Data Mining (W4240 Section 001)
Clustering (part 2)

Giovanni Motta

Columbia University, Department of Statistics

December 2, 2015

Outline

Outline

Mixture Models

The *Gaussian mixture model*: loosely, cluster with *soft* assignments

To generate data from a GMM:

- choose cluster with $c_i \sim Categorical(p_1, ..., p_p)$
- generate point x_i with $x_i | c_i = k \sim \mathcal{N}(\mu_k, \Sigma_k)$
- (μ_k is mean vector, Σ_k is covariance matrix)

As with K-Means, we generated data with:

- observation x_i in cluster c_i
- K clusters
- Our goal is use data to find μ_k (and Σ_k)

Mixture Models

Mixture models are fit using the following iterative steps:

1. E-step:

$$\mathbb{P}(c_i = k \,|\, x_i) = \frac{\pi_k N(x_i \,|\, \mu_k, \Sigma_k)}{\sum_{\ell=1}^{K} \pi_\ell N(x_i \,|\, \mu_\ell, \Sigma_\ell)}$$

π: Prior prob(y==k)

2. M-step:

$$\mu_k^{new} = \frac{1}{N_k} \sum_{i=1}^{n} \mathbb{P}(c_i = k \,|\, x_i) x_i$$

EM 算法

$$\pi_k^{new} = \frac{1}{N} \sum_{i=1}^{n} \mathbb{P}(c_i = k \,|\, x_i)$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{i=1}^{n} \mathbb{P}(c_i = k \,|\, x_i)(x_i - \mu_k^{new})(x_i - \mu_k^{new})^\top$$

- (the Expectation-Maximization algorithm)
- Let's compare this conceptually to K-Means

Clustering with Gaussian Mixture Models

- have real valued data, $X \in \mathbb{R}^{n \times p}$
- fit with a mixture of K clusters
- each observation x_i has cluster variable c_i
- given cluster k, fit data with a Gaussian distribution

$$x_i \,|\, c_i = k \sim N(\mu_k, \Sigma_k)$$

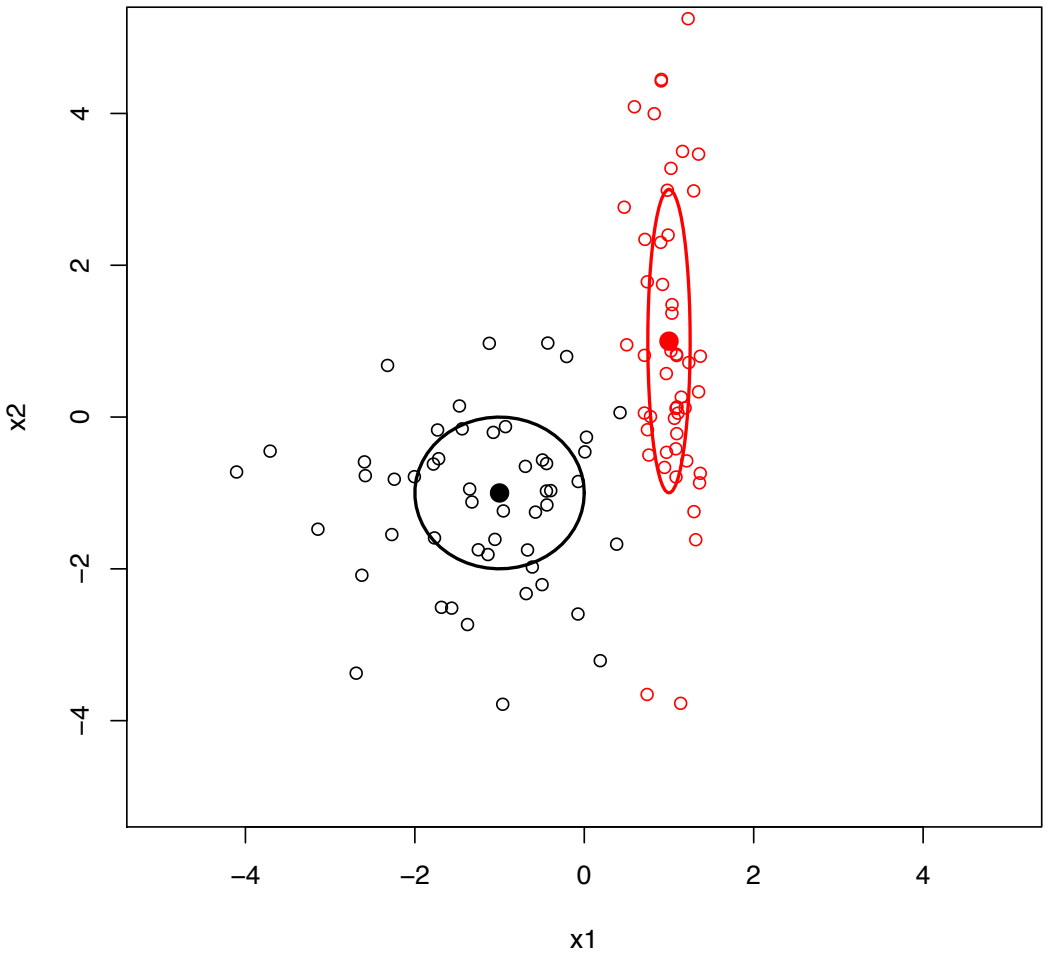
- density of data is

$$\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i; \mu_k, \Sigma_k)$$

Use iterative procedure to cluster distribution, update parameters, update cluster distribution, update parameters...

Mixture Models

Gaussian mixture model ($K = 2$):



Mixture Models

Extensions:

- if we have categorical data, we might model this with a multinomial distribution

$$x_i \,|\, c_i = k \sim Multi(p_{k,1}, \ldots, p_{k,M})$$

- actually, we can use any parametric distribution that we like
- creates a flexible model for mixed data types

