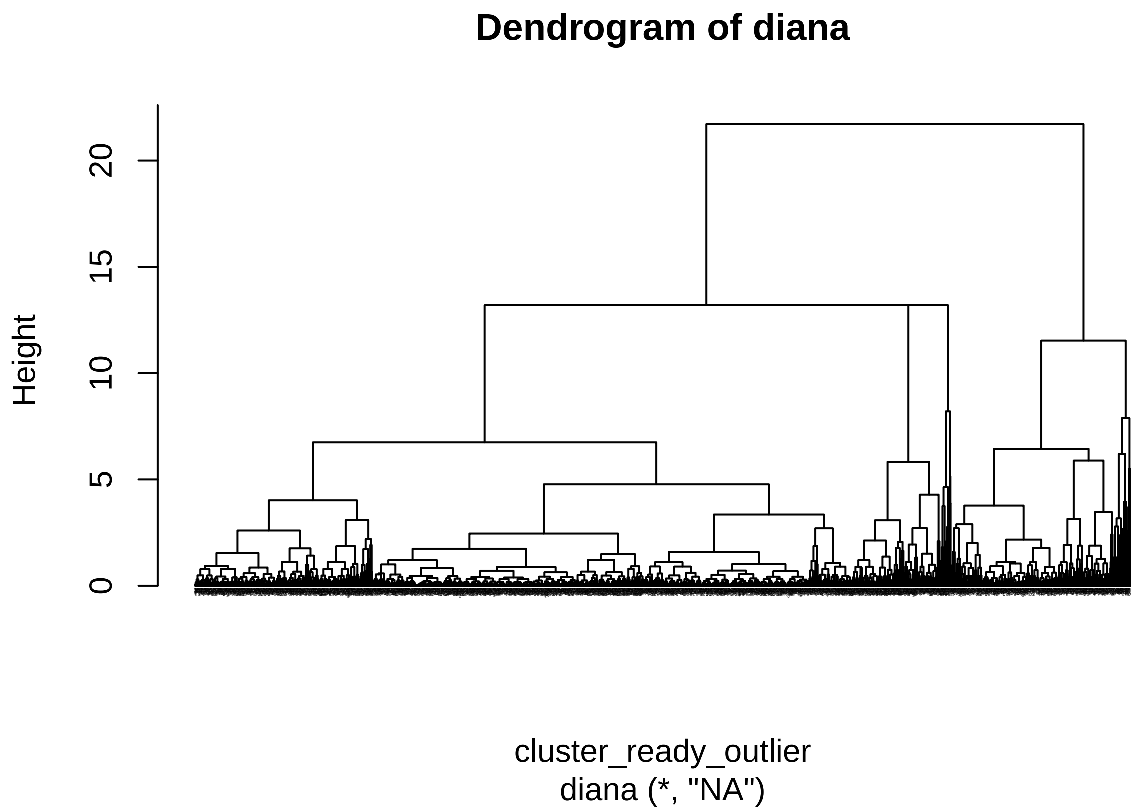
```r
#remove outliers
mean_outlier <- boxplot(stocks2018$diff)$out
```




```r
stocks2018_mean_noOutlier <- stocks2018[-which(stocks2018$diff %in% mean_outlier),]

#get mean of all data by symbol
stocks2018_mean_noOutlier <- stocks2018_mean_noOutlier %>% group_by(symbol) %>% summarise_all(mean)

#change row names to be stock symbols
stocks2018_mean_noOutlier <- column_to_rownames(stocks2018_mean_noOutlier, loc=1)

#scale and center all data
cluster_ready_outlier <- scale(stocks2018_mean_noOutlier)
```

Hierarchical Clustering: Divisive Method

```r
divMeanOutlier <- diana(cluster_ready_outlier)

#display dendrogram of DIANA algorithm
pltree(divMeanOutlier, cex = 0.1, hang = -1, main = "Dendrogram of diana")
```



