Example – Olive Oil data set

Heirarchical Clustering

```
> olive<-read.table("olive.txt",h=T)
> newolive<-olive[,3:10]
```

(1) agnes

Let us try first the “single” linkage:

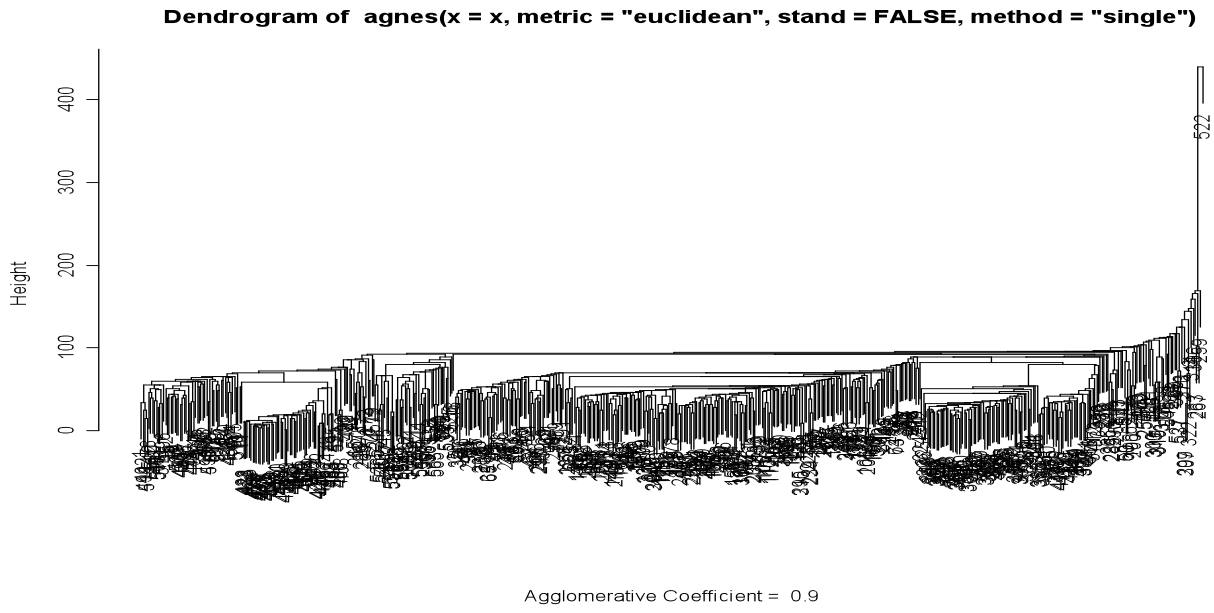
```
> library(cluster)
> x<-daisy(newolive)
> agn<-agnes(x,metric="euclidean",stand=FALSE,method="single")
```

Use the following interactive command for both the “dedrogram” and “banner plot” :

```
> plot(agn,ask=T)
```

or use the following command for only a dendrogram :

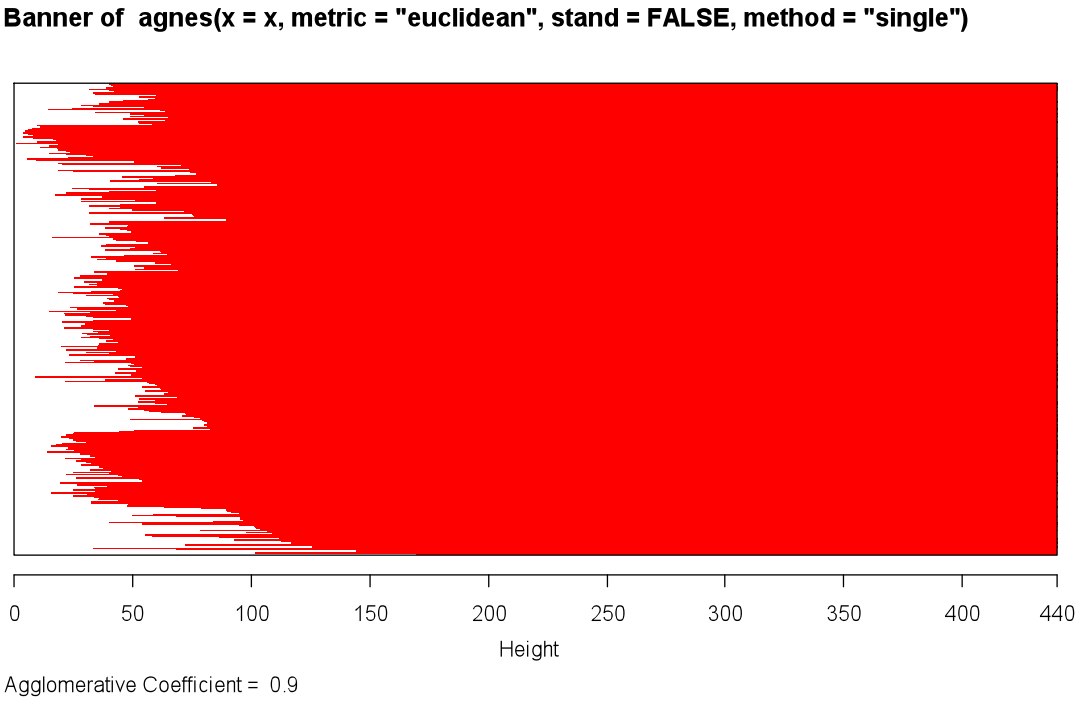
```
> plot(agn,which.plots=2)
```



If you use

```
> plot(agn,which.plots=1)
```

for a “banner plot”, you are not able to get a clear plot since we have more than 500 objects (this is just a horizontal version of the dendrogram).



