Clustering

Code

This notebook tests 3 clustering algorithms: kMeans, Hierarchical and Model-Based.

I am using a dataset on patient survival. (https://www.kaggle.com/datasets/mitishaagarwal/patient)

As there is a lot of data in the dataset, I am going to limit the amount of data used for our clustering algorithms by only using the first 2000 rows and specific relevant columns. I am also making sure I get rid of the NA and invalid fields.

Hide

```
data <- read.csv("dataset.csv", header = TRUE, stringsAsFactors = TRUE) # read in csv
patientData <- data[c(1:2000), c(4, 6, 27, 29, 34, 37, 41, 42, 43, 49, 54, 61, 69, 70, 72, 73, 7
5, 76, 78, 81, 85)] # select relavent columns
patientData <- patientData[complete.cases(patientData), ] # remove all rows with NA in any colum
n
patientData <- patientData[patientData$apache_4a_hospital_death_prob >= 0,] #remove invalid valu
es
patientData <- patientData[patientData$apache_4a_icu_death_prob >= 0,] #remove invalid values
patientData <- scale(patientData)
head(patientData)
```

```
        age elective_surgery resprate_apache ventilated_apache d1_heartrate_max d1_mbp_min
1  0.3296004       -0.5609668       0.3060610        -0.6754347        0.6469094 -1.5208080
2  0.8935729       -0.5609668       0.1106804         1.4795236        0.6021883 -2.0689944
4  1.1442273        1.7814272      -1.7779994         1.4795236        0.5127459  1.0830771
6  0.2669368       -0.5609668       0.2409341        -0.6754347        0.3785825  0.8089840
7 -0.2343721       -0.5609668       1.4132181         1.4795236        0.3338613  1.9738799
8  0.4549276       -0.5609668      -0.2149541         1.4795236        0.6021883 -0.5614819
  d1_resprate_min d1_spo2_max d1_spo2_min d1_temp_min h1_heartrate_max h1_resprate_min d1_glucos
e_min
1    -0.404197416   0.5714806  -1.6378217   1.2451357        1.1258059       0.2118341     -0.12
84017
2     0.005982957   0.5714806  -2.0529092  -1.4705956        0.9071942       1.8612974      0.41
32487
4    -1.019467976   0.5714806   0.5413879  -1.8585572        0.2950813      -0.9427903     -0.72
70678
6    -0.404197416  -2.4226986   0.1263004   0.4692125       -0.4481985      -0.7778439      0.32
77250
7     0.826343703   0.5714806  -0.2887872  -1.5999161       -0.6230879       0.2118341      0.44
17566
8     0.005982957   0.5714806   0.2300723   0.4692125        1.0820836       1.5314047      0.44
17566
  d1_potassium_max apache_4a_hospital_death_prob apache_4a_icu_death_prob  cirrhosis diabetes_me
llitus
1      -0.35363526                    0.07027375              -0.04754737 -0.1489172          1.
646513
2      -0.03666452                    2.58674774               1.91286724 -0.1489172          1.
646513
4       1.23121840                   -0.33780311              -0.21091525 -0.1489172         -0.
606932
6      -0.51212062                   -0.26979030              -0.29259920 -0.1489172          1.
646513
7       1.23121840                    0.07027375              -0.04754737 -0.1489172          1.
646513
8       2.49910133                    0.13828656               0.03413657 -0.1489172         -0.
606932
  immunosuppression solid_tumor_with_metastasis hospital_death
1        -0.2609018                  -0.1930605     -0.2518908
2        -0.2609018                  -0.1930605     -0.2518908
4        -0.2609018                  -0.1930605     -0.2518908
6        -0.2609018                  -0.1930605     -0.2518908
7        -0.2609018                  -0.1930605     -0.2518908
8         3.8302599                  -0.1930605     -0.2518908
```

kMeans Clustering

kMeans clustering is a centroid-based algorithm that uses Euclidean distances between observations/points to assign to its nearest centroid.

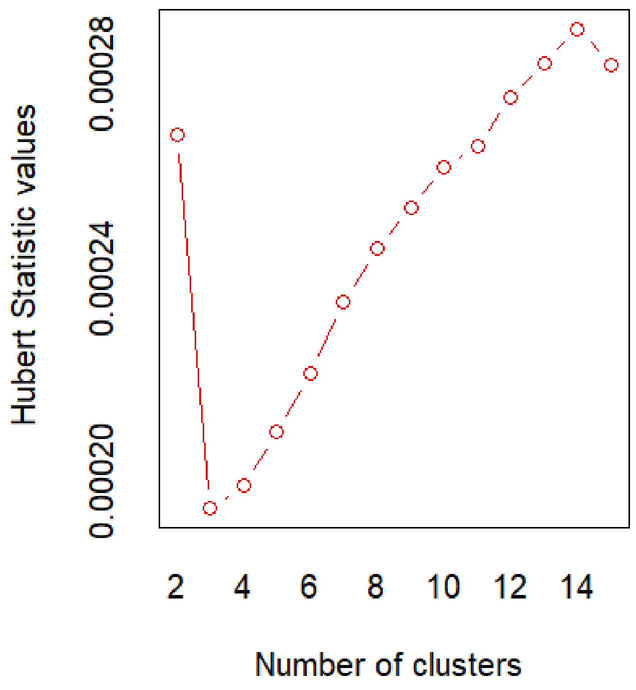
First, I should determine the best number of clusters. I can use the NbClust() function to find the best k value.