Hierarchical Clustering: Agglomerative Method

To determine the ideal agglomerative clustering method, models were created using all four different types of distance algorithms: complete, single, average, and Ward's. Ward's distance was found to have the largest agglomerative coefficient (0.999), which indicated a strong tendency towards clustering.

```
#run agglomerative clustering with four different measures of distance: complete, single, average, ward
agglComplete <- agnes(cluster_ready_outlier, method="complete")
agglSingle <- agnes(cluster_ready_outlier, method="single")
agglAverage <- agnes(cluster_ready_outlier, method="average")
agglWard <- agnes(cluster_ready_outlier, method="ward")

#display agglomerative coefficient
agglComplete$ac
```

```
## [1] 0.9970553
```

```
agglSingle$ac
```

```
## [1] 0.9858992
```

```
agglAverage$ac
```

```
## [1] 0.9939901
```

```
agglWard$ac
```

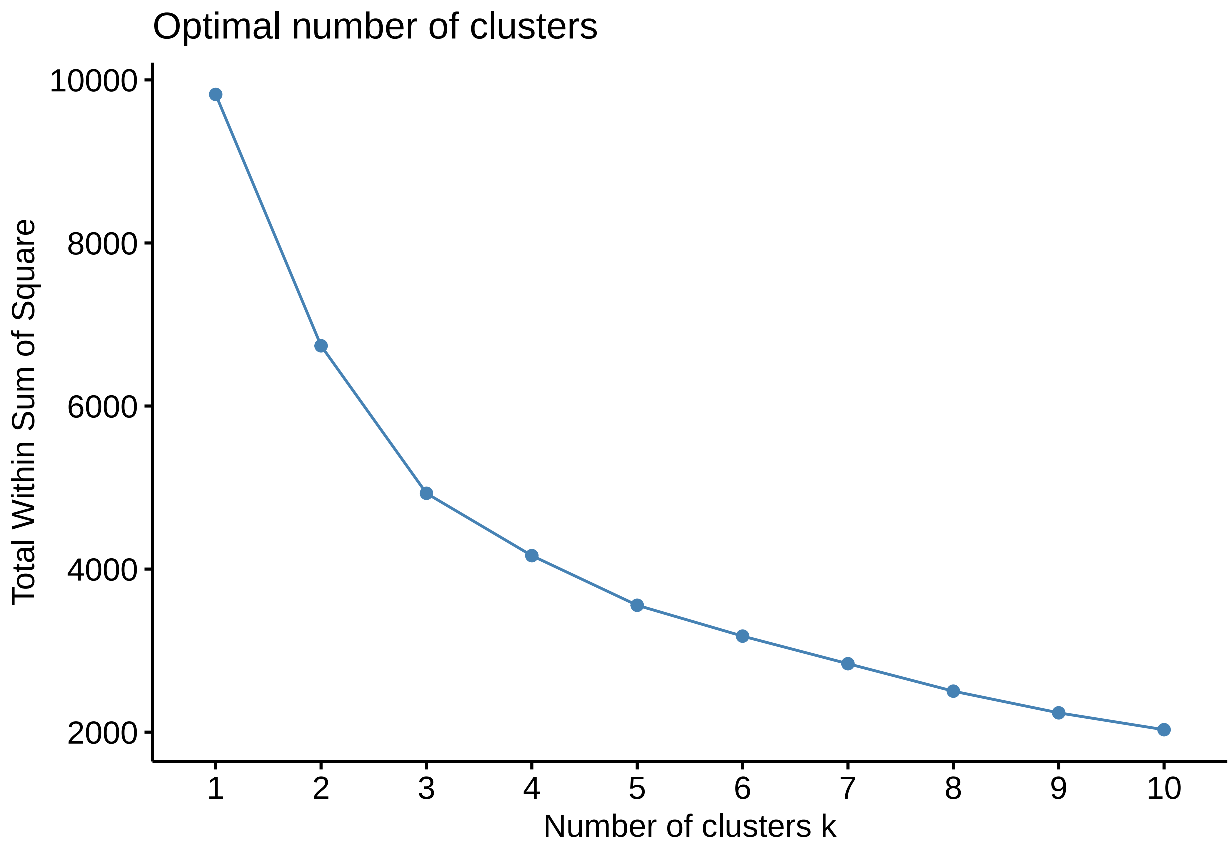
```
## [1] 0.9991861
```

```r
#display dendrogram of clustering using Ward distance
pltree(agglWard, cex = 0.1, hang = -1, main = "Dendrogram of agnes")
```

Optimal Clustering

An elbow diagram was created to determine that the optimal number of clusters was three.