However, from the output the AC (Agglomerative Coefficient) is derived to be 0.9. You can also check out the AC (Agglomerative Coefficient) using

```
> agn$ac
[1] 0.8980742
```

This shows a pretty good clustering structure.

Check that if the resulting grouping agrees with the original “Regions”:

```
> olive[,1][agn$order]
  [1] 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 3 3 3 3 3 3 1 1 3 3 1 3 3 3 3 3 3 3
 [38] 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1
[112] 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[334] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[371] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[408] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[445] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[482] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1
[519] 1 1 1 1 1 1 1 1 3 1 3 1 1 1 1 1 1 1 1 3 1 3 1 1 1 3 3 1 1 1 1 1 1 1 3 1
[556] 3 1 2 3 3 1 2 1 1 1 1 1 1 1 1 3
```

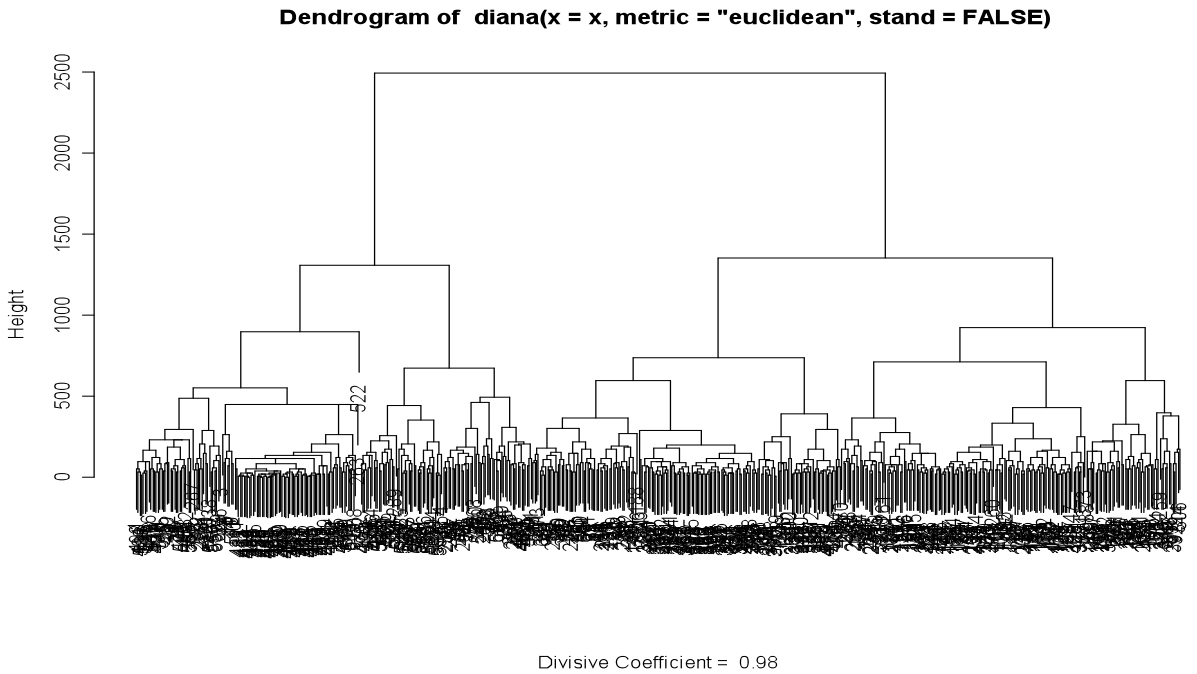
Check that if the resulting grouping agrees with the original “Areas”:

```
> olive[,2][agn$order]
  [1] 1 7 7 7 7 7 7 7 7 7 7 7 7 7 1 1 1 1 1 1 7 7 7 7 7 7 1 1 7 7 1 7 7 7 7 7 7 7
 [38] 7 7 7 7 7 7 7 1 1 7 7 7 7 7 7 7 7 1 1 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 [75] 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 7 9 9 9 7 7 9 9 9 7 1
[112] 7 1 1 1 1 4 1 4 4 4 4 1 4 1 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
[149] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 1 2 2 2 2 2 2 4 2 2 2 2 2 2 3 4
[186] 2 4 2 2 2 2 2 2 2 2 2 4 2 2 2 3 4 2 2 2 4 2 2 2 2 2 2 2 3 3 3 3 2 2 2 3 2
[223] 4 2 2 2 2 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[260] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[297] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[334] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 4 3 3 3
[371] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 4 2 3 2
[408] 2 3 3 4 4 3 2 4 3 4 3 3 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
[445] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
[482] 5 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 2 4
[519] 4 3 4 4 4 4 4 4 7 1 8 2 2 2 2 3 3 3 2 7 4 8 3 3 4 7 8 3 3 3 3 3 3 3 8 4
[556] 8 2 5 8 8 4 6 3 3 3 1 3 3 4 4 4 7
```

