noise detection

New

January 28, 2020

```
OSCE <- read.csv("C:/Users/LUFEMOS/Desktop/Untitled spreadsheet - OSCE 
Results.csv")
```

upload package psych to examine the descriptive statistics of the station score by group

```
library(psych)
```

This command computes the descriptive statistics of the station score across the 5 groups

```
describeBy(OSCE$station_score,OSCE$location_index)

## 
##  Descriptive statistics by group 
## group: 1
##    vars  n  mean    sd median trimmed  mad   min max range  skew kur
tosis
## X1    1 43 81.57 12.24  83.75   82.25 9.27 48.75 100 51.25 -0.46    
-0.22
##      se
## X1 1.87
## ------------------------------------------------------------ 
## group: 2
##    vars   n mean    sd median trimmed   mad   min max range  skew ku
rtosis
## X1    1 141 73.5 15.43     75   74.26 14.83 28.75 100 71.25 -0.42   
-0.05
##     se
## X1 1.3
## ------------------------------------------------------------ 
## group: 3
##    vars  n  mean   sd median trimmed   mad  min max range  skew kurt
osis
## X1    1 60 77.98 16.5  81.88   79.77 15.75 37.5 100  62.5 -0.75    -
0.34
##      se
## X1 2.13
## ------------------------------------------------------------ 
## group: 4
##    vars   n  mean    sd median trimmed   mad  min max range  skew ku
rtosis
```

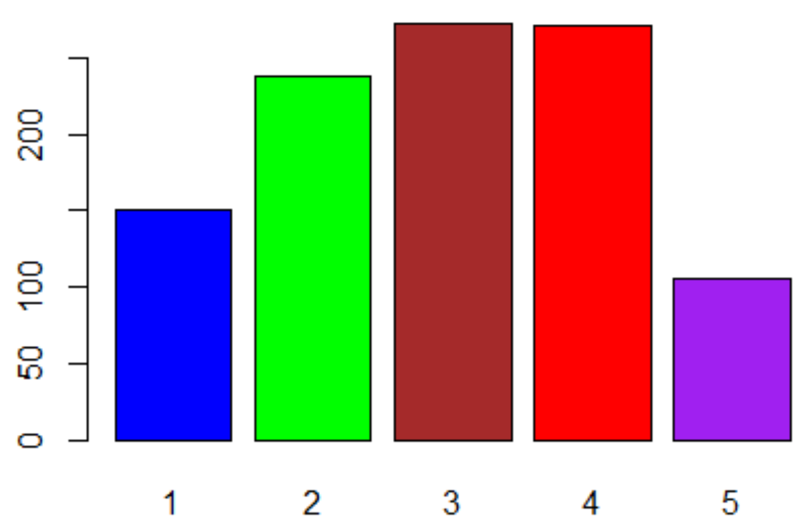
```
## X1    1 152 72.12 16.47   73.75    72.32 14.83 8.75 100 91.25 -0.28
0.37
##      se
## X1 1.34
## --------------------------------------------------------
## group: 5
##    vars  n  mean    sd median trimmed  mad   min   max range skew ku
rtosis
## X1    1 25 79.75 10.25     80   80.54 9.27 46.25 93.75  47.5 -1.1
2.2
##      se
## X1 2.05
```

This code computes the variance of the station score by group to examine the group with the highest dispersion (NOISE)

```
ag <- aggregate(station_score~ location_index, data = OSCE, var)
dispersion=xtabs(station_score ~ ., data = ag)
```

This code plots the level of dispersion across the groups

```
barplot(dispersion, col=c("blue", "green","brown","red","purple"))
```