Important Caveat: clusters need to be well-modeled by components of a mixture

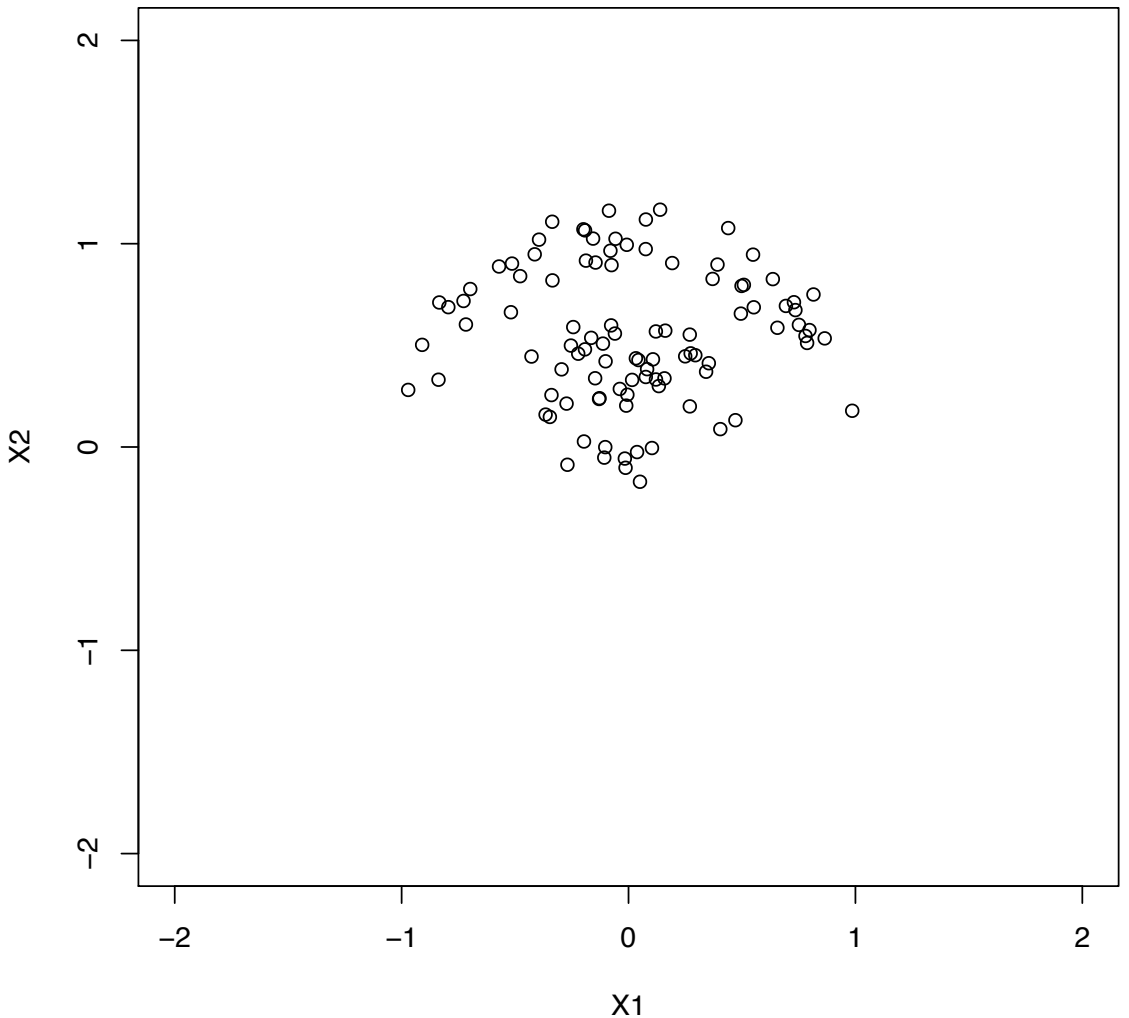
Mixture Models in R

We can use the mclust package in R to fit a mixture model

```
> library(datasets)
> library(mclust)
> faithful.mm <- Mclust(faithful)
> summary(faithful.mm)
> plot(faithful.mm)
> faithful.mm2 <- Mclust(faithful,G=2)
```

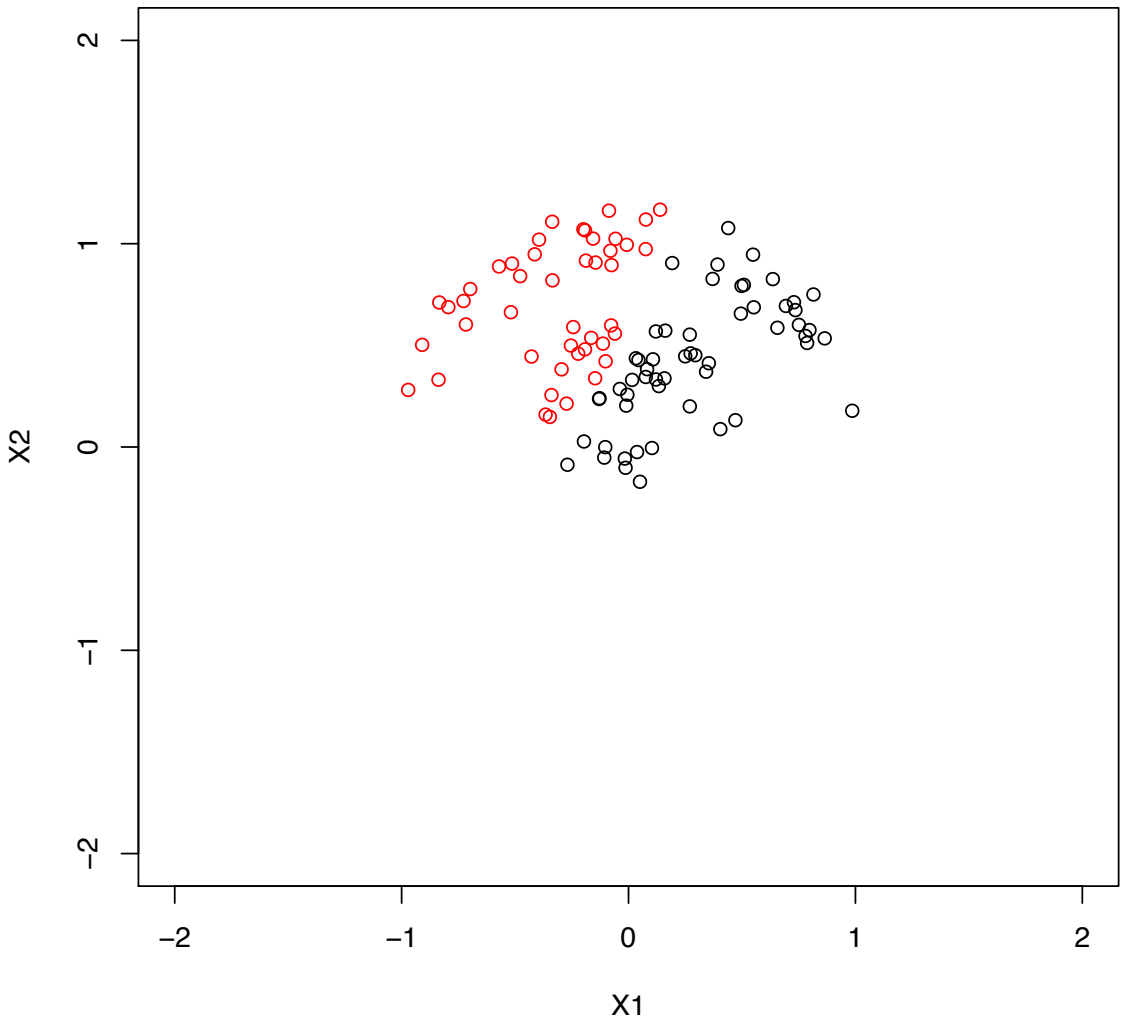
New Data

So what about this data?