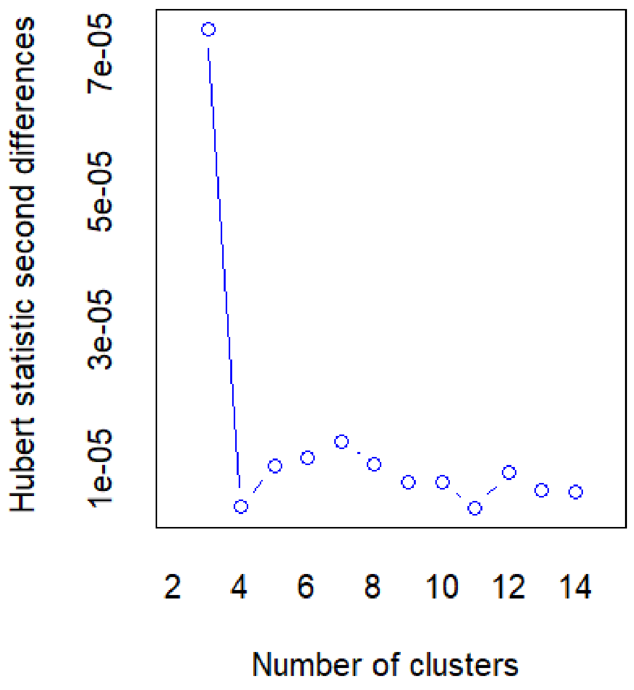
Hide

```
library(NbClust)
set.seed(1234)
nc <- NbClust(patientData, min.nc=2, max.nc=15, method ="kmeans")
```

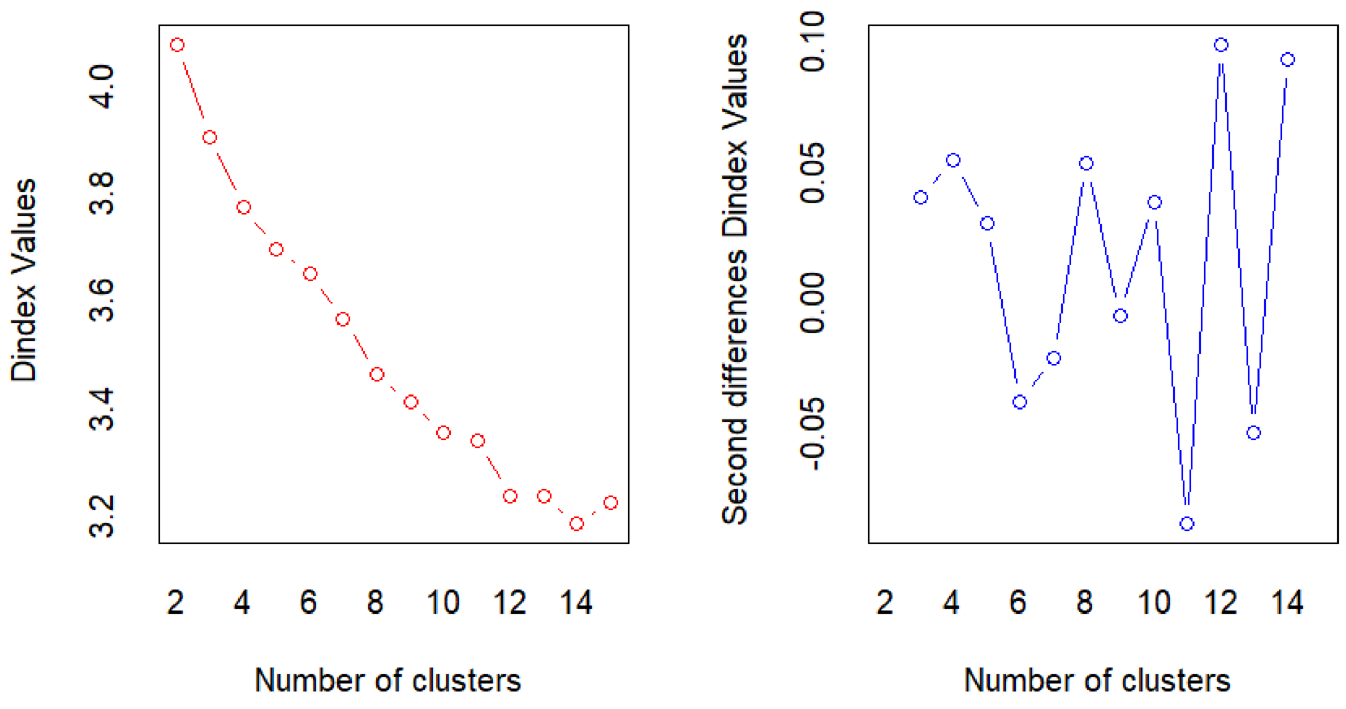
```
Registered S3 methods overwritten by 'htmltools':
  method               from
  print.html           tools:rstudio
  print.shiny.tag      tools:rstudio
  print.shiny.tag.list tools:rstudio
Warning: did not converge in 10 iterationsWarning: did not converge in 10 iterationsWarning: did
not converge in 10 iterationsWarning: did not converge in 10 iterationsWarning: did not converge
in 10 iterationsWarning: did not converge in 10 iterations
```

```
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hub
ert
                index second differences plot.

