# TDA Analysis

*Yakub Akhmerov*

*May 9, 2018*

Intro: I began the semester with a lingering feeling of immersing myself into something intellectually stimulating. I stumbled across topology in my Real Analysis course fall of 2017. I generally enjoyed working on it and noticed the beautiful generalized, objective theory behind it. Since my field is data science and analytics, I figured it would be appropriate to combine two fields which I am genuinely interested in and go from through. Thus, I stumbled upon topological data analysis.

I reached out to several profession sources and eventually stumbled upon Dmitriy Mozorov at the Berkeley Institute of Data Science. He advised me in this research and was a great in my concrete understanding of the subject. To demonstrate my new found abilities in topological data analysis, I choose to apply my understanding to a project.

### Topological Data Analysis

### Problem: How well can TDA be used on cyrptocurrency data?

### Testing on Binance API Data

### Description: We provide daily cryptocurrency data (transaction count, on-chain transaction volume, value of created coins, price, market cap, and exchange volume) in CSV format. The data sample stretches back to December 2013. We explain the column titles and some shortcomings here.

Source: https://coinmetrics.io/data-downloads/ ## ID sample code number (not unique). #V1 date: Date on georgian calender. #V2 txVolume(USD): "on-chain transaction volume." Essentially a broad and largely unadjusted measure of the total value of outputs on the blockchain, on a given day. #V3 txCount: Number of transactions happening on the public blockchain a day. The is a broad estimation. More can be explained in the source as to why that is. #V4 marketcap(USD): The unit price multiplied by the number of units in circulation. #V5 Price: Open price, received from CoinMarketCap. #V6 exchangevolume(USD): The dollar value of the volume at exchanges. Also received from CoinMarketCap. #V7 generatedCoins: The number of new coins that have been brought into existence on that day #V8 Fees: Fees. One important peice of information is that the fees in this data are based on the native currency, not USD.

## Packages needed for the analysis

**(1) Cleaning the data. A lot of wrangling needed to be done. The variables were put into strings in the original dataset so they had to be converted into variables in order to do any sort of analysis.**

```
setwd("~/Documents/TDA/")

btc = read.csv("btc.csv", header = F)
btc = btc[-1,]
btc <- btc[1:8]
```

```r
btc$V2 <- as.numeric(as.character(btc$V2))
btc$V3 <- as.numeric(as.character(btc$V3))
btc$V4 <- as.numeric(as.character(btc$V4))
btc$V5 <- as.numeric(as.character(btc$V5))
btc$V6 <- as.numeric(as.character(btc$V6))
btc$V7 <- as.numeric(as.character(btc$V7))
btc$V8 <- as.numeric(as.character(btc$V8))
breaks <- c(65, 2565, 5065, 7565, 10065, 12065, 15065, 17065, 20065)
btc <- btc %>%
  mutate(Price_bin = cut(V5, breaks)) #categorize the price of the coin into 12 bins.

head(btc) # see header to decode V1-V8;
```