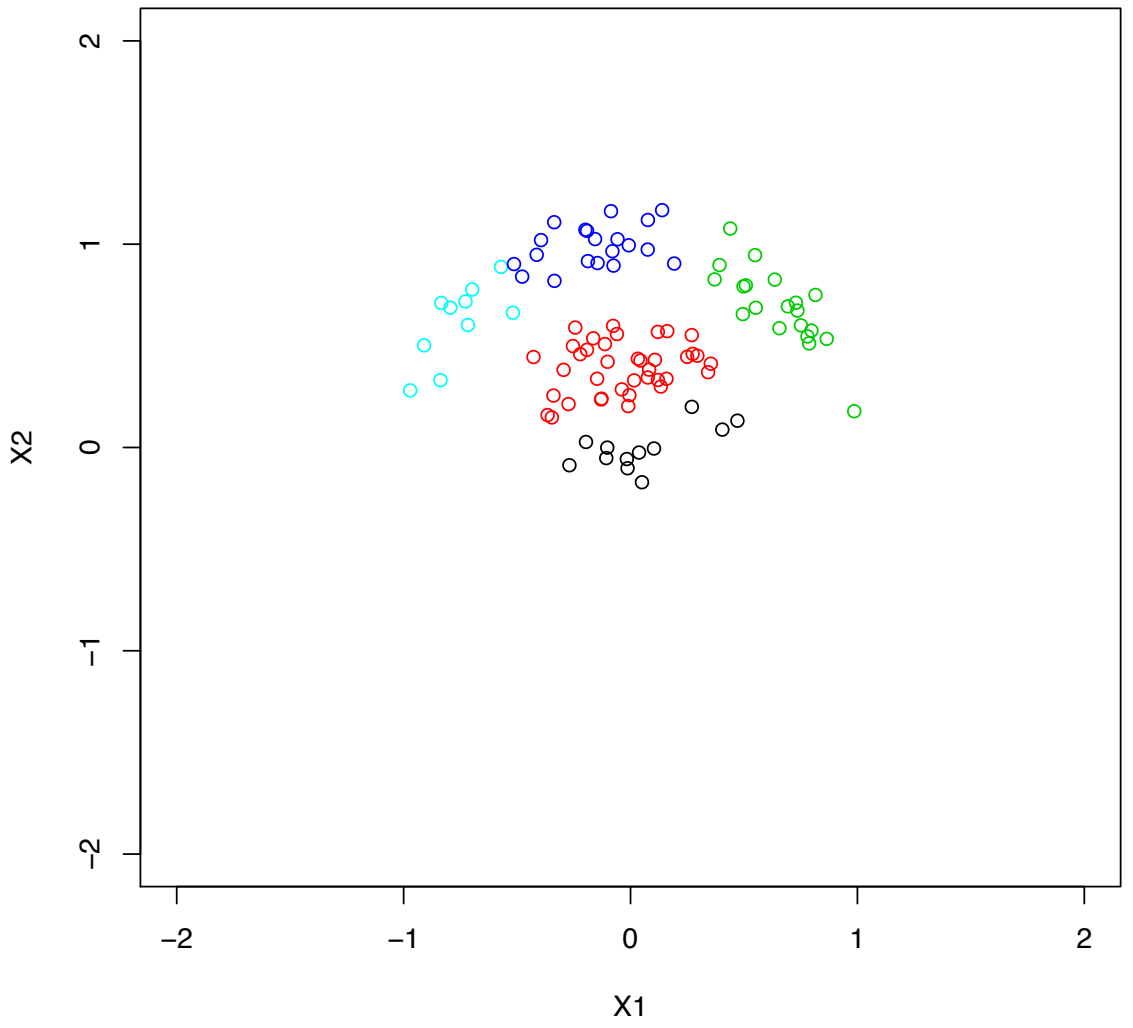
New Data

Fit with K-means:

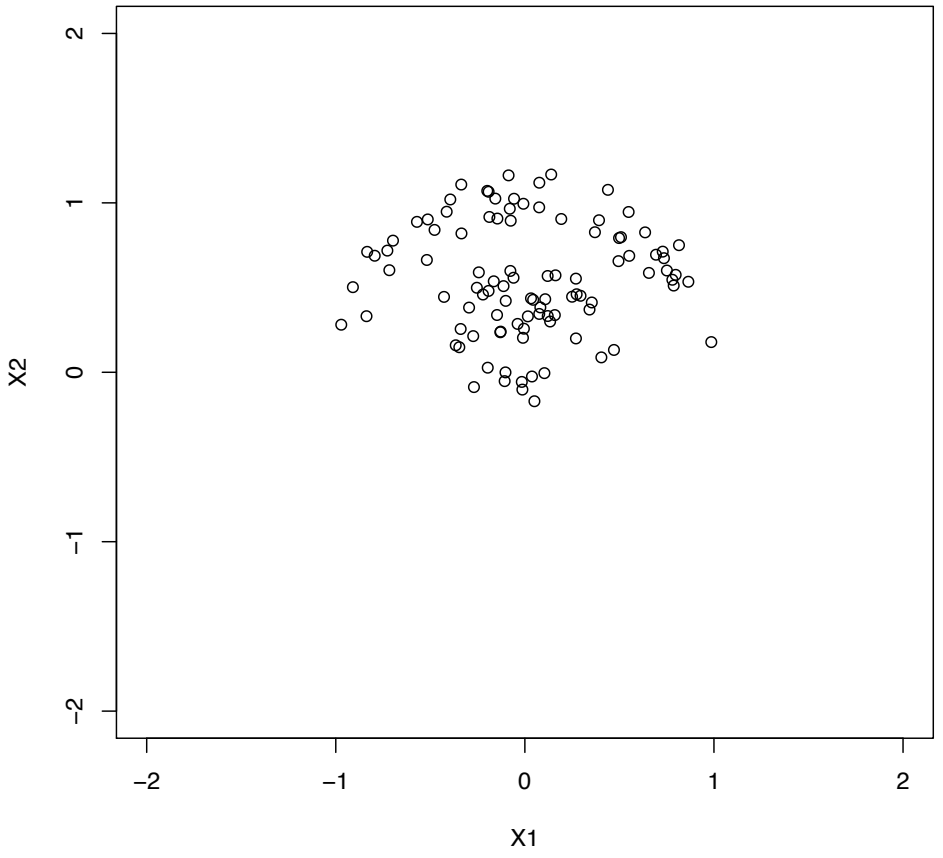


New Data

Fit with a Gaussian mixture model (`Mclust` in the `mclust` package):



New Data



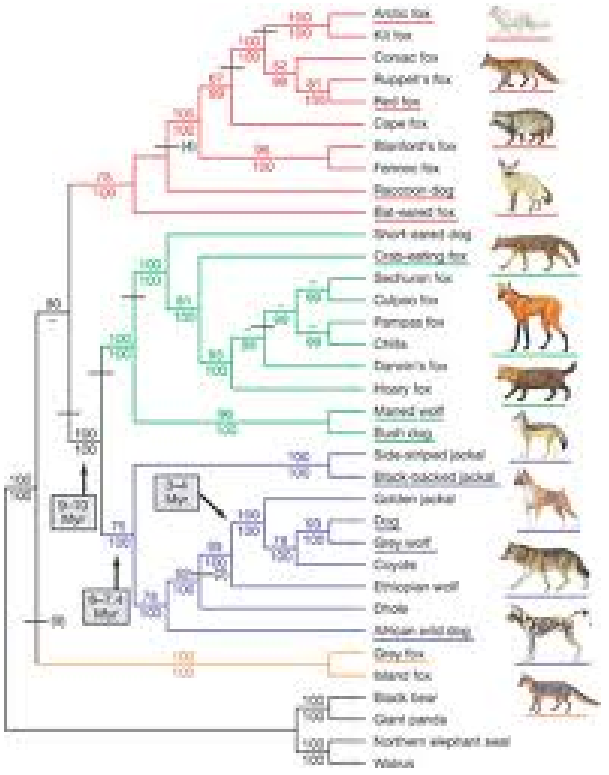
One of the clusters is not well-modeled by a Gaussian distribution...

Outline

Hierarchical Clustering

Idea:

- build a binary tree to represent clustering
- successively merge separate groups
- trees are quite useful for visualizing data



Hierarchical Clustering

What do we need for hierarchical clustering?