```





```
*** : The D index is a graphical method of determining the number of clusters. 
                In the plot of D index, we seek a significant knee (the significant peak in Dind
ex
                second differences plot) that corresponds to a significant increase of the value
of
                the measure. 
 
******************************************************************* 
* Among all indices:                                                
* 10 proposed 2 as the best number of clusters 
* 3 proposed 3 as the best number of clusters 
* 1 proposed 4 as the best number of clusters 
* 2 proposed 7 as the best number of clusters 
* 1 proposed 12 as the best number of clusters 
* 6 proposed 14 as the best number of clusters 
* 1 proposed 15 as the best number of clusters 

                   ***** Conclusion *****                            
 
* According to the majority rule, the best number of clusters is  2 
 
 
******************************************************************* 
```



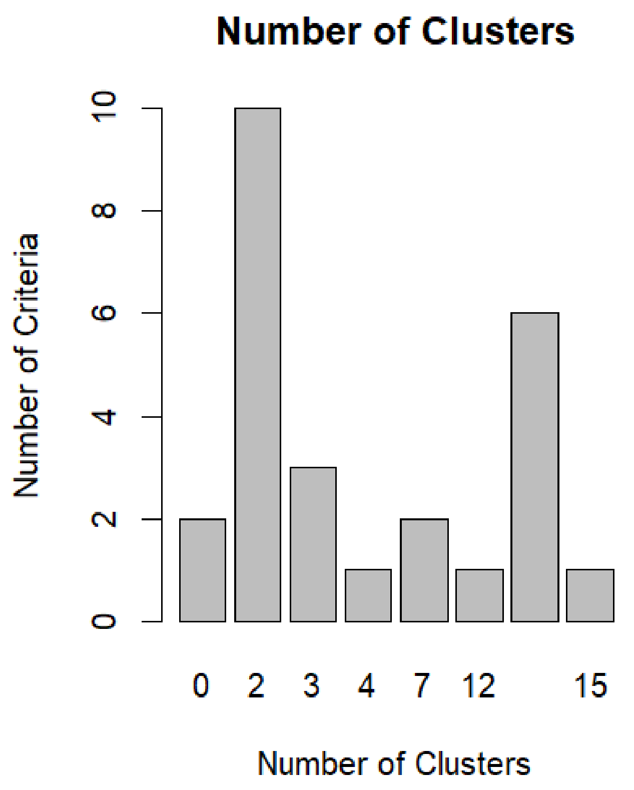
Hide

```
table(nc$Best.n[1,])
```

```

 0  2  3  4  7 12 14 15
 2 10  3  1  2  1  6  1
```

Hide

```
barplot(table(nc$Best.n[1,]),
        xlab="Number of Clusters", ylab="Number of Criteria",
        main="Number of Clusters")
```



The NbClust() function showed that 2 clusters are optimal. Now I will try clustering with kmeans().