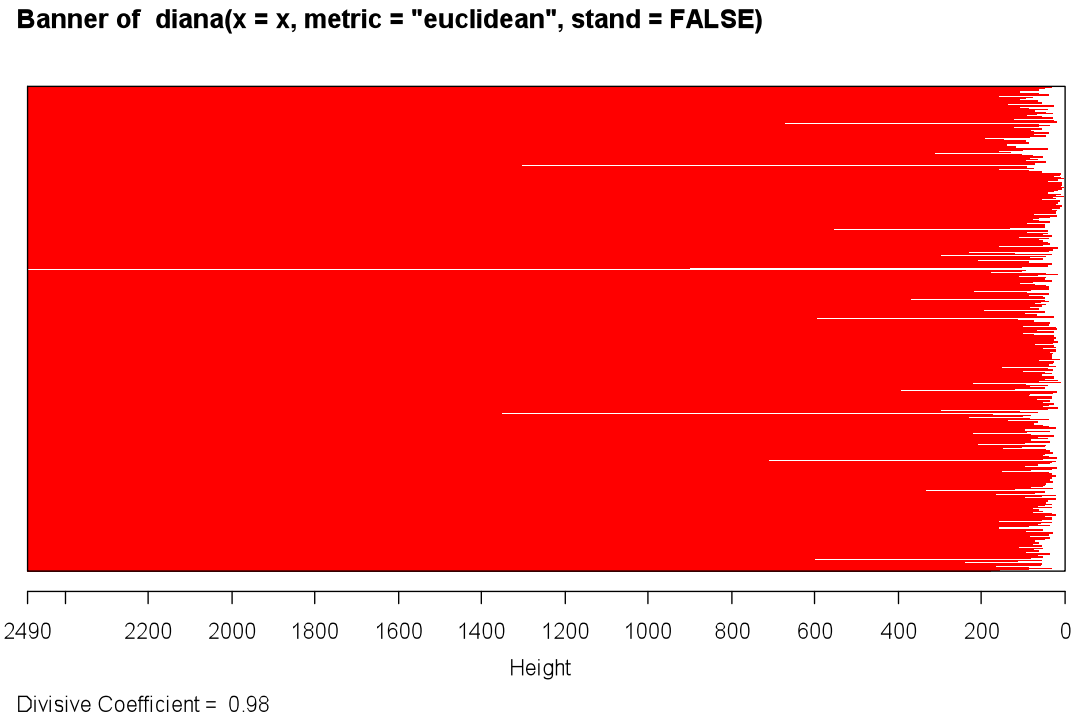
Q: How about using other linkages?

(2) Diana – divisive method (splitting method)

```
> di<-diana(x,metric="euclidean",stand=FALSE)
> print(di)
> plot(di, which.plots=2)
```




```
> plot(di, which.plots=1)
```



You can also check out the DC (Divisive Coefficient) using:

```
> di$dc
[1] 0.977154
```

Note that DC=0.977154 shows a pretty strong clustering.
Check that if the resulting grouping agrees with the original “Regions”:

```
> olive[,1][di$order]
  [1] 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 1 1 1 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1
 [38] 1 1 1 3 3 3 3 3 3 3 1 1 1 1 1 1 3 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [75] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3
[112] 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3
[149] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 3 1 1 3 1 1 3 1 3 1 1 1 1 1 1 3 1 1 1 3 1 1 1 1 3 3 3 3 3 3 3 3 1 1 1 1 1
[223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[297] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[334] 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
[371] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[408] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[445] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[482] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
[519] 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[556] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Check that if the resulting grouping agrees with the original “Areas”:

```
> olive[,2][di$order]
  [1] 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 1 1 7 1 1 1 1 1 1 7 7 7 7 8 8 8 4 1 4 4 4 4
 [38] 1 1 4 8 8 8 8 8 8 8 1 1 1 1 1 1 8 1 1 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 [75] 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 7 9 9 9 1 1 7
[112] 7 7 7 7 7 7 7 7 7 7 4 7 1 7 7 7 7 7 8 1 7 7 8 8 8 8 8 8 8 8 8 4 8 8 8 8 8
[149] 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 2 3 4 2 4 2 2 2 2 4 2 4 2 2 2 2 2
[186] 8 4 3 8 2 4 7 4 7 2 2 2 2 2 2 7 4 4 2 7 4 4 4 4 7 7 7 7 7 7 7 8 1 3 2 4 2
[223] 2 2 2 2 2 2 4 2 2 2 2 2 4 2 2 2 4 2 2 3 3 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
[260] 3 3 2 2 2 2 2 3 4 4 3 3 3 3 3 3 3 8 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
[297] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
[334] 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 5 6 6 6 6 6 6 6 6 6 6 6
[371] 6 6 6 6 6 6 6 6 6 6 6 6 6 5 6 6 2 3 4 4 4 3 3 3 3 3 3 4 3 3 3 3 3 3 3 4 3
[408] 3 3 4 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[445] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3
[482] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 6 6 6 6 6
[519] 6 6 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[556] 3 3 3 3 3 4 3 3 3 3 3 3 3 3 3 3
```

(3) mona() – can be used only for binary data !!

Partitioning Method

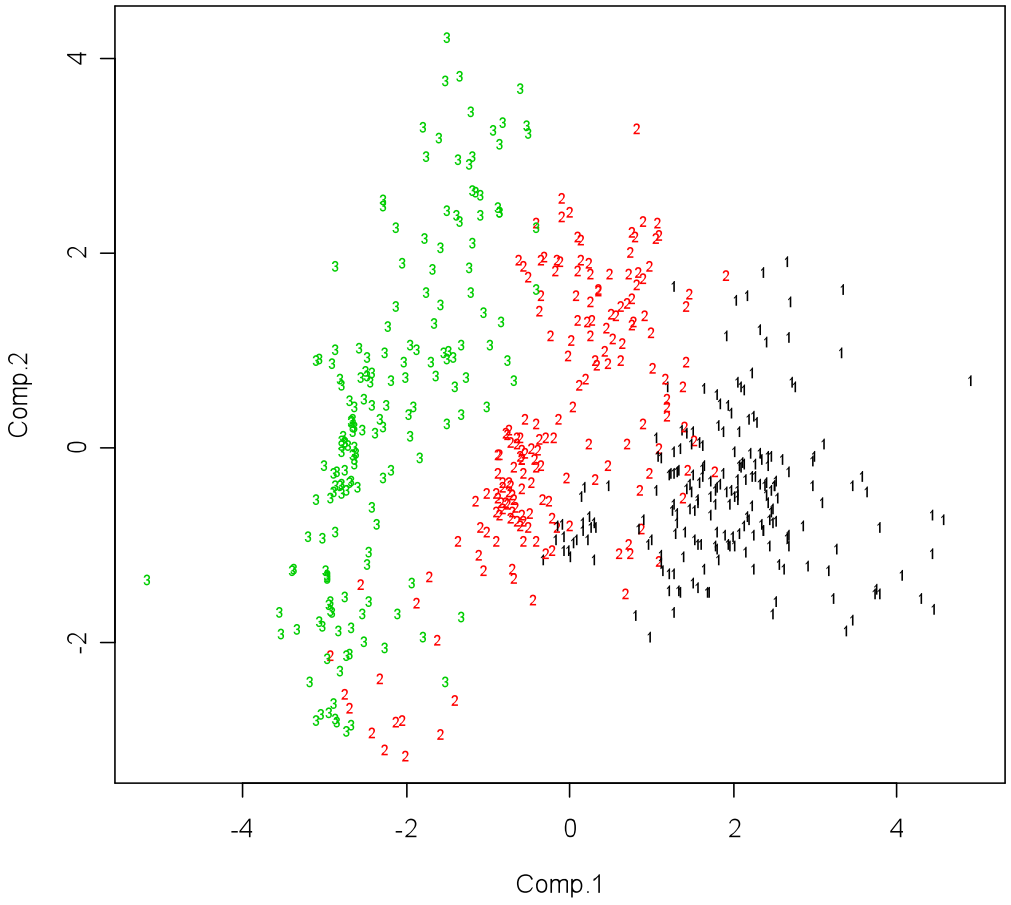
(1) K-means clustering:

Perform K-means clustering considering the number of partitions to be 3 :