K-Means needs:

- a number of clusters K
- an initial clustering
- a distance between points, $d(x_i, x_j)$

Hierarchal clustering needs:

- a distance between *groups of data points*

Outline

Agglomerative Clustering

There are two ways to do hierarchical clustering:

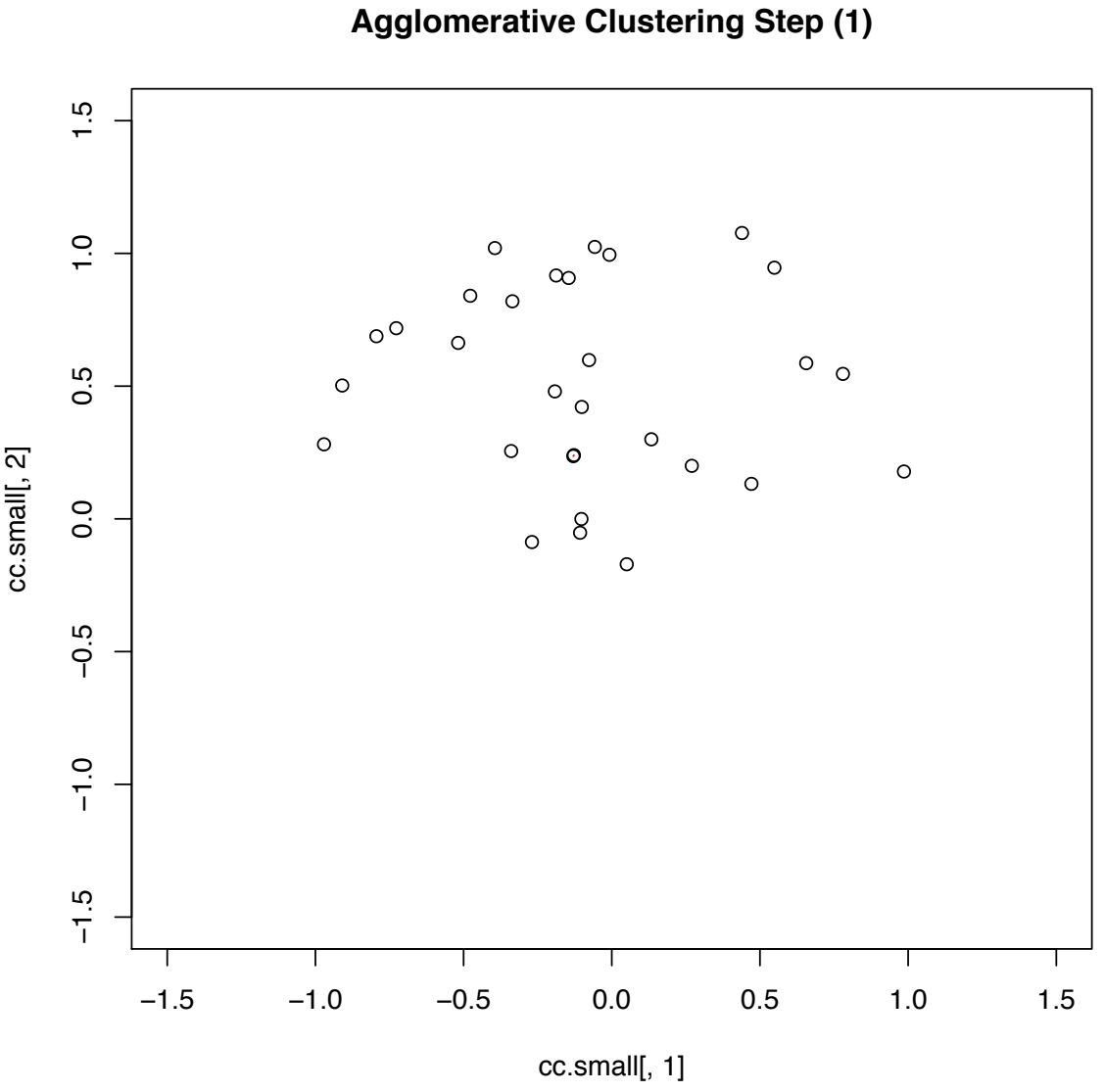
- agglomerative clustering
- divisive clustering

Let's start with *agglomerative clustering*

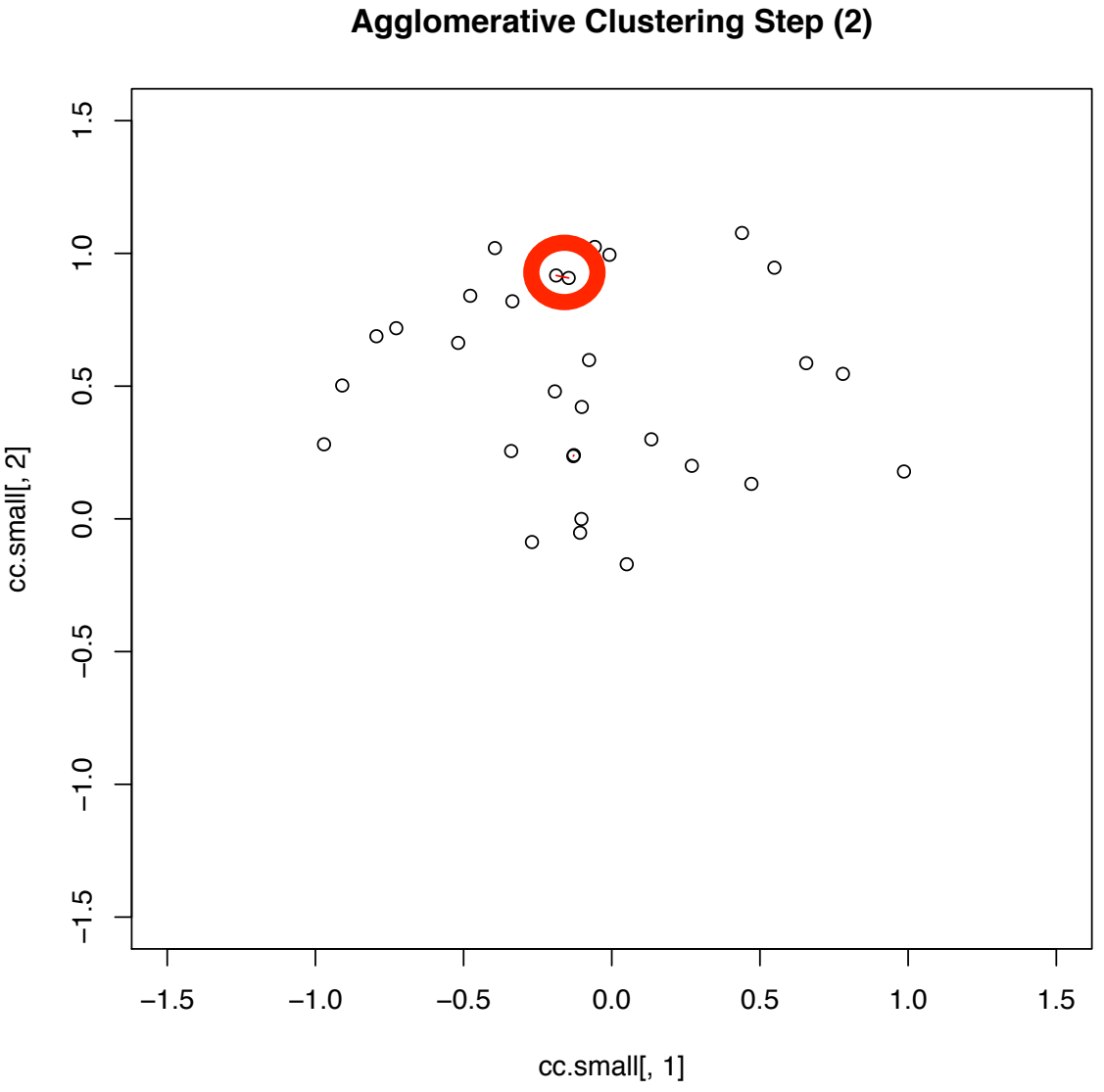
Basic algorithm:

- start with all data points in individual groups
- repeat: merge the two "closest" groups
- stop: when all groups have been merged into a single group