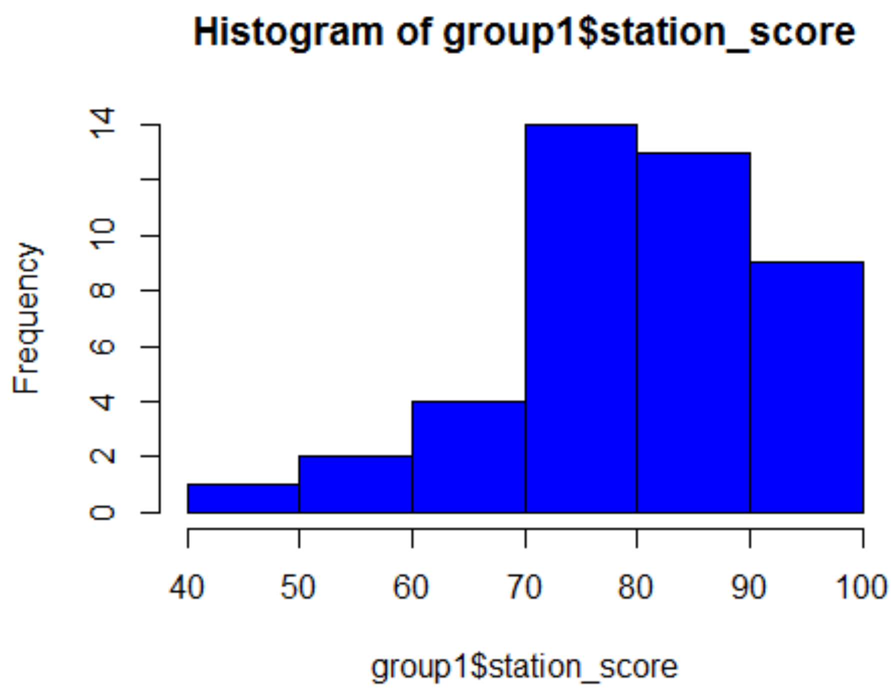
##Separating the dataset into the 5 groups of students

```
group1=OSCE[OSCE$location_index=="1", ]
group2=OSCE[OSCE$location_index=="2", ]
group3=OSCE[OSCE$location_index=="3", ]
```

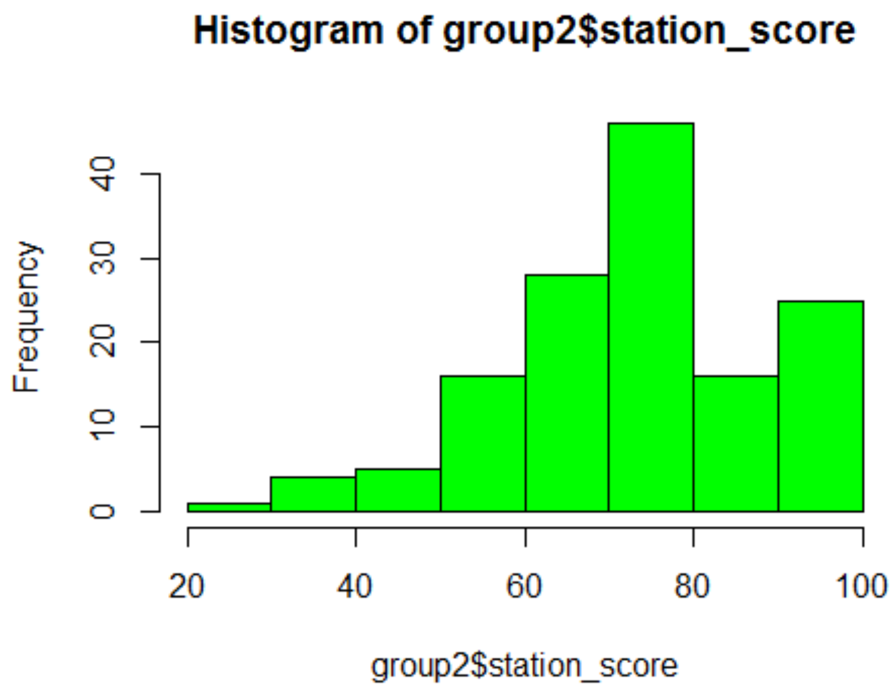
```
group4=OSCE[OSCE$location_index=="4", ]
group5=OSCE[OSCE$location_index=="5", ]
```