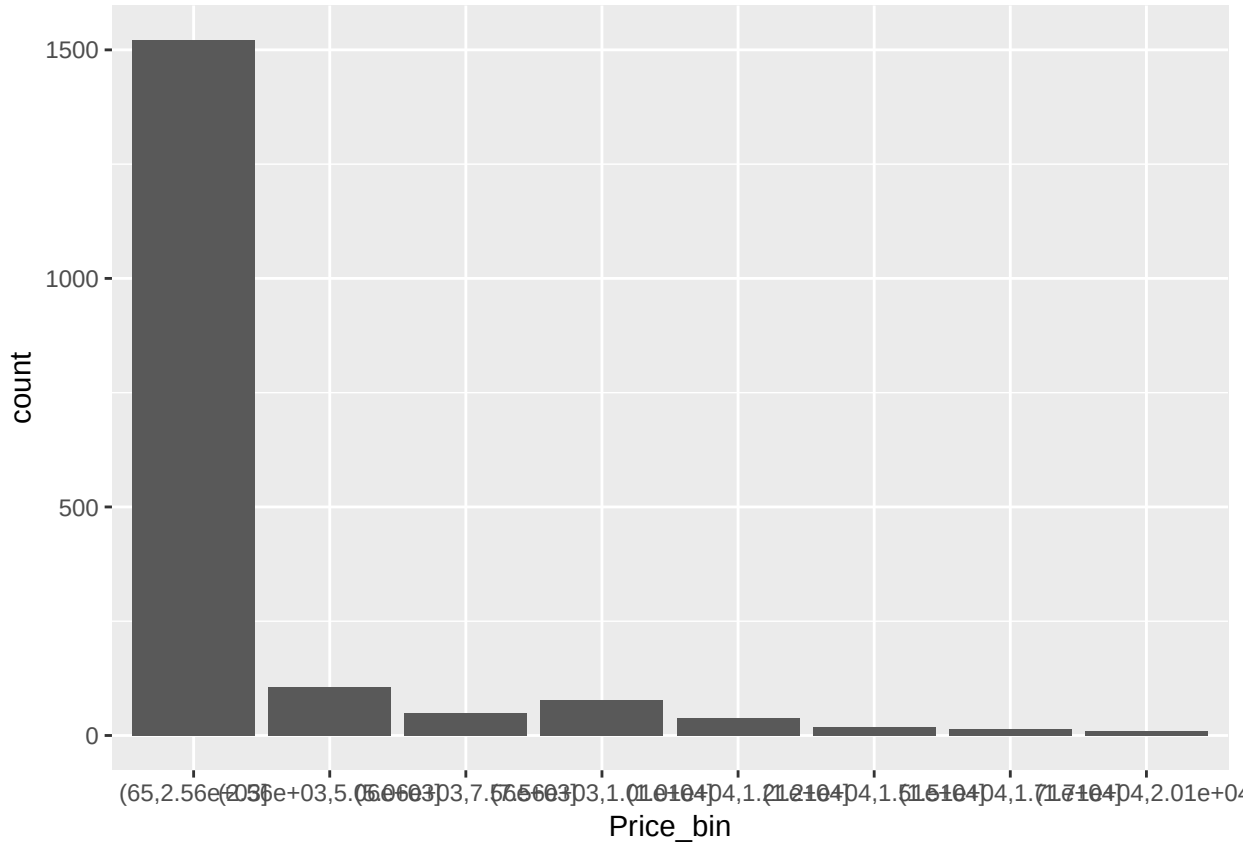
```
##           V1        V2    V3         V4     V5 V6   V7       V8
## 1 2013-05-01 108659660 52443 1542820000 139.00  0 3575 36.80600
## 2 2013-05-02  96958519 55169 1292190000 116.38  0 3425 54.40792
## 3 2013-05-03  84459697 55636 1180070000 106.25  0 3650 48.52677
## 4 2013-05-04  41545845 48595 1089890000  98.10  0 3900 43.41969
## 5 2013-05-05  56205930 49907 1254760000 112.90  0 3875 38.40896
## 6 2013-05-06  78921301 52523 1289470000 115.98  0 4250 51.88924
##       Price_bin
## 1 (65,2.56e+03]
## 2 (65,2.56e+03]
## 3 (65,2.56e+03]
## 4 (65,2.56e+03]
## 5 (65,2.56e+03]
## 6 (65,2.56e+03]
```

```r
summary(btc)
```

```
##           V1             V2                  V3
##  2013-05-01:   1   Min.   :3.379e+07   Min.   : 29566
##  2013-05-02:   1   1st Qu.:2.362e+08   1st Qu.: 71728
##  2013-05-03:   1   Median :6.326e+08   Median :154048
##  2013-05-04:   1   Mean   :2.580e+09   Mean   :162430
##  2013-05-05:   1   3rd Qu.:1.756e+09   3rd Qu.:234748
##  2013-05-06:   1   Max.   :4.835e+10   Max.   :490459
##  (Other)   :1828
##        V4                  V5                V6                  V7
##  Min.   :7.793e+08   Min.   :   68.5   Min.   :0.000e+00   Min.   :1000
##  1st Qu.:4.030e+09   1st Qu.:  282.5   1st Qu.:1.713e+07   1st Qu.:2000
##  Median :7.113e+09   Median :  530.1   Median :4.720e+07   Median :3575
##  Mean   :2.915e+10   Mean   : 1791.2   Mean   :1.103e+09   Mean   :3230
##  3rd Qu.:1.605e+10   3rd Qu.: 1013.5   3rd Qu.:2.010e+08   3rd Qu.:4050
##  Max.   :3.261e+11   Max.   :19475.8   Max.   :2.384e+10   Max.   :6450
##
##        V8                          Price_bin
##  Min.   :   7.856   (65,2.56e+03]      :1521
##  1st Qu.:  15.406   (2.56e+03,5.06e+03]: 105
##  Median :  35.250   (7.56e+03,1.01e+04]:  77
##  Mean   :  89.874   (5.06e+03,7.56e+03]:  50
##  3rd Qu.:  84.965   (1.01e+04,1.21e+04]:  39
##  Max.   :1495.947   (1.21e+04,1.51e+04]:  19
```

```
##                   (Other)            :  23
```

```
ggplot(btc) + aes(x = Price_bin) + geom_bar() #examine how the bins are distributed
```




```
head(btc) # see header to decode V1-V8;
```

```
##           V1        V2    V3         V4     V5 V6   V7       V8
## 1 2013-05-01 108659660 52443 1542820000 139.00  0 3575 36.80600
## 2 2013-05-02  96958519 55169 1292190000 116.38  0 3425 54.40792
## 3 2013-05-03  84459697 55636 1180070000 106.25  0 3650 48.52677
## 4 2013-05-04  41545845 48595 1089890000  98.10  0 3900 43.41969
## 5 2013-05-05  56205930 49907 1254760000 112.90  0 3875 38.40896
## 6 2013-05-06  78921301 52523 1289470000 115.98  0 4250 51.88924
##       Price_bin
## 1 (65,2.56e+03]
## 2 (65,2.56e+03]
## 3 (65,2.56e+03]
## 4 (65,2.56e+03]
## 5 (65,2.56e+03]
## 6 (65,2.56e+03]
```