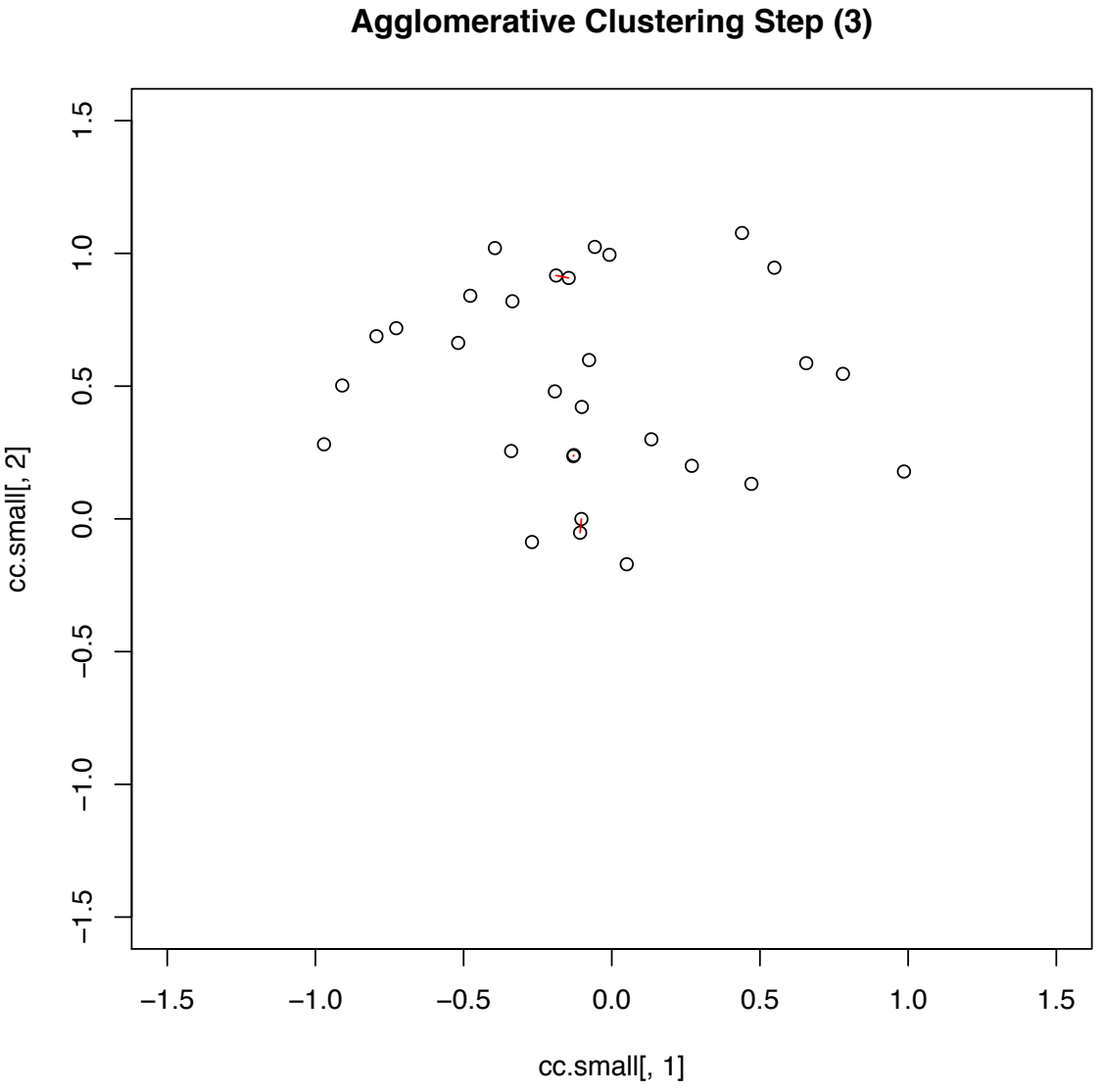
Agglomerative Clustering



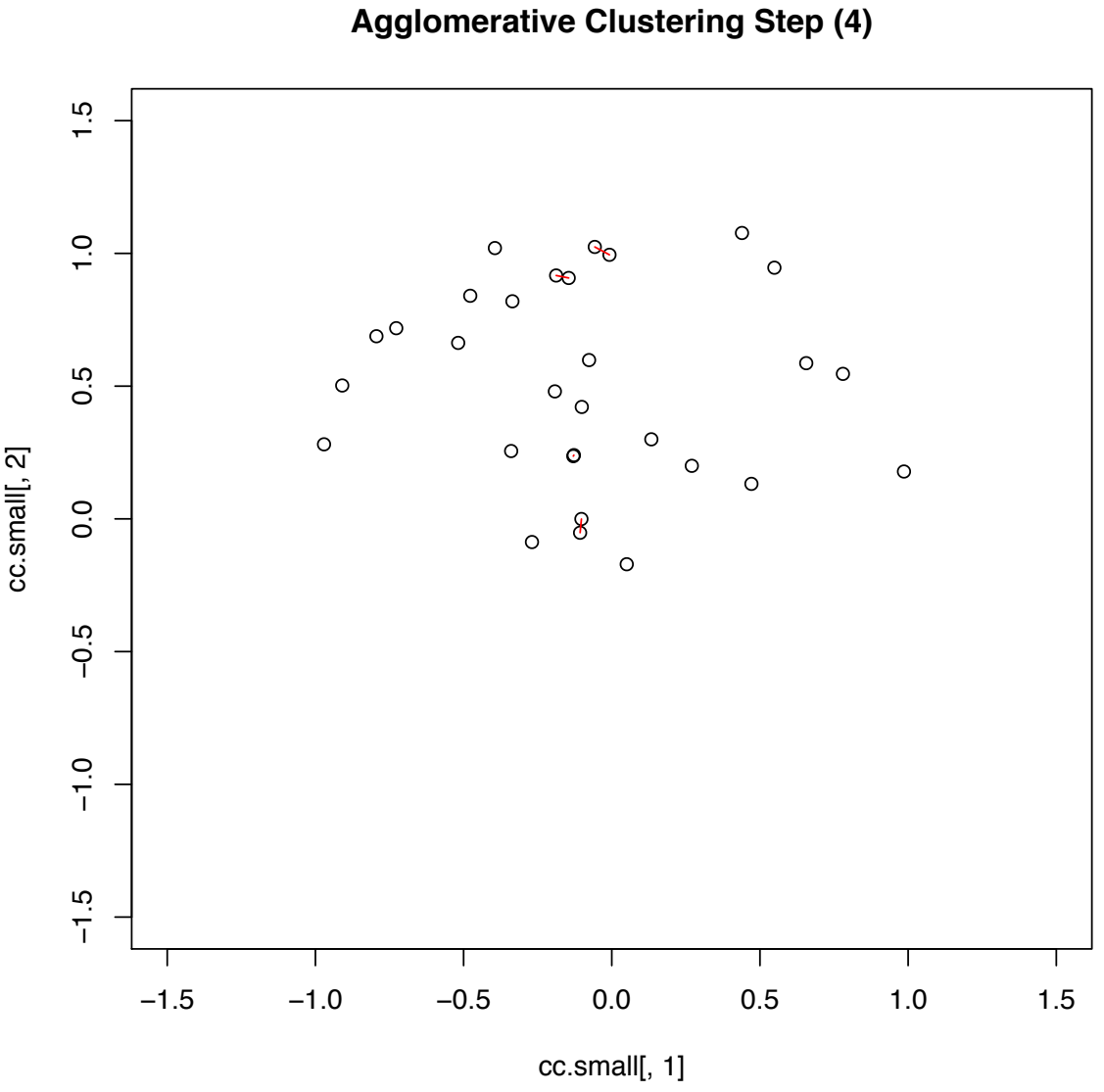
Agglomerative Clustering