```
> km<-kmeans(newolive,3,20)
```

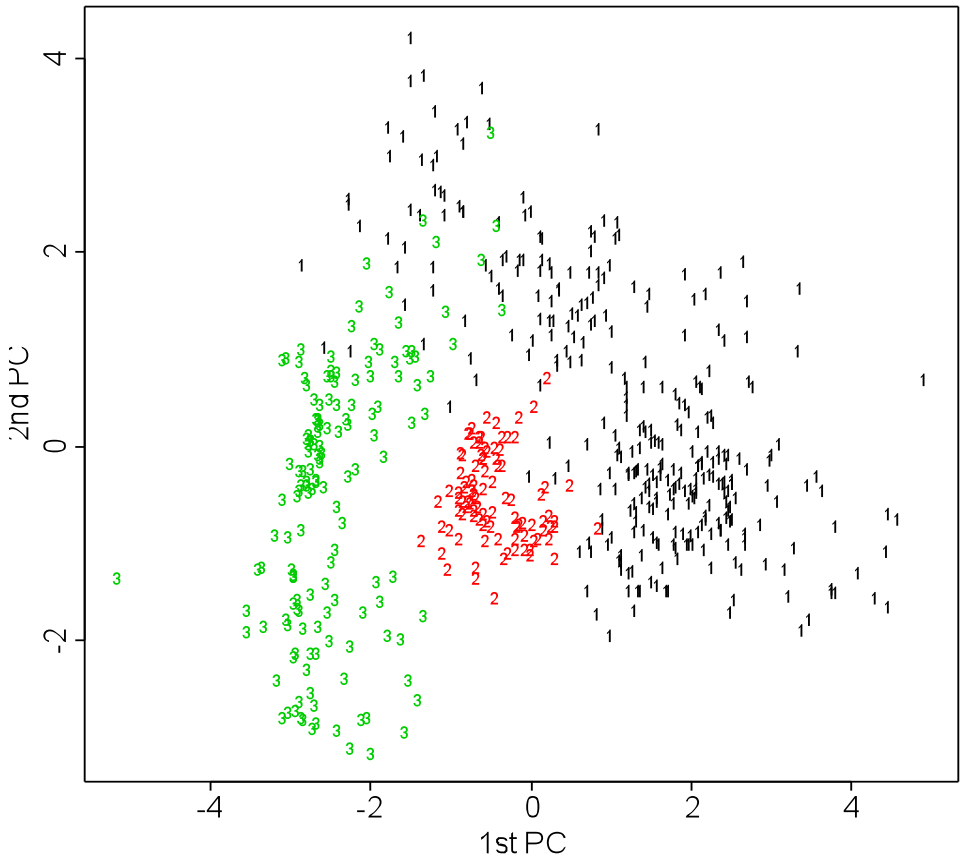
We show the clustering result on the 2-D plane of PC1 vs PC2 :

```
> pca.newolive<-princomp(scale(newolive,scale=TRUE,center=TRUE),cor=FALSE)
> pcs.newolive<-predict(pca.newolive)
> plot(pcs.newolive[,1:2], type="n")
> text(pcs.newolive,as.character(km$cluster),col=km$cluster,cex=0.6)
```



For comparison, a similar plot can be derived from PCA :

```
> plot(pcs.newolive[,1:2],type="n",xlab='1st PC',ylab='2nd PC')
> text(pcs.newolive[,1:2],as.character(olive$Region),col=olive$Region,cex=0.6)
```

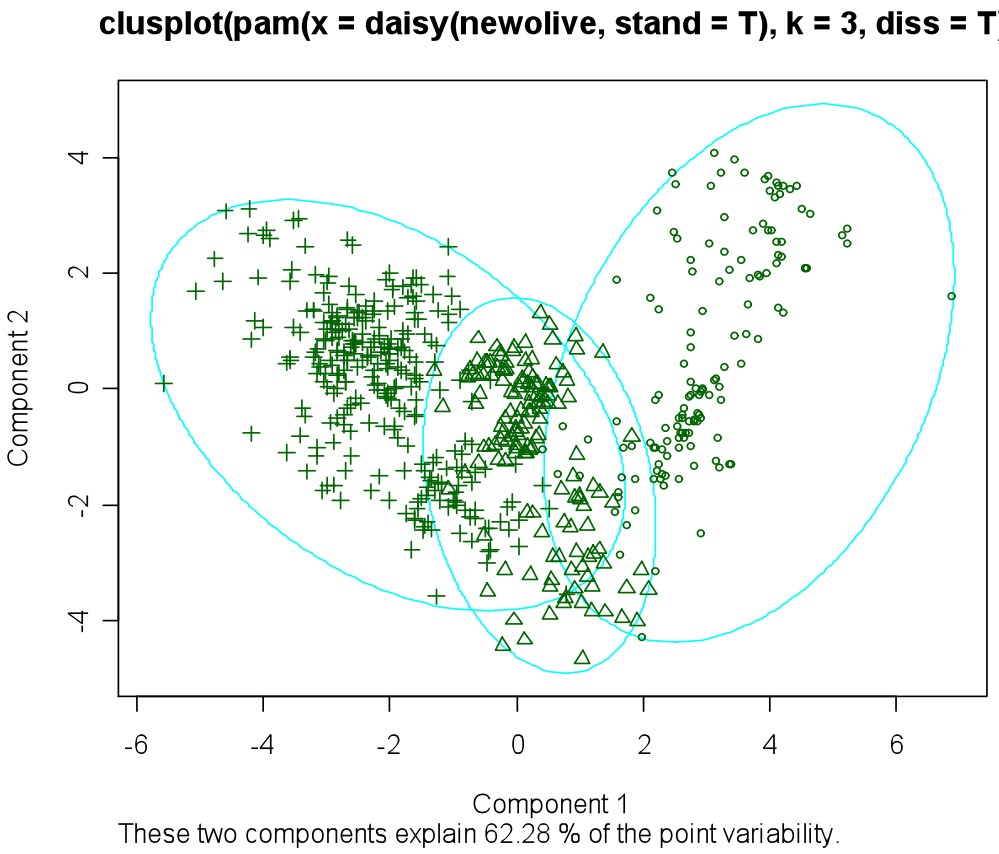


From these two plots, we found that the original regions (shown in PCA) somehow disagree with the K-means clustering, especially on the overlap of “region 1” and “region 2”, the overlap of “region 1” and “region 3”.

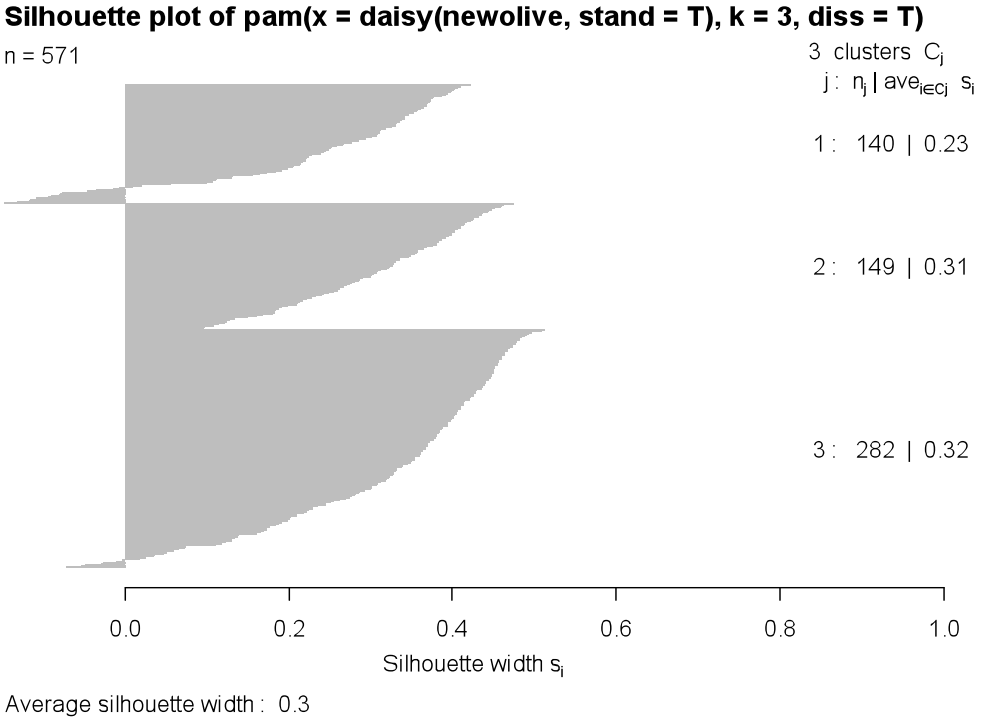
(2) pam