Plotting the station score of each of the 5 groups to examine the spread of station score by group

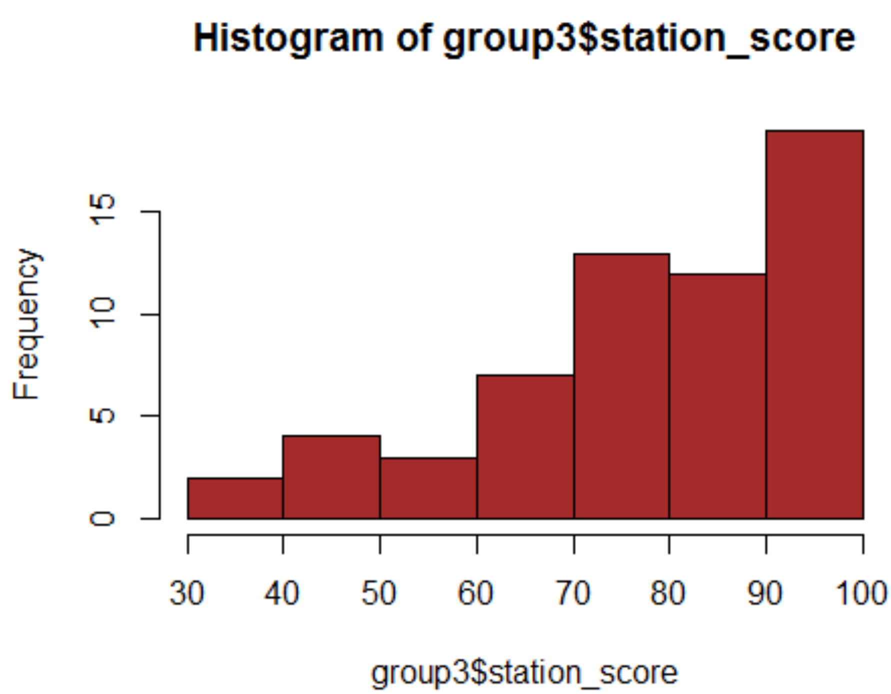
```
hist(group1$station_score, col="blue")
```



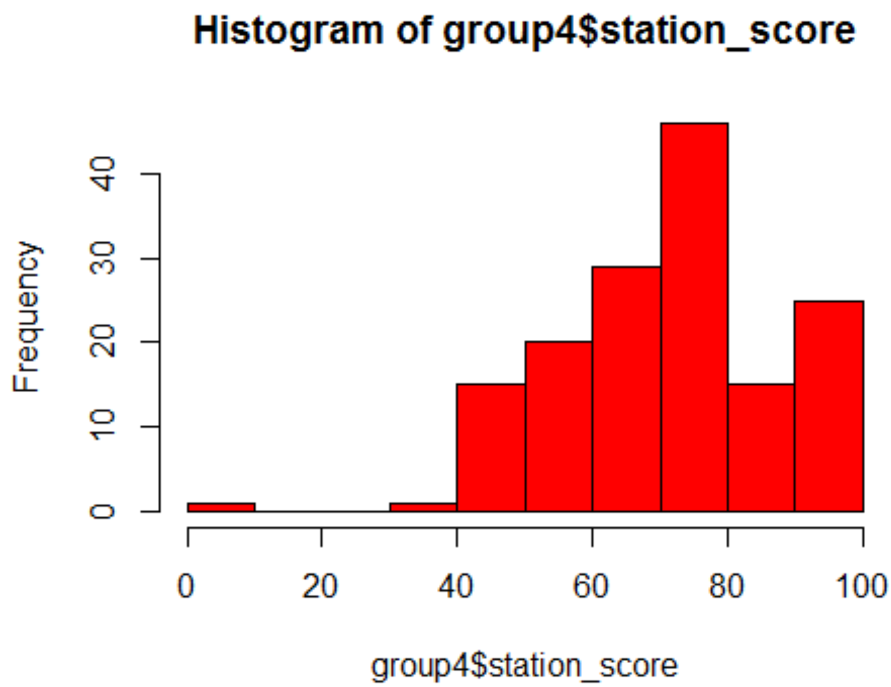

```
hist(group2$station_score, col="green")
```