Hide

```
set.seed(1234)
patientCluster <- kmeans(patientData, 2, nstart=24)
patientCluster
```

```
K-means clustering with 2 clusters of sizes 97, 1377

Cluster means:
         age elective_surgery resprate_apache ventilated_apache d1_heartrate_max  d1_mbp_min
1  0.1500078      -0.51267004      0.29934691        1.27957903       0.81933943 -0.92034616
2 -0.0105670       0.03611401     -0.02108689       -0.09013738      -0.05771672  0.06483194
  d1_resprate_min d1_spo2_max d1_spo2_min d1_temp_min h1_heartrate_max h1_resprate_min d1_glucos
e_min
1     -0.14413460  0.32453799 -1.17566235 -1.36496605        0.5799527       0.3563747     -0.19
98184
2      0.01015327 -0.02286143  0.08281717  0.09615229       -0.0408536      -0.0251041      0.01
40758
  d1_potassium_max apache_4a_hospital_death_prob apache_4a_icu_death_prob    cirrhosis diabetes_
mellitus
1       0.53192215                      2.994123                3.0050742  0.133948758
0.2293982
2      -0.03747019                     -0.210915               -0.2116864 -0.009435751        -
0.0161595
  immunosuppression solid_tumor_with_metastasis hospital_death
1        0.16086748                  0.36047231      2.3579059
2       -0.01133199                 -0.02539275     -0.1660979

Clustering vector:
   1    2    4    6    7    8   10   11   14   15   16   18   19   20   23   24   25   26   27
28   30
   2    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    1    2
2    2
  31   32   34   35   36   37   38   39   40   41   43   44   45   46   47   48   49   50   53
54   56
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    1    1    2    2    2
2    2
  58   59   61   62   63   65   66   67   69   71   72   74   75   78   79   80   82   83   85
86   87
   2    2    2    2    2    1    2    2    2    2    2    1    2    2    2    2    2    2    1
2    2
  88   90   91   93   94   95   96   97   98   99  100  102  103  104  105  106  107  108  109
110  111
   2    2    1    2    2    2    2    2    1    2    2    2    2    2    2    2    2    2    2
2    2
 112  113  114  116  117  119  120  121  122  123  124  125  126  127  128  129  130  131  132
133  135
   2    1    2    2    1    2    2    2    2    2    2    2    2    2    2    1    1    2    2
2    2
 136  138  139  140  141  143  147  148  149  151  152  153  156  157  158  159  160  161  162
163  166
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 167  168  169  170  171  172  173  174  176  177  178  179  180  182  183  186  187  188  189
190  191
   2    2    2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2    2
2    2
 192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210
```

```
211  212
   2    2    2    2    2    2    2    2    2    1    2    1    2    2    2    2    2    2    2
2    2
 213  214  215  216  217  218  220  222  223  225  226  227  229  230  232  234  235  236  237
238  239
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 240  241  242  244  245  246  247  250  251  253  254  256  258  259  261  262  263  265  268
269  270
   2    2    2    2    2    2    2    1    2    2    2    2    2    2    2    2    2    2    2
2    2
 272  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  291  292
294  298
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    1
2    2
 299  300  302  304  305  306  308  309  312  313  314  315  316  317  318  321  322  323  325
326  328
   1    2    2    2    2    2    1    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 330  332  333  335  337  338  339  340  342  343  345  346  348  350  352  353  354  355  356
359  362
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    1    2
2    2
 363  364  366  367  368  369  372  376  377  379  380  381  382  383  384  388  389  391  392
393  394
   2    2    2    2    2    2    2    2    2    1    2    1    2    2    2    2    2    2    2
2    2
 395  396  397  399  400  402  405  406  407  409  410  411  412  413  414  415  416  417  418
419  421
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 422  423  425  426  427  428  429  431  432  433  434  435  436  437  438  439  440  441  442
443  444
   2    2    2    2    2    2    1    2    2    2    1    2    2    2    2    2    2    2    2
2    2
 445  446  447  448  449  450  451  452  453  454  455  456  458  460  461  462  463  466  467
468  469
   1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 471  472  473  475  476  478  479  480  481  484  486  487  488  490  491  492  493  495  496
497  498
   1    2    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 499  500  501  503  504  505  508  511  512  514  515  517  518  520  521  523  524  525  526
528  529
   2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2    2    2    1
2    2
 530  531  532  533  534  535  536  538  540  541  542  544  545  547  548  549  551  552  553
555  556
   2    2    2    2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2
2    2
 557  558  559  561  562  564  565  567  568  569  573  574  575  576  578  579  580  581  582
```

```
583  585
   2    2    2    2    2    2    2    2    1    2    1    2    2    2    2    2    2    2    2
2    2
 587  588  590  591  592  593  594  595  596  599  600  604  606  607  609  611  613  614  616
617  618
   1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 619  620  621  622  623  624  625  626  628  629  632  633  636  637  638  639  640  641  642
643  644
   2    1    2    2    2    2    2    2    2    2    2    2    2    1    1    2    2    2    2
2    2
 645  647  649  651  652  654  655  656  657  658  659  660  663  664  666  668  669  670  671
673  675
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 676  678  679  680  681  682  683  684  685  687  691  692  694  695  696  698  699  700  701
702  703
   2    2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2    2    2
2    2
 704  705  706  707  709  710  711  712  714  715  716  719  724  725  727  728  729  730  731
732  733
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 734  737  738  739  740  743  744  745  746  747  748  749  750  752  753  754  755  756  757
758  759
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    1
 760  761  762  763  764  766  767  768  770  771  772  774  775  776  777  778  779  781  782
783  784
   2    2    2    2    2    2    2    1    2    1    2    1    2    2    2    2    2    2    2
2    2
 785  786  787  788  789  790  791  792  795  798  800  801  802  803  805  806  807  808  809
810  811
   2    2    2    2    1    2    2    2    2    2    2    1    2    2    2    2    2    2    2
2    2
 812  813  815  817  818  819  820  822  823  824  825  826  827  830  832  833  835  836  837
838  839
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 840  842  843  845  846  847  848  849  851  852  854  855  856  857  858  860  861  862  863
866  868
   2    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
 869  870  872  876  877  879  880  881  882  886  888  889  890  891  892  894  895  896  897
898  900
   2    2    2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2    2
2    2
 901  902  905  906  907  908  909  910  912  913  914  915  918  920  921  922  923  925  926
927  929
   2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2    2    2    2
2    2
 931  934  935  936  937  938  939  940  942  943  944  946  947  950  951  958  960  961  964
```