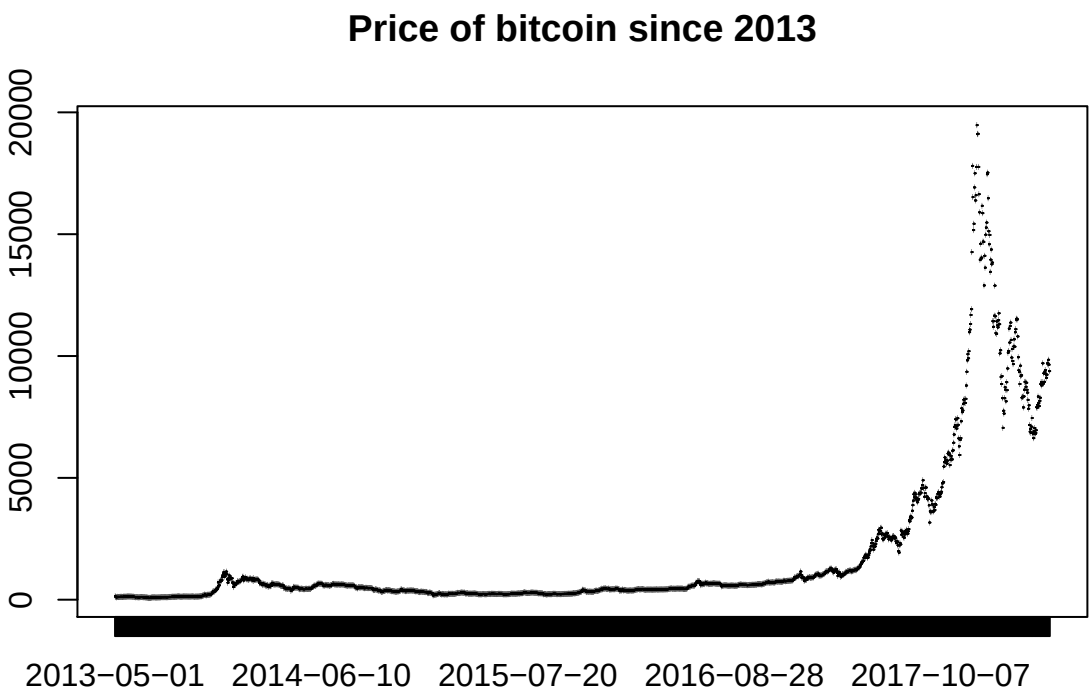
```
summary(btc) #what we're starting with
```

```
##           V1             V2                  V3
##  2013-05-01:   1   Min.   :3.379e+07   Min.   : 29566
##  2013-05-02:   1   1st Qu.:2.362e+08   1st Qu.: 71728
##  2013-05-03:   1   Median :6.326e+08   Median :154048
##  2013-05-04:   1   Mean   :2.580e+09   Mean   :162430
##  2013-05-05:   1   3rd Qu.:1.756e+09   3rd Qu.:234748
```