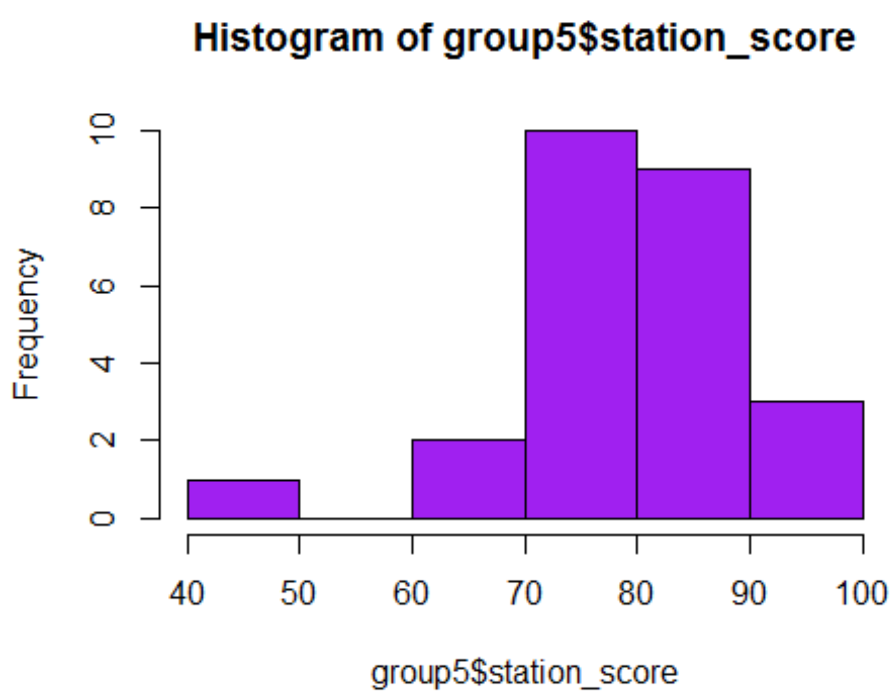

```r
hist(group3$station_score, col="brown")
```




```r
hist(group4$station_score,col="red")
```

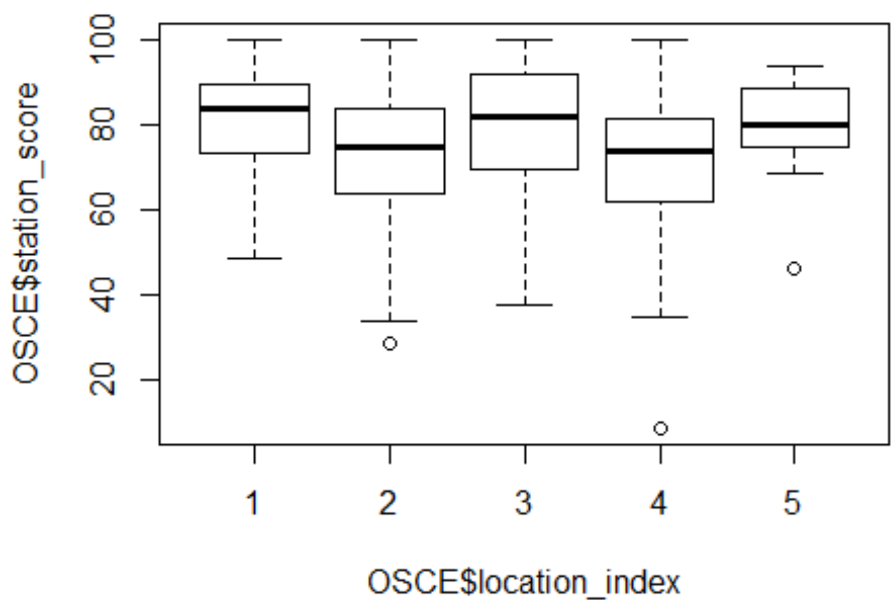



```r
hist(group5$station_score, col="purple")
```