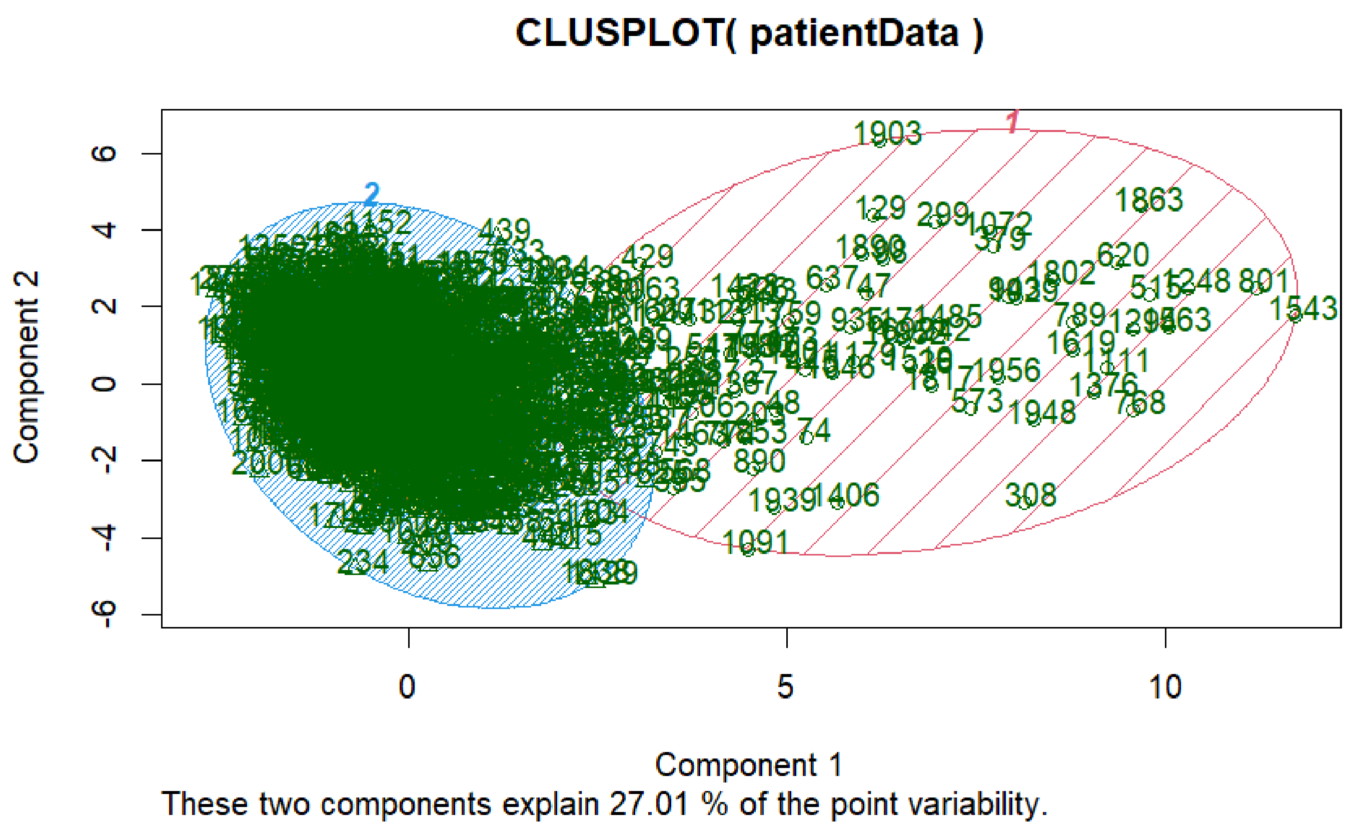
```
965  972
   2    2    1    2    2    2    2    2    1    1    2    2    2    2    1    2    2    2    2
2    2
 973  976  977  978  980  981  982  984  985  987  988  989  990  991  992  993  994  997  998 1
001 1002
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
1    2
1005 1006 1007 1010 1011 1012 1013 1015 1016 1018 1020 1022 1023 1024 1025 1026 1027 1028 1030 1
031 1033
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
1034 1035 1036 1037 1039 1040 1041 1045 1046 1047 1049 1051 1053 1054 1058 1060 1061 1063 1064 1
065 1066
   2    2    2    2    2    2    2    2    1    2    2    2    2    2    2    2    2    1    2
2    2
1068 1069 1070 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1083 1085 1086 1087 1088 1089 1
090 1091
   2    2    2    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    1
1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1103 1104 1106 1107 1108 1109 1110 1111 1112 1
113 1114
   2    2    2    1    2    2    2    2    1    2    2    2    2    2    2    2    2    1    2
2    2
1115 1116 1118 1119 1121 1122 1123 1124 1125 1126 1128 1129 1130 1132 1133 1134 1135 1136 1137 1
138 1141
   2    2    2    2    2    2    2    2    2    2    2    2    2    1    2    2    2    2    2
2    2
1142 1144 1145 1146 1147 1149 1151 1152 1153 1154 1155 1156 1158 1159 1160 1161 1163 1164 1165 1
166 1167
   2    2    2    2    2    1    2    2    2    2    2    2    2    2    2    2    2    2    1
2    2
1168 1169 1170 1171 1173 1174 1175 1176 1179 1180 1181 1182 1183 1184 1186 1187 1188 1190 1191 1
192 1194
   2    2    2    2    2    2    2    2    1    2    2    2    2    2    2    2    2    2    2
2    2
1195 1197 1198 1199 1201 1202 1204 1205 1206 1211 1213 1214 1218 1219 1220 1222 1223 1224 1226 1
227 1229
   2    2    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
1230 1231 1232 1234 1235 1236 1237 1238 1240 1241 1242 1244 1245 1246 1248 1249 1250 1251 1252 1
253 1257
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    1    2    2    2    2
2    2
1258 1259 1260 1262 1265 1266 1268 1271 1273 1274 1276 1277 1278 1279 1280 1281 1282 1283 1284 1
285 1286
   2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
2    2
1287 1288 1289 1290 1292 1293 1294 1296 1297 1298 1299 1300 1301 1302 1304 1305 1306 1307 1308 1
309 1310
   2    2    2    2    2    2    1    2    2    2    1    2    2    2    2    2    2    2    2
2    2
1311 1312 1313 1314 1315 1317 1318 1320 1322 1324 1325 1326 1329 1330 1331 1333 1334 1335 1336 1
```

```
339 1340
   2    2    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2 
2    2 
1342 1343 1344 1345 1346 1347 1349 1350 1351 1352 1353 1355 1357 
   2    2    2    2    2    2    2    2    2    2    2    2    2 
 [ reached getOption("max.print") -- omitted 474 entries ]

Within cluster sum of squares by cluster:
[1]  4039.627 23625.791
 (between_SS / total_SS =  10.6 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"   
[7] "size"         "iter"         "ifault"      
```

Now I will plot the kmeans clustering.

