Clustering

Bishal Neupane, Saugat Gyawali, Spencer Gray, Michael Stinnett

Data Set Link (https://archive.ics.uci.edu/ml/datasets/Dry+Bean+Dataset)

I used data describing 7 types of dry beans. Originally this data was intended to be used for classification, but I omitted the target bean class to enable clustering.

Set up Data

```
rm(list=ls())
```

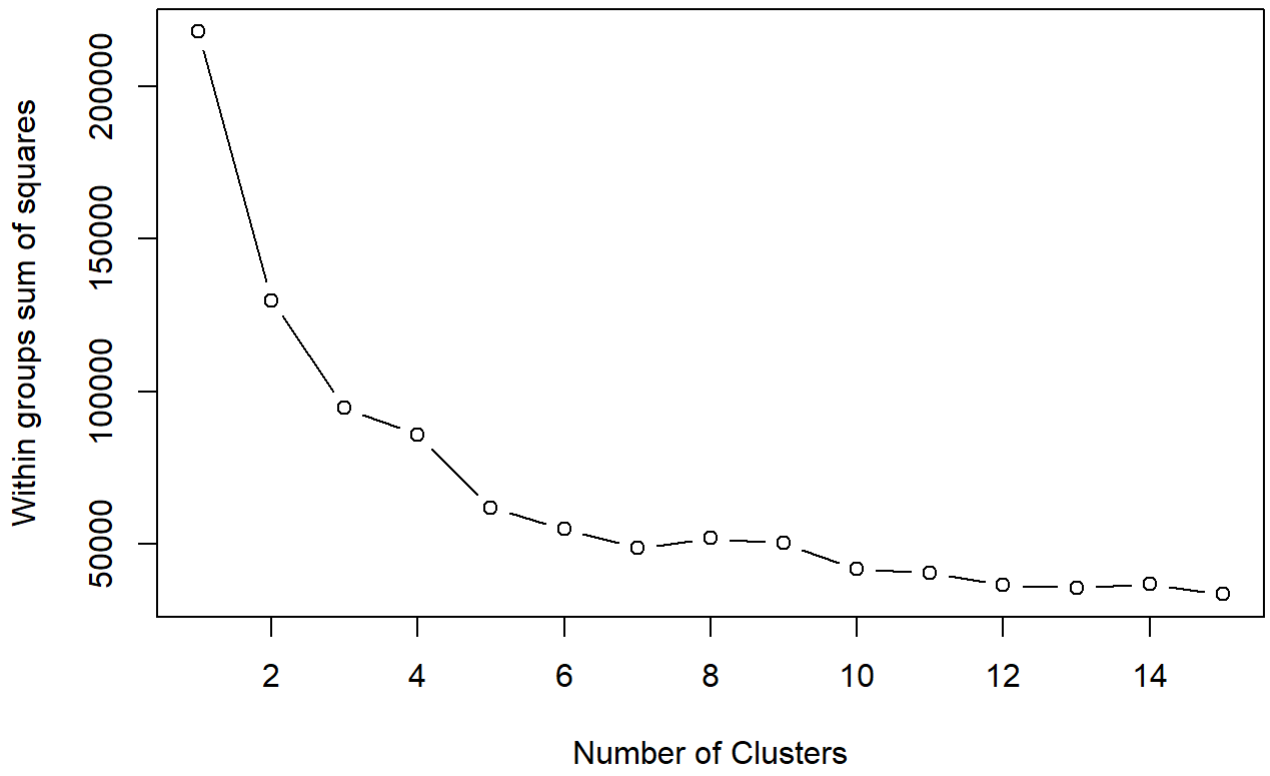
```
set.seed(1234)
data <- read.csv("Dry_Bean_Dataset.csv")
data <- na.omit(data)
```

```
data.scaled <- data[, -c(17)]
data.scaled <- scale(data.scaled)
```

Kmeans

Graphing a few groups sum of squares with different cluster size to see if 7 clusters will really work out.

```
wss <- (nrow(data.scaled)-1)*sum(apply(data.scaled,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(data.scaled,
   centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
  ylab="Within groups sum of squares")
```



Performing kmeans operation and putting the cluster profiles back into the data

```
fit <- kmeans(data.scaled, 7)
aggregate(data.scaled,by=list(fit$cluster),FUN=mean)
```