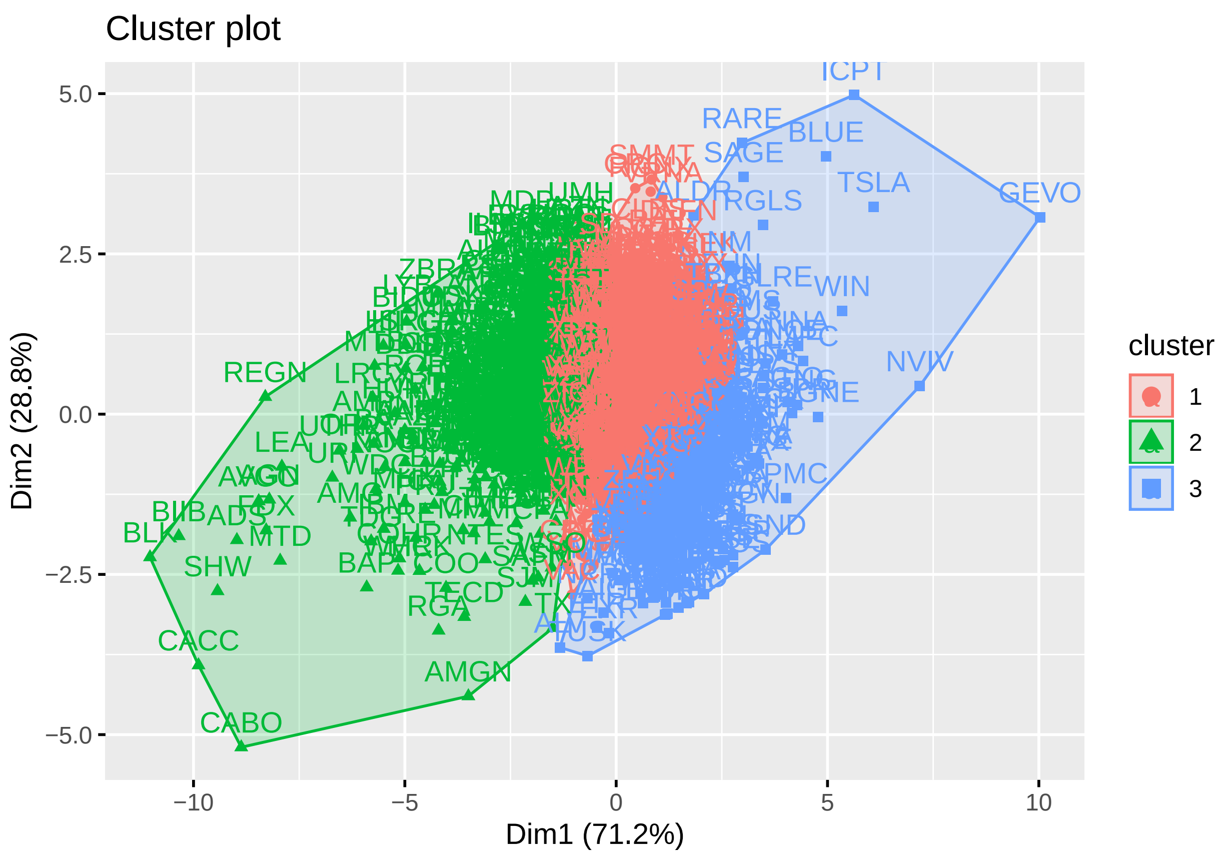
```r
#create elbow diagram
fviz_nbclust(cluster_ready_outlier, FUN = hcut, method = "wss")
```