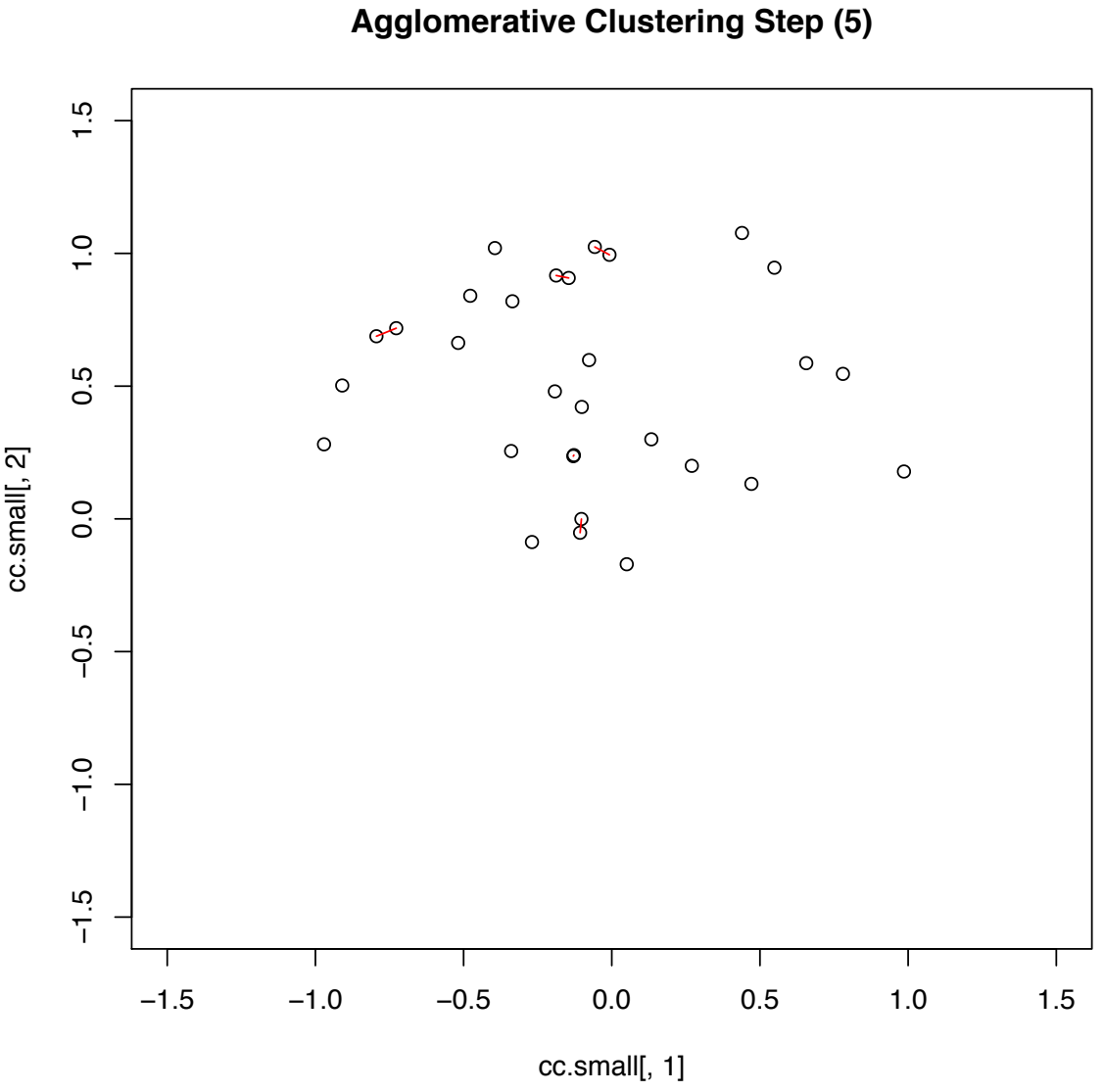
Agglomerative Clustering



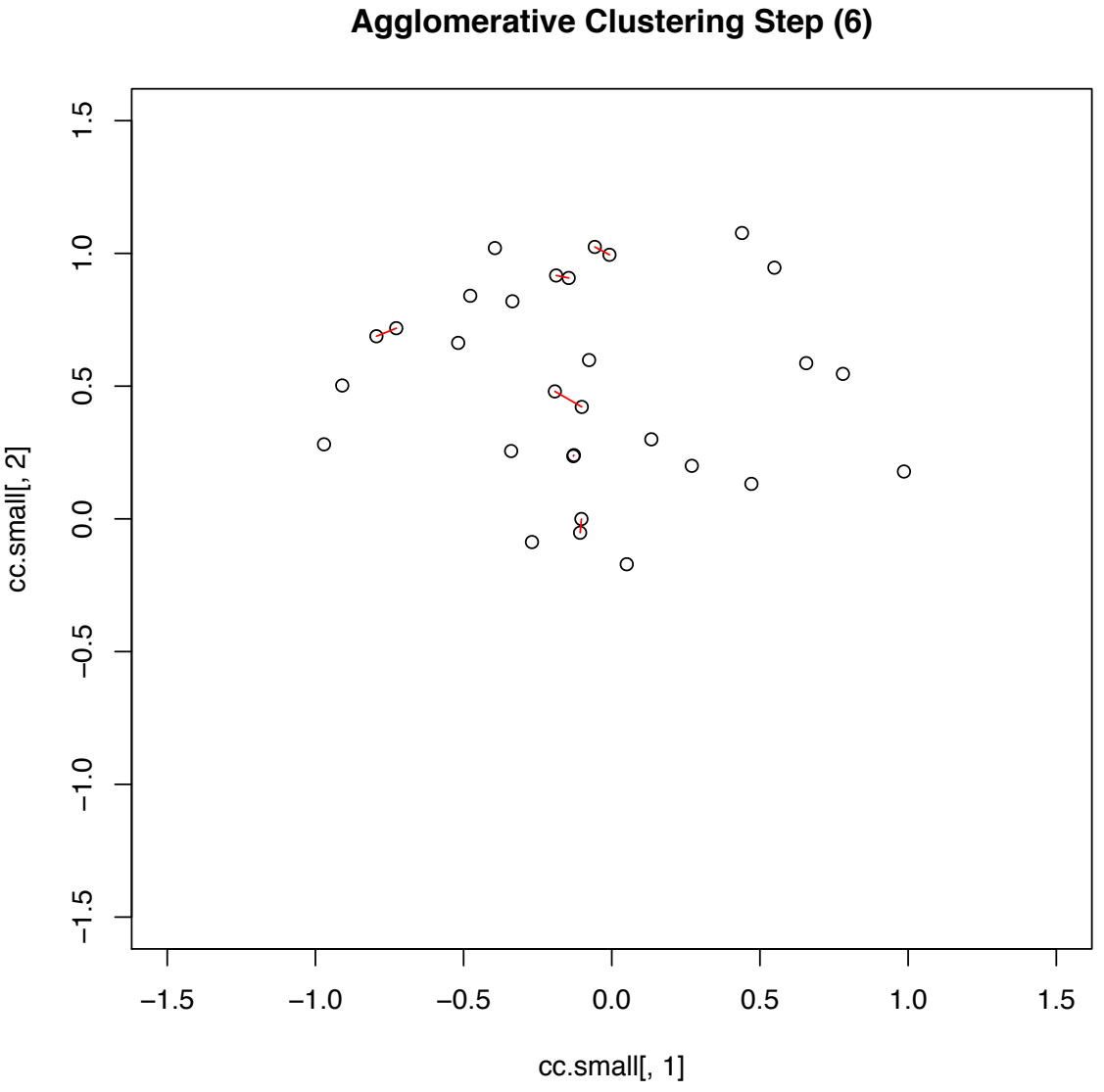
Agglomerative Clustering