Box plot of the station score by group

```
boxplot(OSCE$station_score~OSCE$location_index)
```



Outlier detection using the nearesk neighbor method

```
library(OutlierDetection)

## Warning: package 'OutlierDetection' was built under R version 3.6.2

nn(OSCE, k = 0.05 * nrow(OSCE), cutoff = 0.95, Method = "euclidean", rn
ames = FALSE, boottimes = 100)

## Warning in dist(data, diag = T, upper = T, method = Method): NAs int
roduced
## by coercion

## $`Outlier Observations`
##     date_of_hand_exam location location_index station_score
## 8            7/9/2019   Non-UK              1         48.75
## 64          7/10/2019       UK              2         43.75
## 97          7/10/2019       UK              2         28.75
## 110         7/10/2019       UK              2         41.25
## 133         7/10/2019       UK              2         41.25
## 154         7/10/2019       UK              2         33.75
## 160         7/10/2019       UK              2         35.00
## 178         7/10/2019       UK              2         38.75
## 181         7/10/2019       UK              2         40.00
## 197         7/10/2019   Non-UK              3         42.50
```

```
## 221         7/10/2019   Non-UK              3         38.75
## 244         7/10/2019   Non-UK              3         37.50
## 285         7/11/2019       UK              4         42.50
## 327         7/11/2019       UK              4         35.00
## 340         7/11/2019       UK              4          8.75
## 
## $`Location of Outlier`
##  [1]   8  64  97 110 133 154 160 178 181 197 221 244 285 327 340
## 
## $`Outlier Probability`
##  [1] 0.95 0.96 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.96 1.00 1.00 0.9
8 1.00
## [15] 1.00
## 
## $`3Dplot`

## Warning: `line.width` does not currently support multiple values.

## Warning: `line.width` does not currently support multiple values.
```

Outlier Detection using the Connectiveity Based Outlier Factor algorithm

COF computes the connectivity-based outlier factor for observations, being the comparison of chaining-distances between observation subject to outlier scoring and neighboring observations.The COF function is useful for outlier detection in clustering and other multidimensional domains.