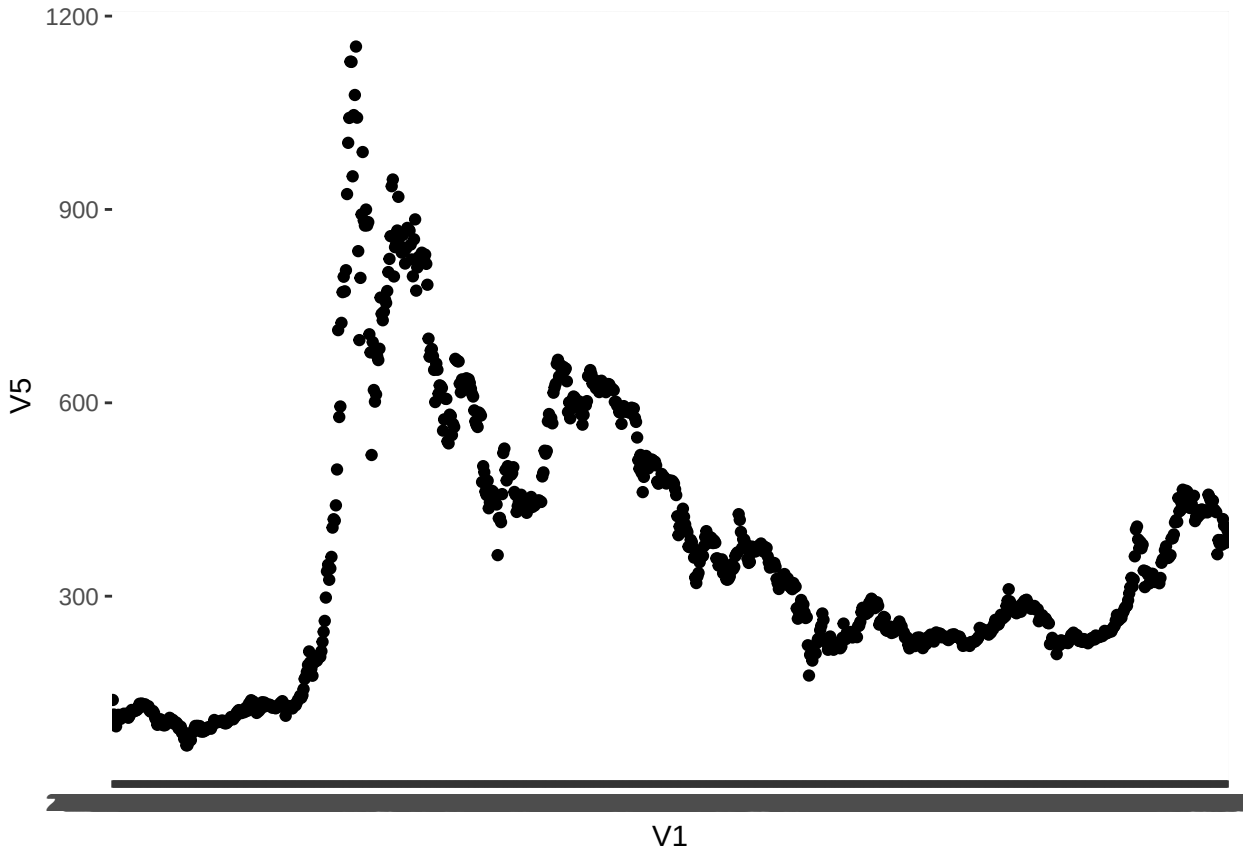
```
##  2013-05-06:   1   Max.   :4.835e+10   Max.   :490459
##  (Other)   :1828
##        V4                  V5                V6                  V7
##  Min.   :7.793e+08   Min.   :   68.5   Min.   :0.000e+00   Min.   :1000
##  1st Qu.:4.030e+09   1st Qu.:  282.5   1st Qu.:1.713e+07   1st Qu.:2000
##  Median :7.113e+09   Median :  530.1   Median :4.720e+07   Median :3575
##  Mean   :2.915e+10   Mean   : 1791.2   Mean   :1.103e+09   Mean   :3230
##  3rd Qu.:1.605e+10   3rd Qu.: 1013.5   3rd Qu.:2.010e+08   3rd Qu.:4050
##  Max.   :3.261e+11   Max.   :19475.8   Max.   :2.384e+10   Max.   :6450
##
##        V8                        Price_bin
##  Min.   :   7.856   (65,2.56e+03]      :1521
##  1st Qu.:  15.406   (2.56e+03,5.06e+03]: 105
##  Median :  35.250   (7.56e+03,1.01e+04]:  77
##  Mean   :  89.874   (5.06e+03,7.56e+03]:  50
##  3rd Qu.:  84.965   (1.01e+04,1.21e+04]:  39
##  Max.   :1495.947   (1.21e+04,1.51e+04]:  19
##                     (Other)            :  23
```