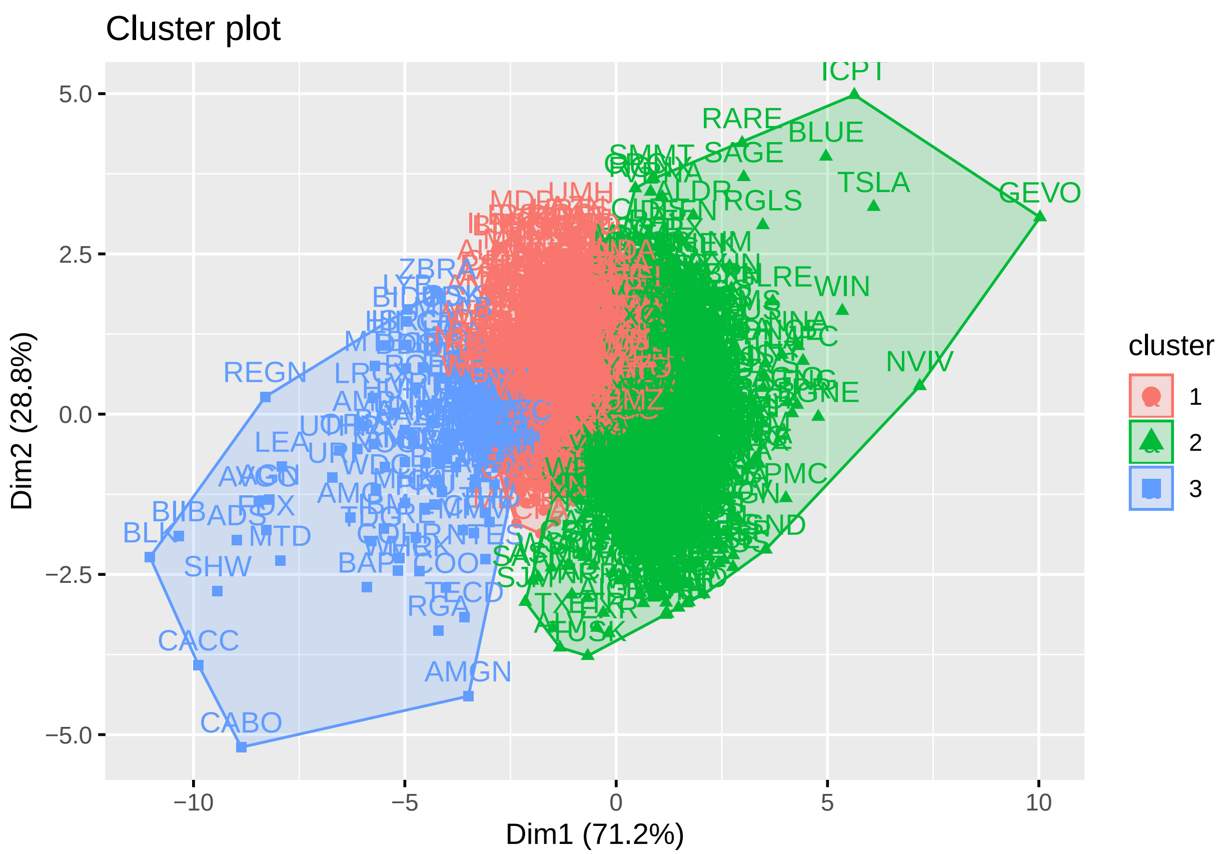

```r
#create k=3 clusters
sub_grp_diana <- cutree(as.hclust(divMeanOutlier), k=3)
sub_grp_agnes <- cutree(as.hclust(agglWard), k=3)
```

Final Clustering

```r
#divisive
fviz_cluster(list(data = cluster_ready_outlier, cluster = sub_grp_diana))
```




```r
#agglomerative
fviz_cluster(list(data = cluster_ready_outlier, cluster = sub_grp_agnes))
```


Cluster plot
Dim2 (28.8%)
Dim1 (71.2%)
5.0
2.5
0.0
−2.5
−5.0
−10
−5
0
5
10
cluster
1
2
3
ICPT
RARE
BLUE
TSLA
GEVO
NVIV
WIN
REGN
LEA
SHW
CACC
CABO
AMGN
BLK
MTD
RGA
TECD
RGLS
PMC