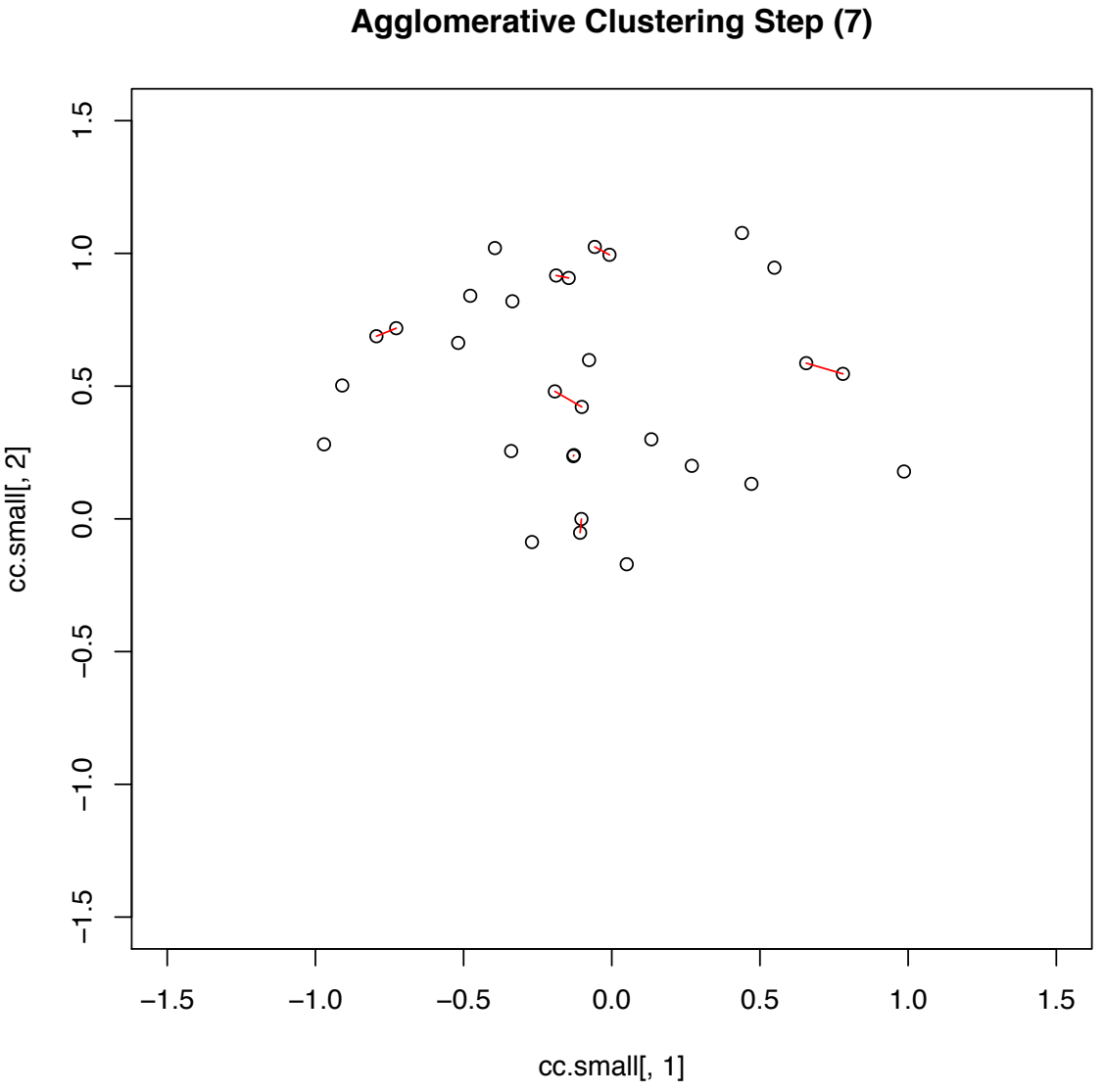
Agglomerative Clustering



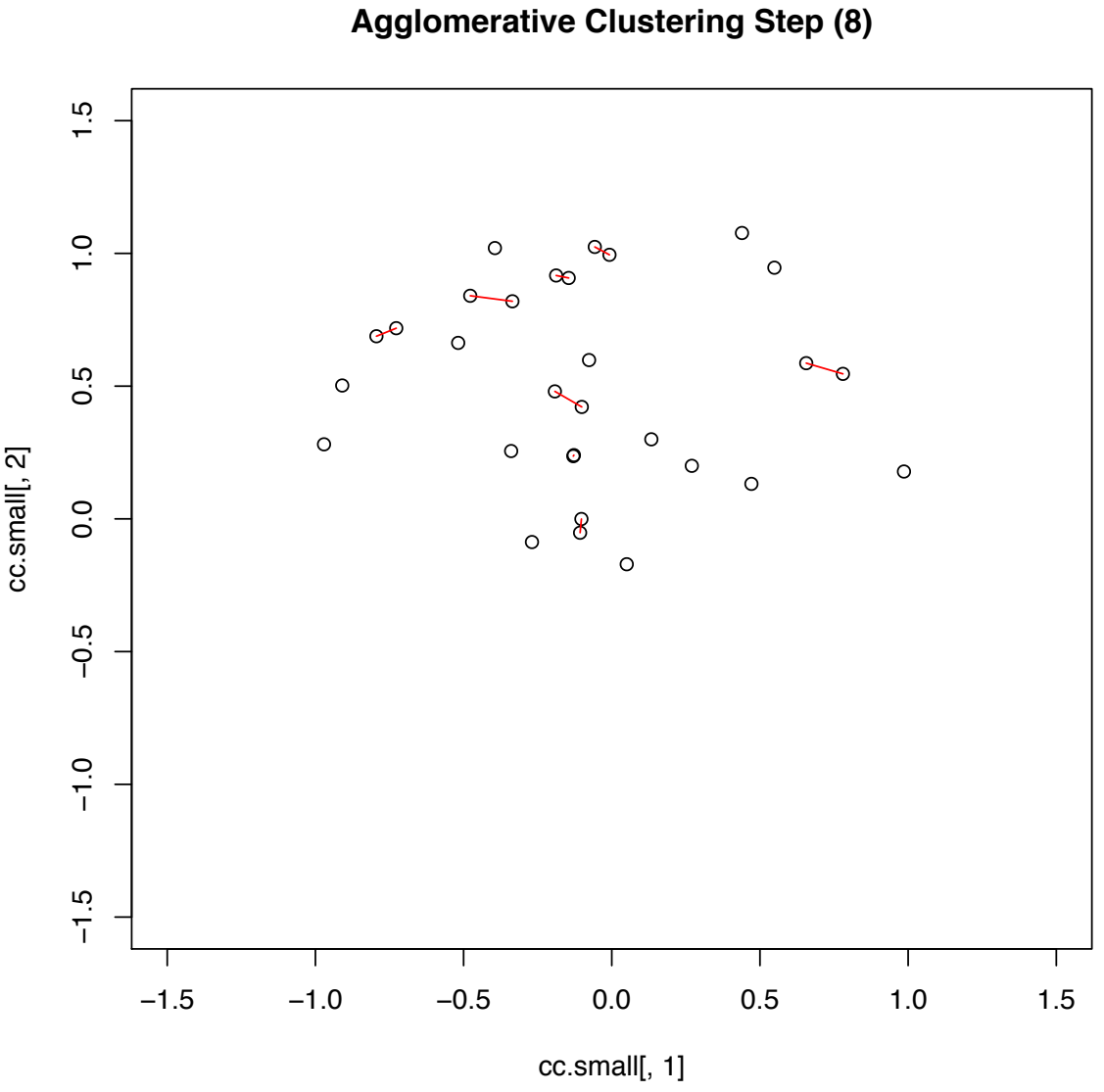
Agglomerative Clustering