```
> pa<-pam(daisy(newolive,stand=T),3,diss=T)
> plot(pa,ask=T)
```

The clustering result (which takes a few seconds) is projected on a 2-D PC or MDS space:



Plot the silhouette plot :



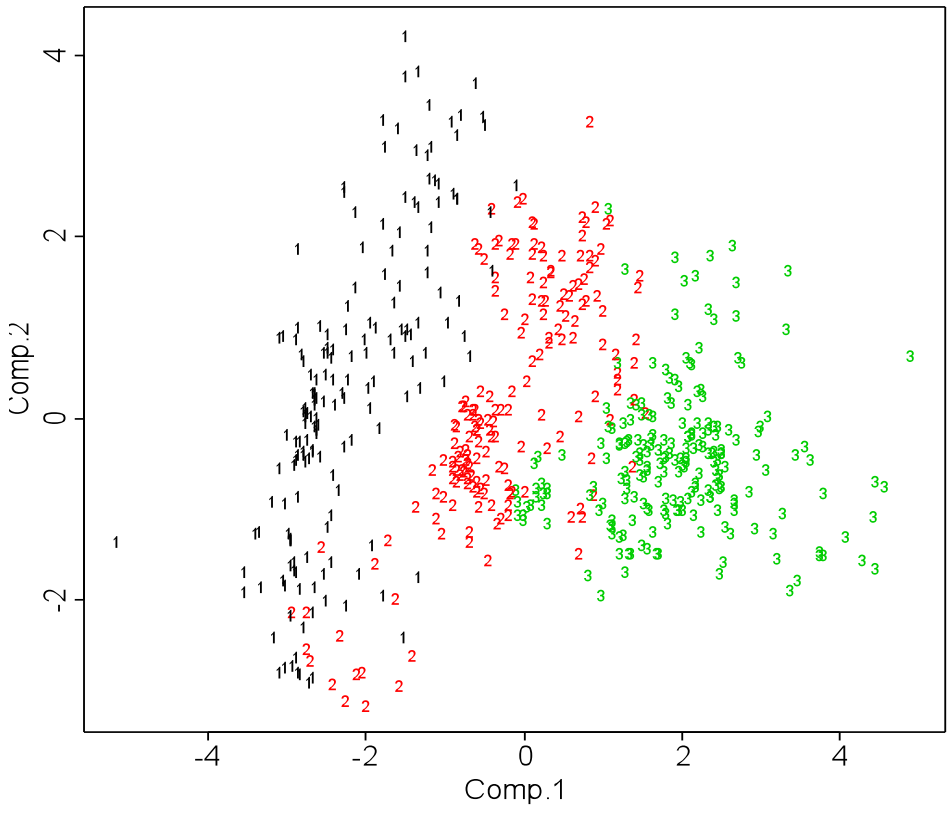
The SC (Silhouette Coefficient) is derived to be 0.3, which shows a weak structure of clustering.

We can use the following command to see if the clustering recovers the original groups of "Regions" :