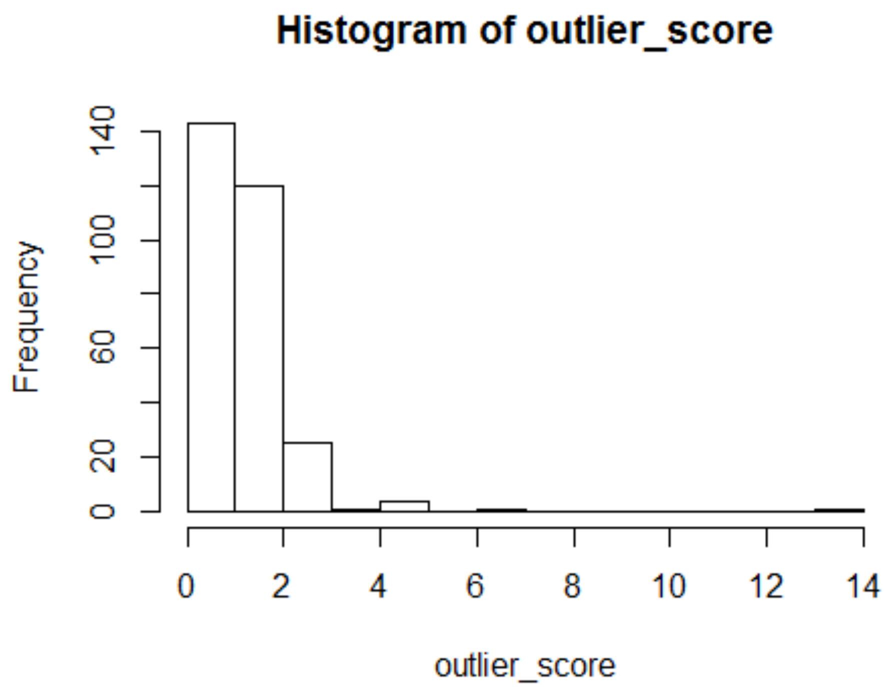
```
library(DDoutlier)

## Warning: package 'DDoutlier' was built under R version 3.6.2

outlier_score=COF(OSCE, k = 5)

## Warning in dist(dataset): NAs introduced by coercion

names(outlier_score) <- 1:nrow(OSCE)
sort(outlier_score, decreasing = TRUE)

##        13         15         29         66        187        210 
##       Inf        Inf        Inf        Inf        Inf        Inf 
##       401        404        398        374        208        236 
##       Inf        Inf 13.4515621  6.6666667  5.0000000  5.0000000 
##       320        340        205        352        367         44 
##  5.0000000  4.3430635  3.7500000  2.8409091  2.6785714  2.6041667 
##       397          1          7         12         36         38 
##  2.6041667  2.5000000  2.5000000  2.5000000  2.5000000  2.5000000 
##       189        190        200        229        239        263 
##  2.5000000  2.5000000  2.5000000  2.5000000  2.5000000  2.5000000 
##       266        267        272        292        372        145
```

```
##  2.5000000  2.5000000  2.5000000  2.5000000  2.5000000  2.4752475
##         41        256        211          5        193        216
##  2.4553571  2.3584906  2.3076923  2.0114943  1.9767442  1.9607843
##        252        222        232        355        360         97
##  1.9503546  1.8666667  1.7647059  1.7613636  1.7613636  1.7042586
##        188        230          9         14         22         25
##  1.6847826  1.6847826  1.6666667  1.6666667  1.6666667  1.6666667
##         33         34        117        138        158        198
##  1.6666667  1.6666667  1.6666667  1.6666667  1.6666667  1.6666667
##        201        215        220        223        237        242
##  1.6666667  1.6666667  1.6666667  1.6666667  1.6666667  1.6666667
##        400        403        405        406        195        226
##  1.6666667  1.6666667  1.6666667  1.6666667  1.6666667  1.6666667
##        228         64        132        399        409        206
##  1.6225166  1.5763006  1.5734266  1.5687150  1.5687150  1.5555556
##        408         20         59         71        183         85
##  1.5432099  1.5408805  1.5277778  1.5277778  1.5277778  1.5217391
##         92        199        224        250        389        102
##  1.5217391  1.5217391  1.5217391  1.5217391  1.5217391  1.4917127
##        349        392        207         60        165         31
##  1.4876033  1.4876033  1.4855072  1.4527027  1.4527027  1.4388489
##         39        407        412        415        362          3
##  1.4367816  1.4077670  1.4077670  1.4077670  1.3976589  1.3945578
##         28        255          4         40        416         16
##  1.3945578  1.3841808  1.3358779  1.3358779  1.3297136  1.3125000
##         17        240         61        197        257        330
##  1.3125000  1.3085938  1.2926829  1.2831955  1.2790698  1.2790698
##        336        154         21         35         37         67
##  1.2790698  1.2550248  1.2500000  1.2500000  1.2500000  1.2500000
##         81        177        410        418        122          8
##  1.2500000  1.2500000  1.2500000  1.2500000  1.2416107  1.2027491
##         99        143        166         72         76         27
##  1.2011173  1.2011173  1.1842105  1.1813187  1.1813187  1.1744966
##        414        244        327        160        417        130
##  1.1589404  1.1505012  1.1477140  1.1451432  1.1432110  1.1418685
##        219        285        279        385        196        231
##  1.1407767  1.1211243  1.1111111  1.1111111  1.0989011  1.0989011
##        150        184        234        254        383         50
##  1.0459184  1.0459184  1.0459184  1.0342217  1.0342217  1.0317460
##        106        129        170        191        178        221
##  1.0317460  1.0317460  1.0317460  1.0256410  1.0222083  1.0222083
##        283        334        354        381         11         18
##  1.0156250  1.0156250  1.0156250  1.0156250  1.0000000  1.0000000
##         30         32         42         73         98        111
##  1.0000000  1.0000000  1.0000000  1.0000000  1.0000000  1.0000000
##        114        116        137        148        162        164
##  1.0000000  1.0000000  1.0000000  1.0000000  1.0000000  1.0000000
##        186        209        213        217        264        268
##  1.0000000  1.0000000  1.0000000  1.0000000  1.0000000  1.0000000
##        273        302        308         96        134        152
```