Grou… <int>	Area <dbl>	Perimeter <dbl>	MajorAxisLength <dbl>	MinorAxisLength <dbl>	AspectRation <dbl>	Eccent
1	-0.32903010	-0.3619829	-0.3541018	-0.25560484	-0.25238297	-0.00772
2	-0.77058050	-0.9731079	-0.9464942	-0.89101907	-0.43432667	-0.20666
3	-0.46778917	-0.6257123	-0.8395457	-0.01951913	-1.41908510	-1.88675
4	4.11098583	3.4109128	3.1890420	3.82874389	0.01084531	0.21483
5	0.03344718	0.3224118	0.6405849	-0.40215522	1.84805407	1.29378
6	-0.30537665	-0.2687848	-0.2260681	-0.34456047	0.11939338	0.33521
7	0.71393517	0.9612035	0.8849435	0.82087086	0.30962112	0.44550

7 rows | 1-7 of 17 columns

```
data <- data.frame(data, fit$cluster)
```

```
options(max.print = 250)
fit
```

```
## K-means clustering with 7 clusters of sizes 2137, 2659, 1992, 521, 1871, 1652, 2779
##
## Cluster means:
##          Area  Perimeter MajorAxisLength MinorAxisLength AspectRation
## 1 -0.32903010 -0.3619829      -0.3541018     -0.25560484  -0.25238297
## 2 -0.77058050 -0.9731079      -0.9464942     -0.89101907  -0.43432667
## 3 -0.46778917 -0.6257123      -0.8395457     -0.01951913  -1.41908510
## 4  4.11098583  3.4109128       3.1890420      3.82874389   0.01084531
## 5  0.03344718  0.3224118       0.6405849     -0.40215522   1.84805407
## 6 -0.30537665 -0.2687848      -0.2260681     -0.34456047   0.11939338
## 7  0.71393517  0.9612035       0.8849435      0.82087086   0.30962112
##   Eccentricity  ConvexArea EquivDiameter      Extent    Solidity  roundness
## 1 -0.007720825 -0.33212226    -0.3115265  0.55563334  0.41634123  0.4337139
## 2 -0.206667666 -0.77098734    -0.9551559  0.07158789  0.23497487  0.6471257
## 3 -1.886759577 -0.47203971    -0.5027221  0.47527666  0.69190861  1.2422810
## 4  0.214830539  4.10281827     3.6508369  0.54758730 -0.05251343 -0.1492242
## 5  1.293789530  0.03488961     0.1458502 -0.94985006 -0.29444279 -1.3630292
## 6  0.335214864 -0.30487805    -0.2829010 -0.70924229 -0.13014269 -0.1041134
## 7  0.445509869  0.72001278     0.8993504  0.12200539 -0.75550329 -0.8356275
##   Compactness ShapeFactor1 ShapeFactor2 ShapeFactor3 ShapeFactor4
## 1   0.1699453    0.1432647    0.1281886    0.1371831   0.33266875
## 2   0.3717169    1.1934613    0.8785208    0.3427285   0.45390427
## 3   1.6328400   -0.1819300    1.4625033    1.6978747   0.74926526
## 4  -0.1189516   -2.7681633   -1.4640294   -0.1470306  -0.73866950
## 5  -1.6432523    0.3891801   -1.1505301   -1.5693487  -0.63324382
## 6  -0.2187317    0.2929399   -0.1463962   -0.2499168  -0.03121877
## 7  -0.3981079   -1.0388791   -0.8513799   -0.4177491  -0.64381489
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3   3   3   3
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   2   3   3
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   3   3   3   3   3   3   3   3   3   3   2   3   3   3   3   3   3   3   3   3
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
```

```
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
## 241 242 243 244 245 246 247 248 249 250
##   3   3   3   3   3   3   3   3   3   3
##  [ reached getOption("max.print") -- omitted 13361 entries ]
##
## Within cluster sum of squares by cluster:
## [1]  4088.097  5738.789  4285.756  3426.074 12808.067  5222.223 17708.724
##  (between_SS / total_SS =  75.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

This is a small example showing that it seems like the clustering generally placed the observations into the already their already defined classifications that was omitted from the clustering analysis