Hide

```
library(cluster)
clusplot(patientData, patientCluster$cluster, color=TRUE, shade=TRUE,
         labels=2, lines=0)
```



Hierarchical Clustering

Hierarchical Clustering uses distance to group observations/points into clusters organized in a hierarchy.

I am using the pvclust package. The pvclust() function performs hierarchical clustering based on p-values, values from 0 to 1 that shows how strong the cluster is supported by the data. I will use Euclidean distance and Ward’s method to generate clusters.The pvrect() function then adds rectangles in the dendogram to show the clusters.

Hide

```
library(pvclust)
```

```
Warning: package ‘pvclust’ was built under R version 4.2.3
```

Hide

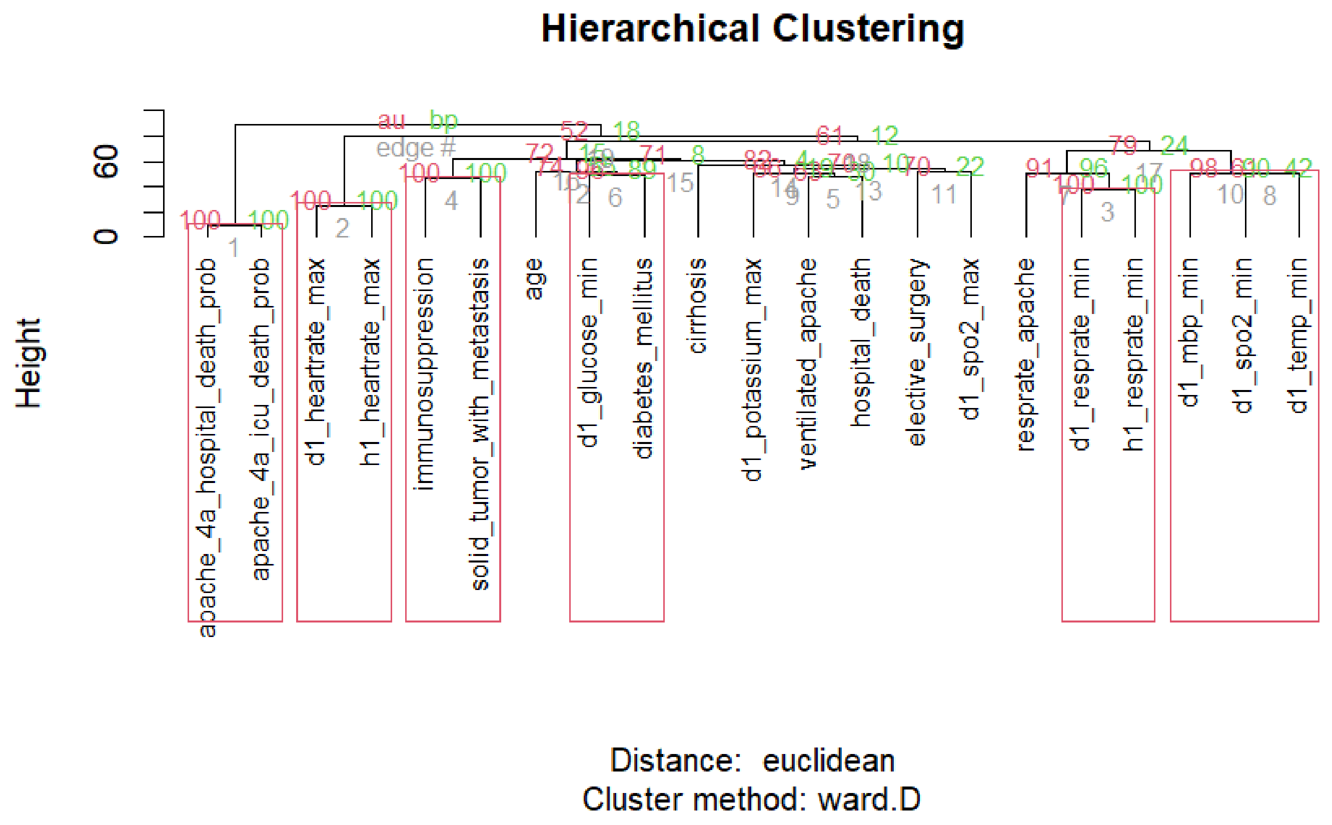
```
fit <- pvclust(patientData, method.hclust = "ward", method.dist="euclidean")
```

```
The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.6)... Done.
Bootstrap (r = 0.7)... Done.
Bootstrap (r = 0.8)... Done.
Bootstrap (r = 0.9)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.1)... Done.
Bootstrap (r = 1.2)... Done.
Bootstrap (r = 1.3)... Done.
Bootstrap (r = 1.4)... Done.
```

Hide

```
plot(fit, hang=-1, cex=.8,
     main="Hierarchical Clustering")
pvrect(fit, alpha=.95)
```



Model-Based Clustering

Model-Based Clustering creates multiple data models and tries to identify the most likely clustering based of a maximum likelihood estimation.