```
##  1.0000000  1.0000000  1.0000000  0.9920635  0.9920635  0.9920635
##         48         58        153        281        307        368
##  0.9659091  0.9659091  0.9659091  0.9565217  0.9565217  0.9565217
##        275        276        375        260        300        357
##  0.9523810  0.9523810  0.9523810  0.9293836  0.9293836  0.9293836
##        278        395        181         65        174        110
##  0.8974359  0.8974359  0.8859091  0.8823529  0.8823529  0.8712521
##        133        411        420        421          2         10
##  0.8712521  0.8659231  0.8659231  0.8659231  0.8333333  0.8333333
##         26         43         83        103        126        147
##  0.8333333  0.8333333  0.8333333  0.8333333  0.8333333  0.8333333
##        280        309        348        373        185        203
##  0.8333333  0.8333333  0.8333333  0.8333333  0.8130081  0.8130081
##        227        344        371        394        212        235
##  0.8130081  0.7911392  0.7911392  0.7911392  0.7692308  0.7692308
##        243         62        182        306        311        313
##  0.7692308  0.7575758  0.7575758  0.7500000  0.7500000  0.7500000
##        318        261        329        376        277        325
##  0.7500000  0.7396450  0.7396450  0.7396450  0.7352941  0.7352941
##        351        364         80         84        146        172
##  0.7352941  0.7352941  0.7303371  0.7303371  0.7303371  0.7303371
##          6         19         23         24         69         82
##  0.7142857  0.7142857  0.7142857  0.7142857  0.7142857  0.7142857
##        119        175        321        328        332        346
##  0.7142857  0.7142857  0.7142857  0.7142857  0.7142857  0.7142857
##        402        413        419        194        204        233
##  0.7142857  0.7142857  0.7142857  0.6976744  0.6976744  0.6976744
##        238         57         77         86         87         88
##  0.6976744  0.6250000  0.6250000  0.6250000  0.6250000  0.6250000
##         89         90        109        113        155        246
##  0.6250000  0.6250000  0.6250000  0.6250000  0.6250000  0.5555556
##        253        269        270        293        316        317
##  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556
##        339        343        356        378        382        384
##  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556
##        388        118        131        139        140        157
##  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556
##        294        299        319        359        361        288
##  0.5555556  0.5555556  0.5555556  0.5555556  0.5555556  0.5357143
##        310        377        393
##  0.5357143  0.5357143  0.5357143
```

Inspect the distribution of outlier scores

```r
hist(outlier_score)
```




```
OSCE=cbind(OSCE,outlier_score)
```

This code computes the average outlier score across the group and identify the group with highest dispersion (noise) level

```
agg <- aggregate(outlier_score~ location_index, data = OSCE, FUN= "mean
")
agg

##   location_index outlier_score
## 1              1           Inf
## 2              2           Inf
## 3              3           Inf
## 4              4       1.18266
## 5              5           Inf
```

Notes: All methods which include visualization and classification indicates that the data does not account for noise in group 3. The analysis only points at noise in group 4 and 5. Though group 3 has the highest variance. but this cannot be established beyond that.