```
head(data[, c(17,18)])
```

	Class <chr>	fit.cluster <int>
1	SEKER	3
2	SEKER	3
3	SEKER	3
4	SEKER	3
5	SEKER	3
6	SEKER	3

6 rows

```
tail(data[, c(17,18)])
```

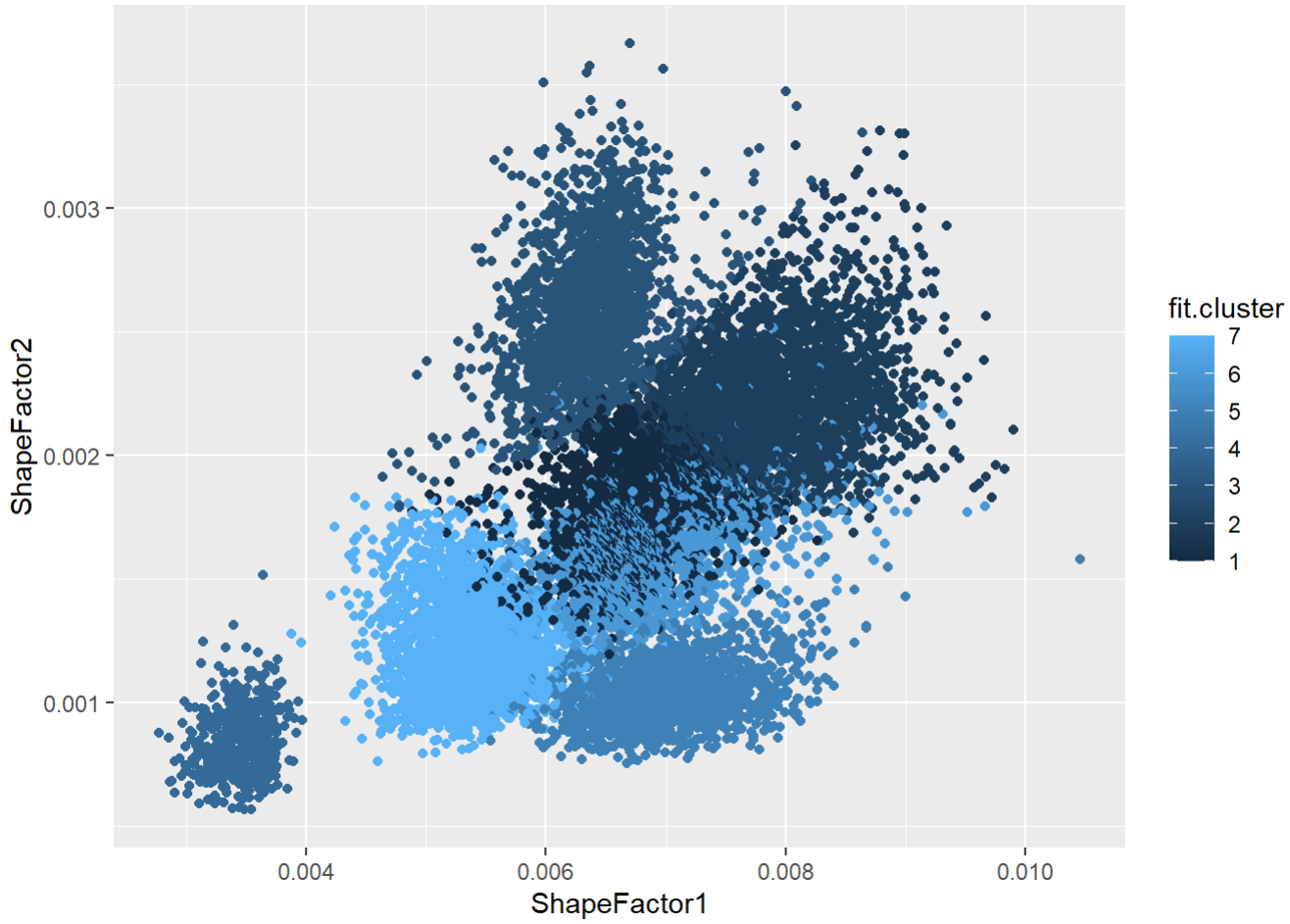
	Class <chr>	fit.cluster <int>
13606	DERMASON	1
13607	DERMASON	1
13608	DERMASON	1
13609	DERMASON	1
13610	DERMASON	6
13611	DERMASON	1

6 rows

A simple k means cluster plot

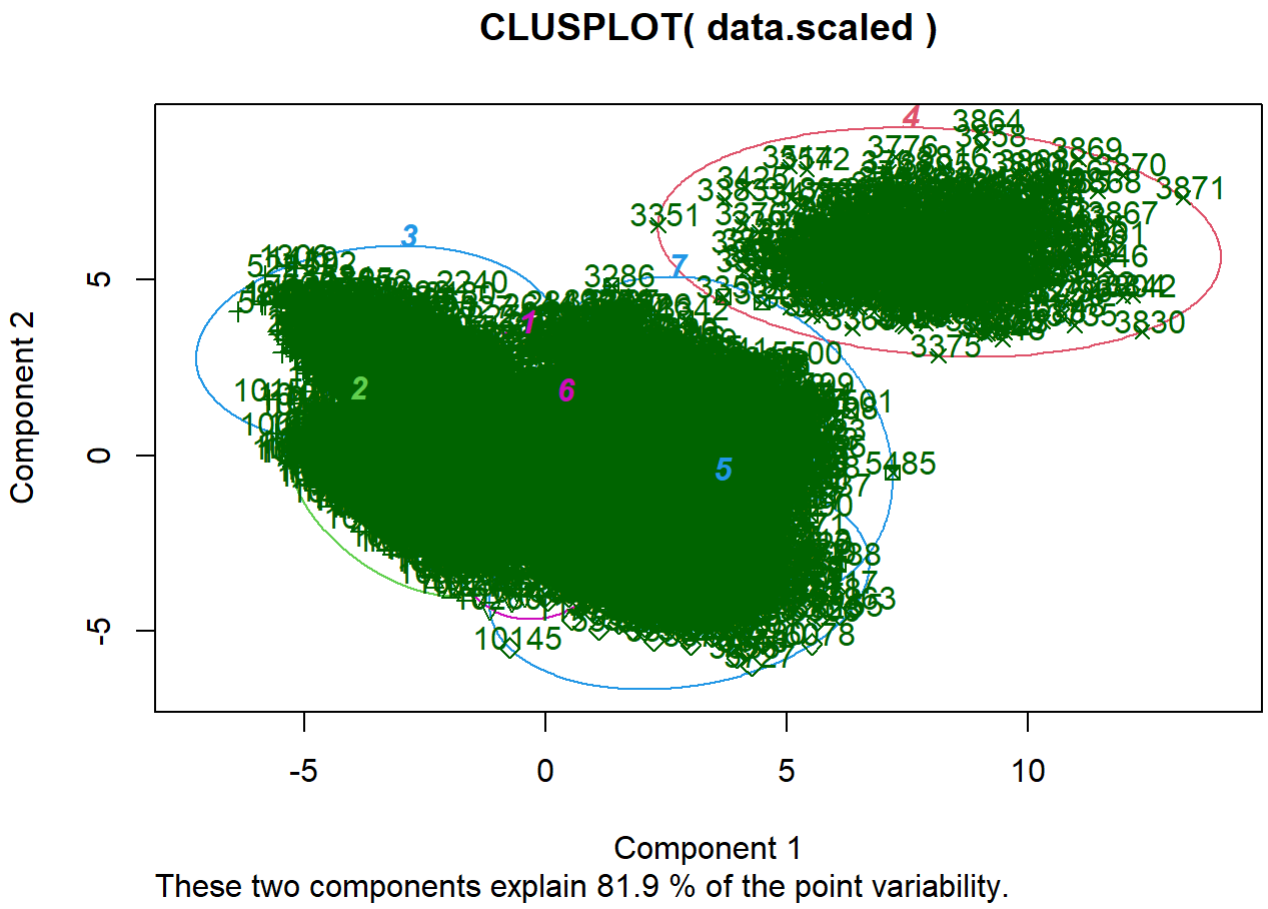
```
library("ggplot2")