I am using the mclust package. The Mclust() function provides model-based clustering based on parameterized Gaussian mixture models. The model with the largest Bayesian Information Criterion (BIC) is picked as the most optimal.

Hide

```
library(mclust)
```

```
Warning: package ‘mclust’ was built under R version 4.2.3              __          __
   ____ ___  _____/ /_  _______/ /_
  / __ `__ \/ ___/ / / / / ___/ __/
 / / / / / / /__/ / /_/ (__  ) /_
/_/ /_/ /_/\___/_/\__,_/____/\__/   version 6.0.0
Type 'citation("mclust")' for citing this R package in publications.
```

Hide

```
fit <- Mclust(patientData)
```

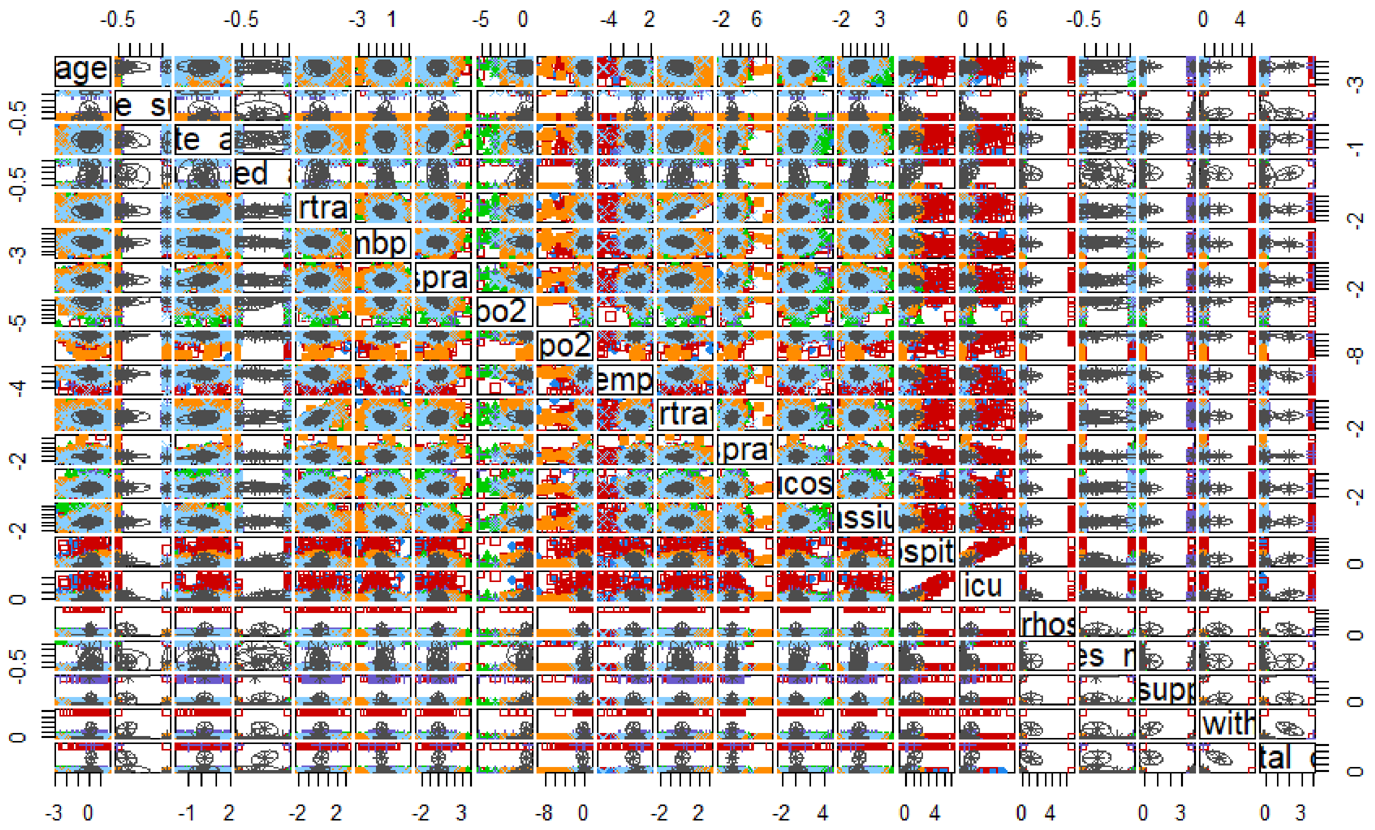
```
fitting ...

  |                                                                            
  |                                                                      |   0%
  |                                                                            
  |=                                                                     |   1%
  |                                                                            
  |=                                                                     |   2%
  |                                                                            
  |==                                                                    |   2%
  |                                                                            
  |===                                                                   |   3%
  |                                                                            
  |====                                                                  |   4%
  |                                                                            
  |====                                                                  |   5%
  |                                                                            
  |=====                                                                 |   6%
  |                                                                            
  |======                                                                |   6%
  |                                                                            
  |=======                                                               |   7%
  |                                                                            
  |=======                                                               |   8%
  |                                                                            
  |========                                                              |   9%
  |                                                                            
  |=========                                                             |   9%
  |                                                                            
  |==========                                                            |  10%
  |                                                                            
  |==========                                                            |  11%
  |                                                                            
  |===========                                                           |  12%
  |                                                                            
  |============
|  13%
  |
  |=============
|  13%
  |
  |=============
|  14%
  |
  |==============
|  15%
  |
  |===============
|  16%
  |
  |================
|  17%
  |
  |=================
|  18%
  |
  |==================
|  19%
  |
  |===================
|  20%
  |
  |====================
|  21%
  |
  |=====================
|  22%
  |
  |======================
|  23%
  |
  |======================
|  24%
  |
  |=======================
|  24%
  |
  |========================
|  25%
  |
  |=========================
|  26%
  |
  |=========================
|  27%
  |
  |==========================
|  28%
```

```
  |
  |===========================
|  28%
  |
  |============================
|  29%
  |
  |============================
|  30%
  |
  |=============================
|  31%
  |
  |==============================
|  31%
  |
  |===============================
|  32%
  |
  |===============================
|  33%
  |
  |================================
|  34%
  |
  |=================================
|  35%
  |
  |==================================
|  35%
  |
  |==================================
|  36%
  |
  |===================================
|  37%
  |
  |====================================
|  38%
  |
  |=====================================
|  39%
  |
  |======================================
|  40%
  |
  |=======================================
|  41%
  |
  |========================================
|  42%
  |
  |========================================
|  43%
  |
  |=========================================
|  43%
  |
  |==========================================
|  44%
  |
  |===========================================
|  45%
  |
  |===========================================
|  46%
  |
  |============================================
|  46%
  |
  |=============================================
|  47%
  |
  |==============================================
|  48%
  |
  |==============================================
|  49%
  |
  |===============================================
|  50%
  |
  |================================================
|  50%
  |
  |=================================================
|  51%
  |
  |=================================================
|  52%
  |
  |==================================================
|  53%
  |
  |===================================================
|  54%
  |
  |====================================================
|  54%
  |
  |====================================================
|  55%
  |
  |=====================================================
|   56%
  |
  |======================================================