```
> pa$clustering
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1
 21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
  1   1   1   1   1   2   2   2   2   2   2   3   2   2   2   2   2   2   2   2
.....
.....
.....
561 562 563 564 565 566 567 568 569 570 571 572
  1   1   1   1   1   1   1   2   1   1   1   1
```

We can also compare this result with PCA :

```
> plot(pcs.newolive[,1:2], type="n")
> text(pcs.newolive,as.character(pa$clustering),col=pa$clustering,cex=0.6)
```



Q: Does the resulting grouping recover the original groups of "Regions" ?

Q: Change the number of clusters, is the result better?

(3) Try **clara()**, which is faster for large data.

Self-Organizing Maps (SOM)

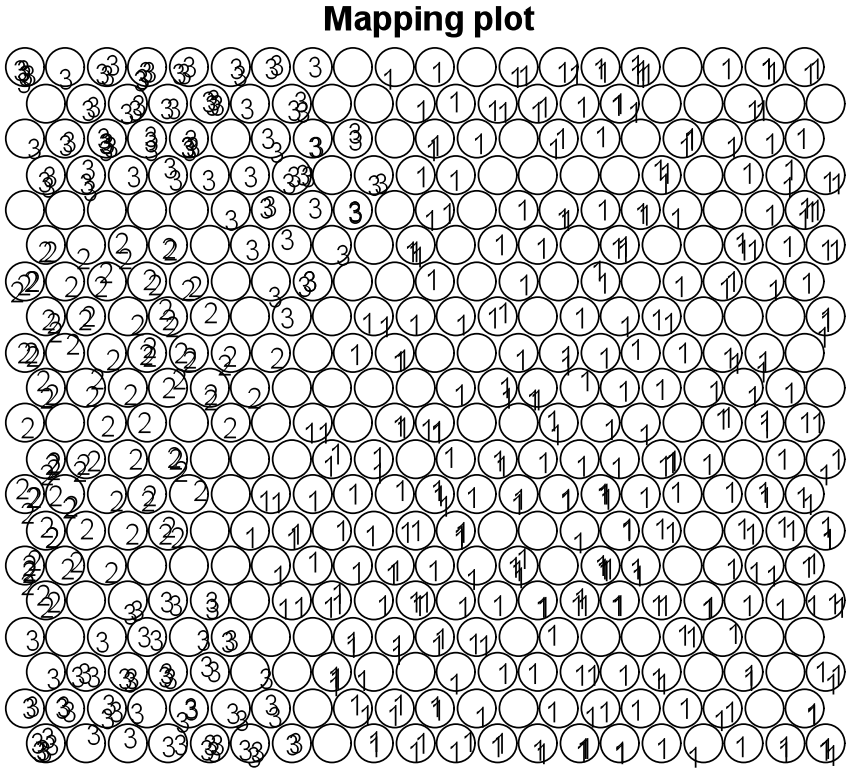
```
> install.packages('som')
> library(som)
> n.newolive<-normalize(newolive)    /* Standardize variables*/

> install.packages('kohonen')
> library(kohonen)
```

Make a plot with 20x20=400 grids (neurons):

```
> olive.som<-som(n.newolive,grid = somgrid(20, 20, "hexagonal"))
```

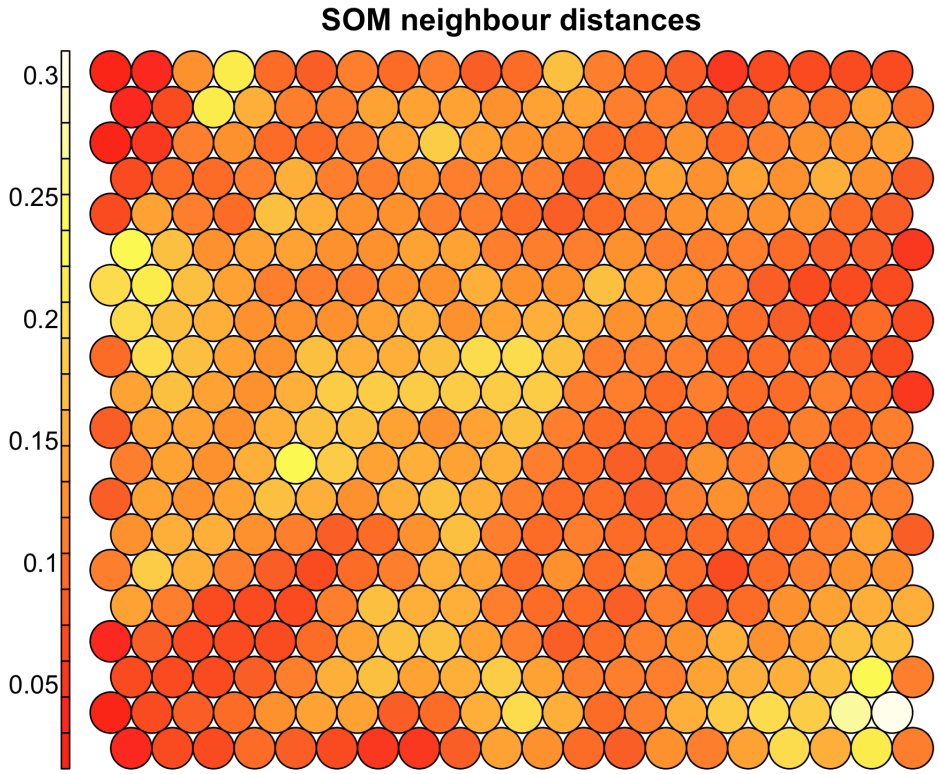
We first mark the labels of “Region” in the resulting SOM:

```
> plot(olive.som,type="mapping",labels=olive[,1])
```



Q: Does the resulting SOM recover the grouping based on “Region”?

Another display to show clustering:
> plot(olive.som, type="dist.neighbours", main = "SOM neighbour distances")

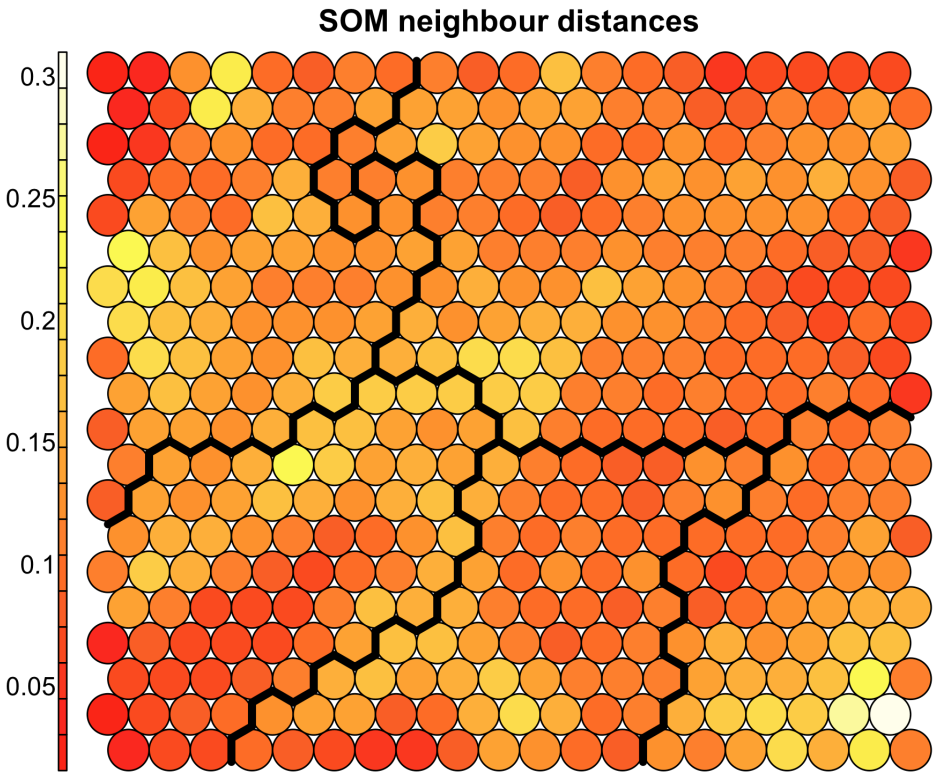


>
We can use the **agglomerative hierachical clustering** to help us cluster the codebook/weight vectors (the object "olive.som$codes") and then add the boundaries of clusters in the plot:

> som.hc <- cutree(hclust(dist(olive.som$codes)), 5)

Here we ask for the display of 5 clusters, which is based on the highest SC of pam().
Adding the boundaries in the plot:

> add.cluster.boundaries(olive.som,som.hc)



Observe the detailed clustering for each object:

```
> cutree(hclust(dist(olive.som$codes)), 5)
[1] 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 1 1 1 1 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 1 1 1 1 1 2 2 2
2 2 2 2 2 2 3 3 3 3 3 3 1 1
[63] 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 1 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 1 1 1 1 1 1 1 1
2 2 2 2 2 3 3 3 3 3 3 1 1 1 1
[125] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 3 4 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 4 1 1 1 1 1 1 1 1 2 2
2 2 2 2 2 3 3 3 3 4 4 4 4 4 1
[187] 1 1 1 1 5 5 5 5 5 5 5 3 3 3 4 4 4 4 4 4 4 1 1 1 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 5 5 5 5 5 5
5 5 5 5 5 5 5 4 4 4 4 4 4 4 5
[249] 5 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5
5 5 5 5 5 4 4 4 4 4 4 5 4 5 5
[311] 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 5 4 4 5 5 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5
5 5 5 4 4 4 4 4 4 4 5 5 5 5 5
[373] 5 5 5 5 5 5 5 5 5 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5
```