ggplot(data,aes(x=ShapeFactor1,y=ShapeFactor2,group=fit.cluster)) +
  geom_point(aes(color=fit.cluster))
```

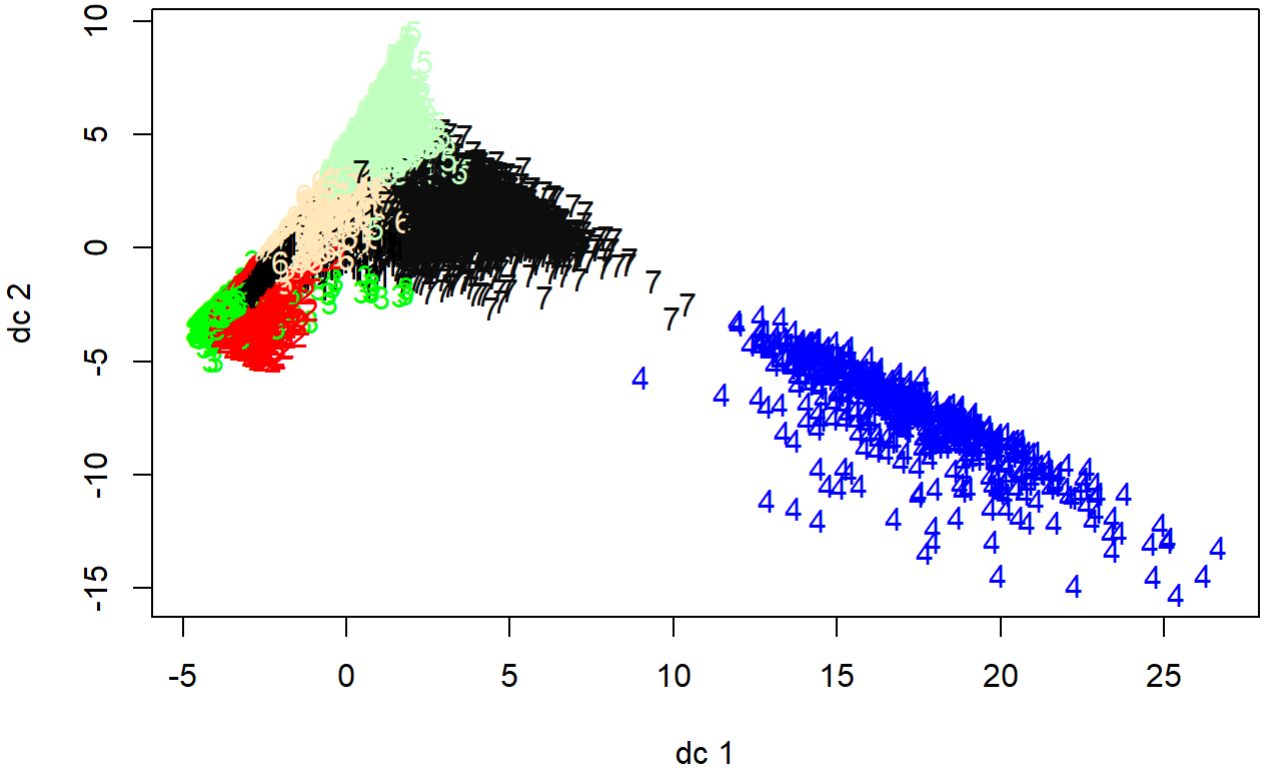


Some more complex cluster plots