## EDA

```
plot(x = btc$V1, y = btc$V5, main = "Price of bitcoin since 2013")
```

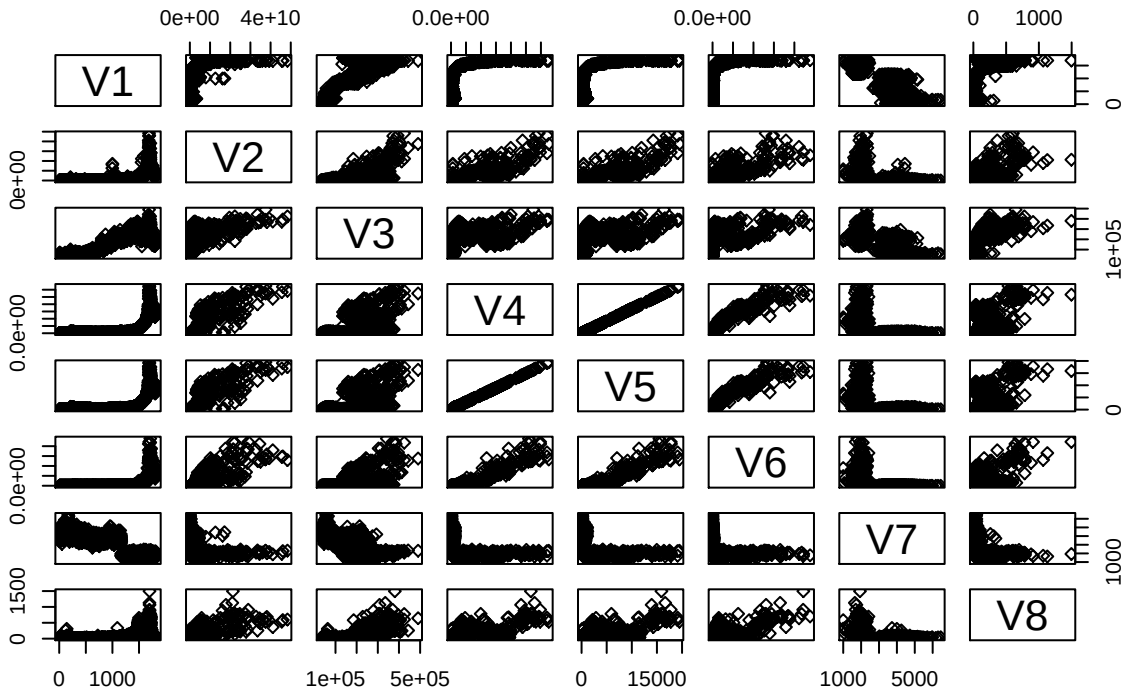



```
ggplot(btc[1:1000,]) + aes(x=V1, y = V5) + geom_point()
```




```
#Examine how the bins are distributed

plot(btc[1:8], main="bitcoin data set", pch=23, bg = c("red", "green")
     [unclass(btc$price)])
```




```
#Examine how all the variables interact with one another


data <- btc[2:8] ## Take out date variable
D = dist(scale(data)) # use Euclidean distance on data
## DIST: This function computes and returns the distance matrix computed by using the specified distanc
```