|   57%
  |
  |=======================================================
|   57%
  |
  |=======================================================
|   58%
  |
  |========================================================
|   59%
  |
  |=========================================================
|   60%
  |
  |==========================================================
|   61%
  |
  |===========================================================
|   62%
  |
  |============================================================
|   63%
  |
  |=============================================================
|   64%
  |
  |=============================================================
|   65%
  |
  |==============================================================
|   65%
  |
  |===============================================================
|   66%
  |
  |================================================================
|   67%
  |
  |================================================================
|   68%
  |
  |=================================================================
|   69%
  |
  |==================================================================
|   69%
  |
  |===================================================================
|   70%
```

```
  |
  |===================================================================
|  71%
  |
  |====================================================================
|  72%
  |
  |=====================================================================
|  72%
  |
  |======================================================================
|  73%
  |
  |======================================================================
|  74%
  |
  |=======================================================================
|  75%
  |
  |========================================================================
|  76%
  |
  |=========================================================================
|  76%
  |
  |=========================================================================
|  77%
  |
  |==========================================================================
|  78%
  |
  |===========================================================================
|  79%
  |
  |============================================================================
|  80%
  |
  |=============================================================================
|  81%
  |
  |==============================================================================
|  82%
  |
  |===============================================================================
|  83%
  |
  |================================================================================
|  84%
  |
  |=================================================================================
|  85%
  |
  |==================================================================================
|  86%
  |
  |==================================================================================
|  87%
  |
  |===================================================================================
|  87%
  |
  |====================================================================================
|  88%
  |
  |=====================================================================================
|  89%
  |
  |=====================================================================================
|  90%
  |
  |======================================================================================
|  91%
  |
  |=======================================================================================
|  91%
  |
  |========================================================================================
|  92%
  |
  |========================================================================================
|  93%
  |
  |=========================================================================================
|  94%
  |
  |==========================================================================================
|  94%
  |
  |===========================================================================================
|  95%
  |
  |===========================================================================================
|  96%
  |
  |============================================================================================
|  97%
  |
  |=============================================================================================
|  98%
  |
  |=============================================================================================
= |  98%
  |
  |=============================================================================================
= |  99%
  |
  |==========================================================================================
==| 100%
```

Hide

```
plot(fit, what = "classification") # plot results
```



Hide

```
summary(fit) # display the best model
```

```
----------------------------------------------------
Gaussian finite mixture model fitted by EM algorithm
----------------------------------------------------

Mclust EEV (ellipsoidal, equal volume and shape) model with 6 components:
```

	log-likelihood <dbl>	n <int>	df <dbl>	BIC <dbl>	ICL <dbl>
	-27106.63	1474	1412	-64514.85	-64608.75

1 row

```
Clustering table:
  1   2   3   4   5   6 
151 162 291  68 496 306 
```

Analysis

The kMeans, hierarchical and model-based clustering had varied results:

- KMeans Clustering: 2 Clusters
- Hierarchical Clustering: 6 Clusters
- Model-based Clustering: 6 Clusters

I think model-based clustering showed the best results. KMeans seemed to group most of the points in one cluster and the outliers in another cluster. Hierarchical clustering and model-based clustering provided the same result and had more relevant and useful clusters. However, there is no evidence of a hierarchical structure in our data so I think model-based clustering is more relevant. Model-based clustering also seems the most thorough.