```
library(cluster)
clusplot(data.scaled, fit$cluster, color=TRUE,
   labels=2, lines=0)
```

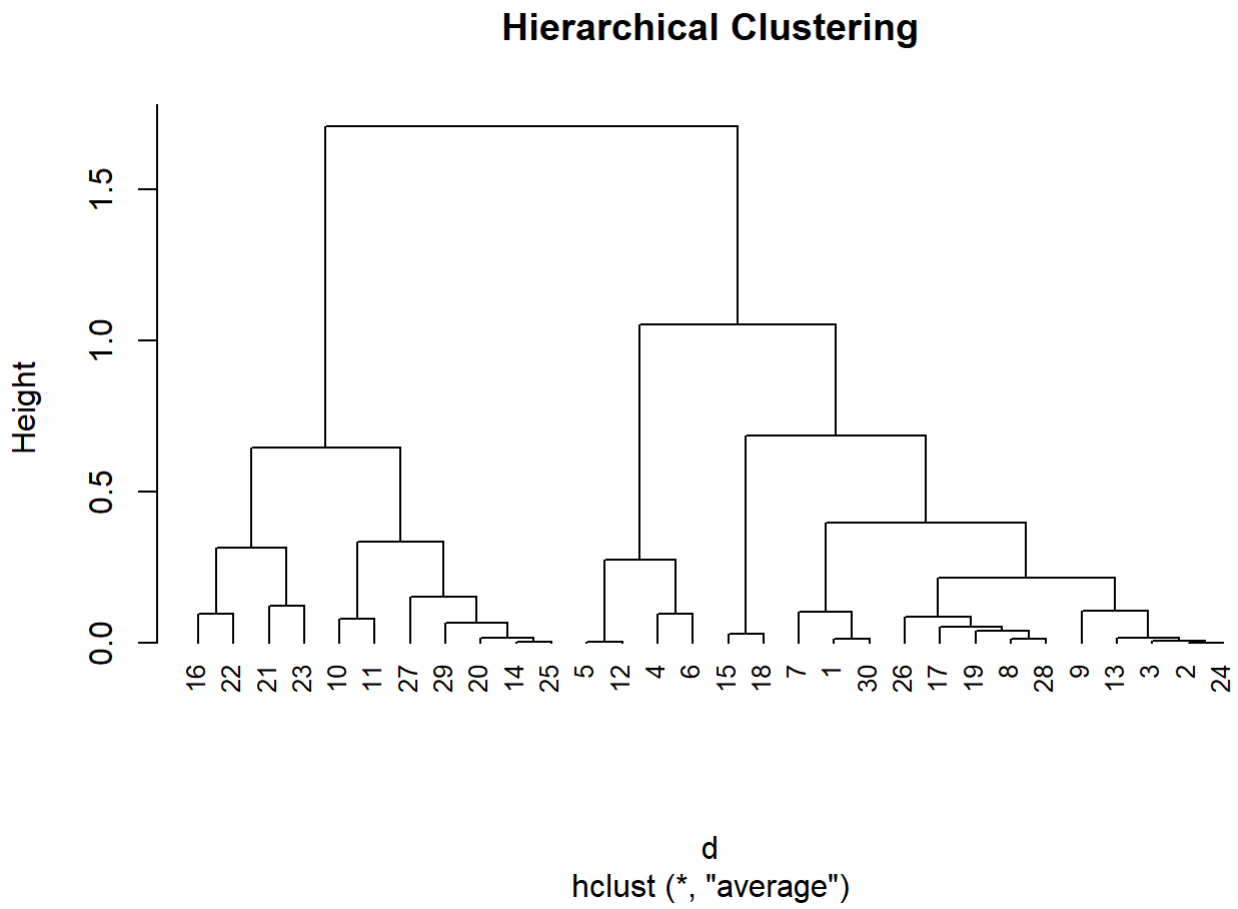