**DIFFUSE: Description : Uses the pair-wise distance matrix for a data set to compute the diffusion map coefficients. Computes the Markov transition probability matrix, and its eigenvalues and left & right eigenvectors. Returns a 'dmap' object.**

The following code cannot be run due to technical difficulties with the igraph package. However rather than putting it into a code chance I figured it'd be good to leave it here and work on it once Rstudio fixes the bugs in this package and updates it.

dmap = diffuse(D, eps.val=10, t=1, neigen=2) ## just run with the standard default settings plot(dmap$X[,1], dmapX[,2],col=outcome,pch=paste(outcome), xlab="Diffusion Map Coordinate 1", ylab="Diffusion Map Coordinate 2", main="Diffusion Map of Bitcoin data")

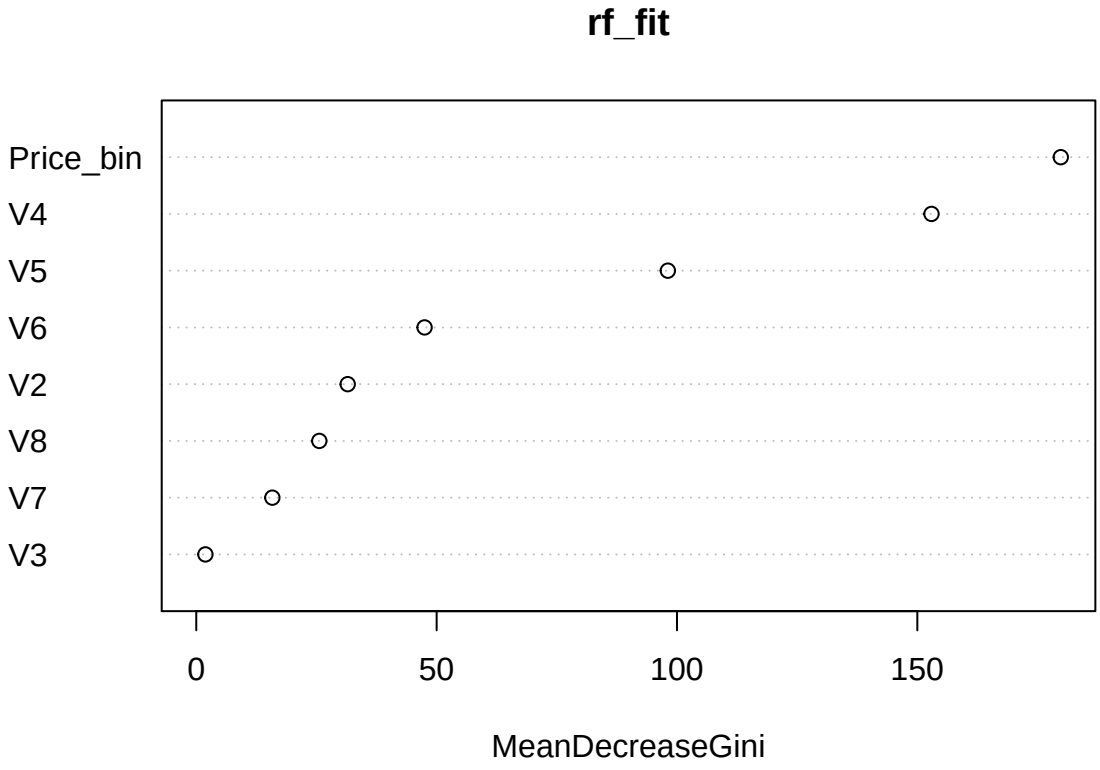
## 2) OK - now Use random forest "Department" classifier to define distances

```
bins = btc$Price_bin
data = btc[2:9]
```

```r
rf_fit = randomForest(data, bins, ntree=10, proximity=TRUE)
print(rf_fit)
```

```
##
## Call:
##  randomForest(x = data, y = bins, ntree = 10, proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 10
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.44%
## Confusion matrix:
##                      (65,2.56e+03] (2.56e+03,5.06e+03] (5.06e+03,7.56e+03]
## (65,2.56e+03]                 1496                    0                    0
## (2.56e+03,5.06e+03]              0                  104                    0
## (5.06e+03,7.56e+03]              0                    0                   48
## (7.56e+03,1.01e+04]              0                    0                    0
## (1.01e+04,1.21e+04]              0                    0                    0
## (1.21e+04,1.51e+04]              0                    0                    1
## (1.51e+04,1.71e+04]              0                    0                    0
## (1.71e+04,2.01e+04]              0                    0                    0
##                      (7.56e+03,1.01e+04] (1.01e+04,1.21e+04]
## (65,2.56e+03]                          0                    0
## (2.56e+03,5.06e+03]                    0                    0
## (5.06e+03,7.56e+03]                    1                    0
## (7.56e+03,1.01e+04]                   77                    0
## (1.01e+04,1.21e+04]                    2                   37
## (1.21e+04,1.51e+04]                    0                    0
## (1.51e+04,1.71e+04]                    0                    0
## (1.71e+04,2.01e+04]                    0                    0
##                      (1.21e+04,1.51e+04] (1.51e+04,1.71e+04]
## (65,2.56e+03]                          0                    0
## (2.56e+03,5.06e+03]                    0                    0
## (5.06e+03,7.56e+03]                    0                    0
## (7.56e+03,1.01e+04]                    0                    0
## (1.01e+04,1.21e+04]                    0                    0
## (1.21e+04,1.51e+04]                   16                    2
## (1.51e+04,1.71e+04]                    2                   11
## (1.71e+04,2.01e+04]                    0                    0
##                      (1.71e+04,2.01e+04] class.error
## (65,2.56e+03]                          0  0.00000000
## (2.56e+03,5.06e+03]                    0  0.00000000
## (5.06e+03,7.56e+03]                    0  0.02040816
## (7.56e+03,1.01e+04]                    0  0.00000000
## (1.01e+04,1.21e+04]                    0  0.05128205
## (1.21e+04,1.51e+04]                    0  0.15789474
## (1.51e+04,1.71e+04]                    0  0.15384615
## (1.71e+04,2.01e+04]                    9  0.00000000
```

```r
varImpPlot(rf_fit)
```




```
#version
D2 = 1-rf_fit$proximity # use 1 - proximity
```

**Due to the igraph package this code could not be run either.**

dmap2 = diffuse(D2,eps.val=40, t=.01, neigen=2) #original dmap1 = diffuse(D1,eps.val=.1, t=1, neigen=2) head(dmap2)

cluster2 = hclust(dist(dmap2$X[,1:2])) plot(cluster2); abline(h=2.0, col='red',lwd=3)

plot(dmap2$X[,1], dmap2X[,2],col=outcome,pch=paste(outcome), xlab="Diffusion Map Coordinate 1", ylab="Diffusion Map Coordinate 2")

dmap = diffuse(D3,eps.val=400, t=1, neigen=2) plot(dmap3$X[,1], dmap3X[,2],col=outcome,pch=paste(outcome), xlab="Diffusion Map Coordinate 1", ylab="Diffusion Map Coordinate 2", main = "BTC Random Forest Prediction") ## using RF method we generate plot similar t

**The actual TDA plot. Due to rstudio difficulties it could not be seen either**

TDA_plot <- gvisBubbleChart(rf_fit, idvar="cluster", xvar="x", yvar="y", colorvar="count", sizevar="count", options=list( title='BTC Clustered Data - TDA Exploration', hAxis='{minValue:-5, maxValue:6}', vAxis='{minValue:-2, maxValue:7}' ) ) plot(TDA_plot)