```
library(fpc)
plotcluster(data.scaled, fit$cluster)
```



Hierarchical

```
d <- dist(sample(data.scaled[, c(13,14,15,16)],30))
fit.average <- hclust(d, method="average")
plot(fit.average, hang=-1, cex=.8,
     main="Hierarchical Clustering")
```



Model Based

```
library(mclust)
```

```
## Package 'mclust' version 5.4.10
## Type 'citation("mclust")' for citing this R package in publications.
```

```
fit.m <- Mclust(data)
summary(fit.m)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust VEV (ellipsoidal, equal shape) model with 9 components:
##
##  log-likelihood     n   df     BIC     ICL
##        745655.7 13611 1573 1476339 1476318
##
## Clustering table:
##    1    2    3    4    5    6    7    8    9
##  624 1807 1492  521 1747 1545 3036 1203 1636
```

Comparisons

These three clustering solutions set out to find different things about the data.

Kmeans tries to group observations into meaningful groups. What exactly the groups mean is not always easy to find. In our case we knew to try 7 clusters because this data has observations for 7 different kinds of beans. Kmeans was able catch on to this classification pretty closely, however, classification does not really matter to Kmeans.

Hierarchical clustering tries to find if there is some type of hierarchical taxonomy within the data. Interestingly it found that most beans belong to one of two main families. This is not something that was directly said within the data set, but both hierarchical and kmeans point towards this type of dichotomy.

Finally model-based clustering just told us the general shape of our data. It is VEV (ellipsoidal, equal shape). I could not find much on what exactly that means.

Ultimately these models reaffirmed things we already knew about the data set and gave us some hints into something that can be investigated furthered, like the bean family hierarchy.