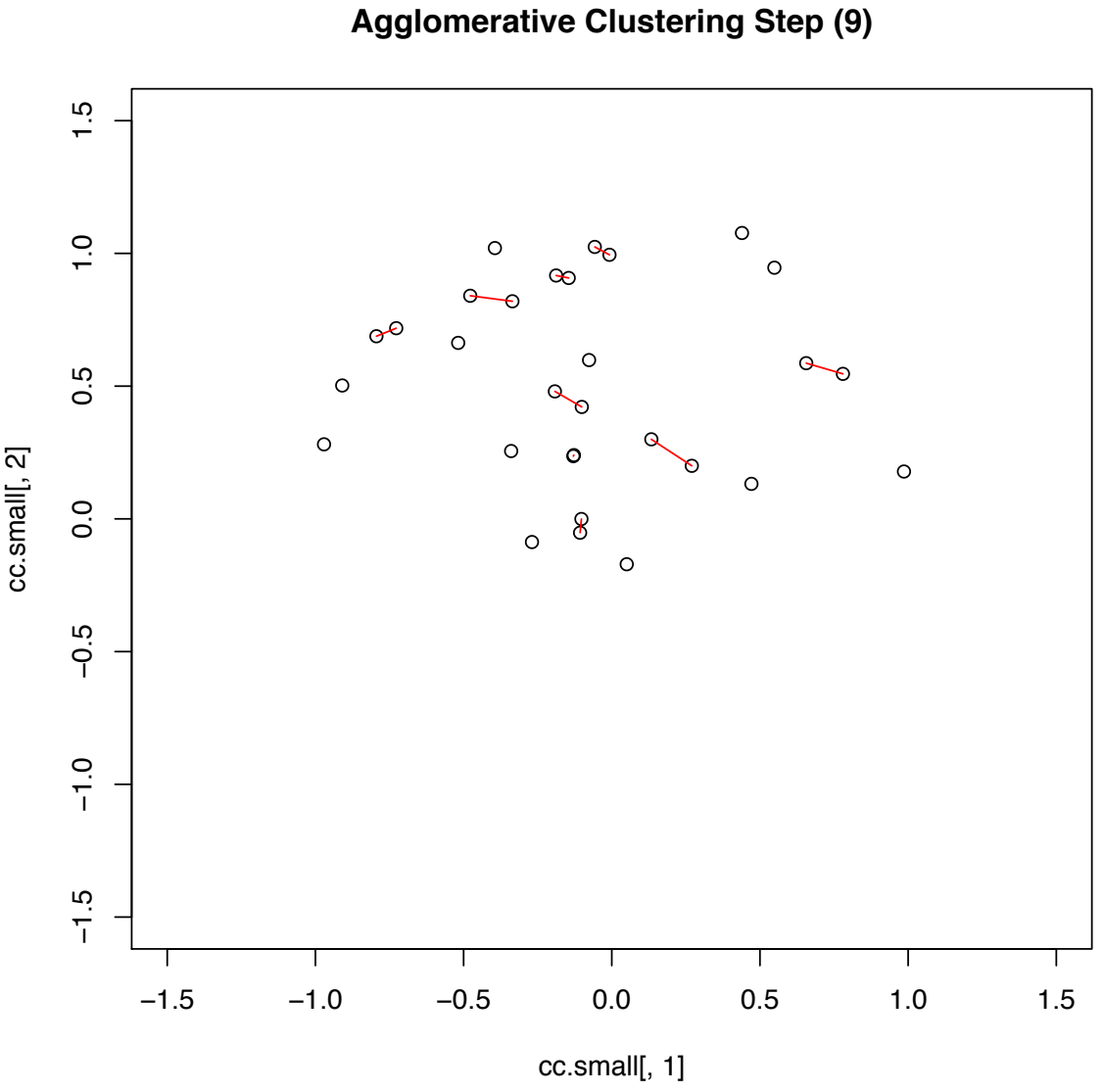
Agglomerative Clustering



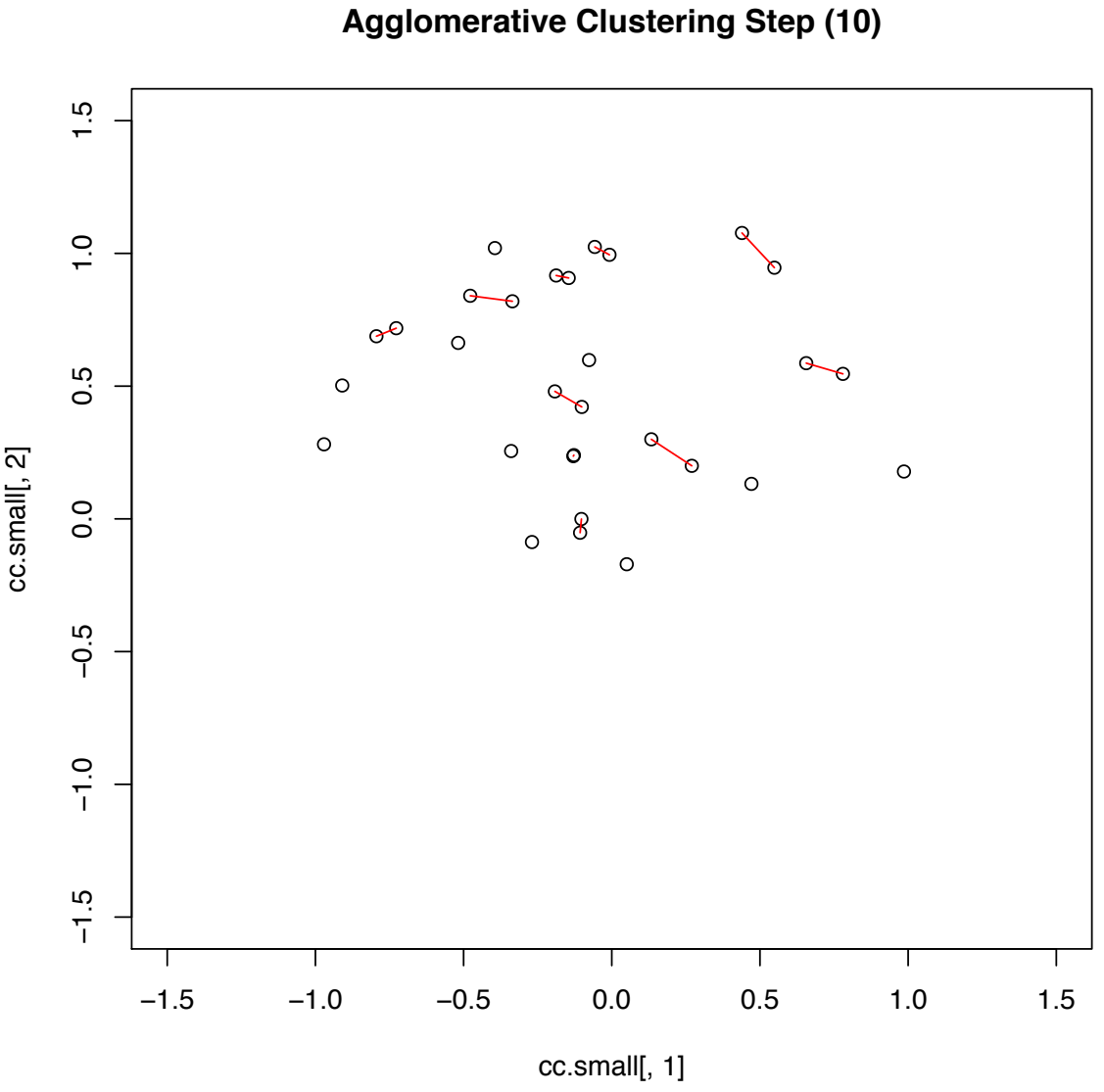
Agglomerative Clustering



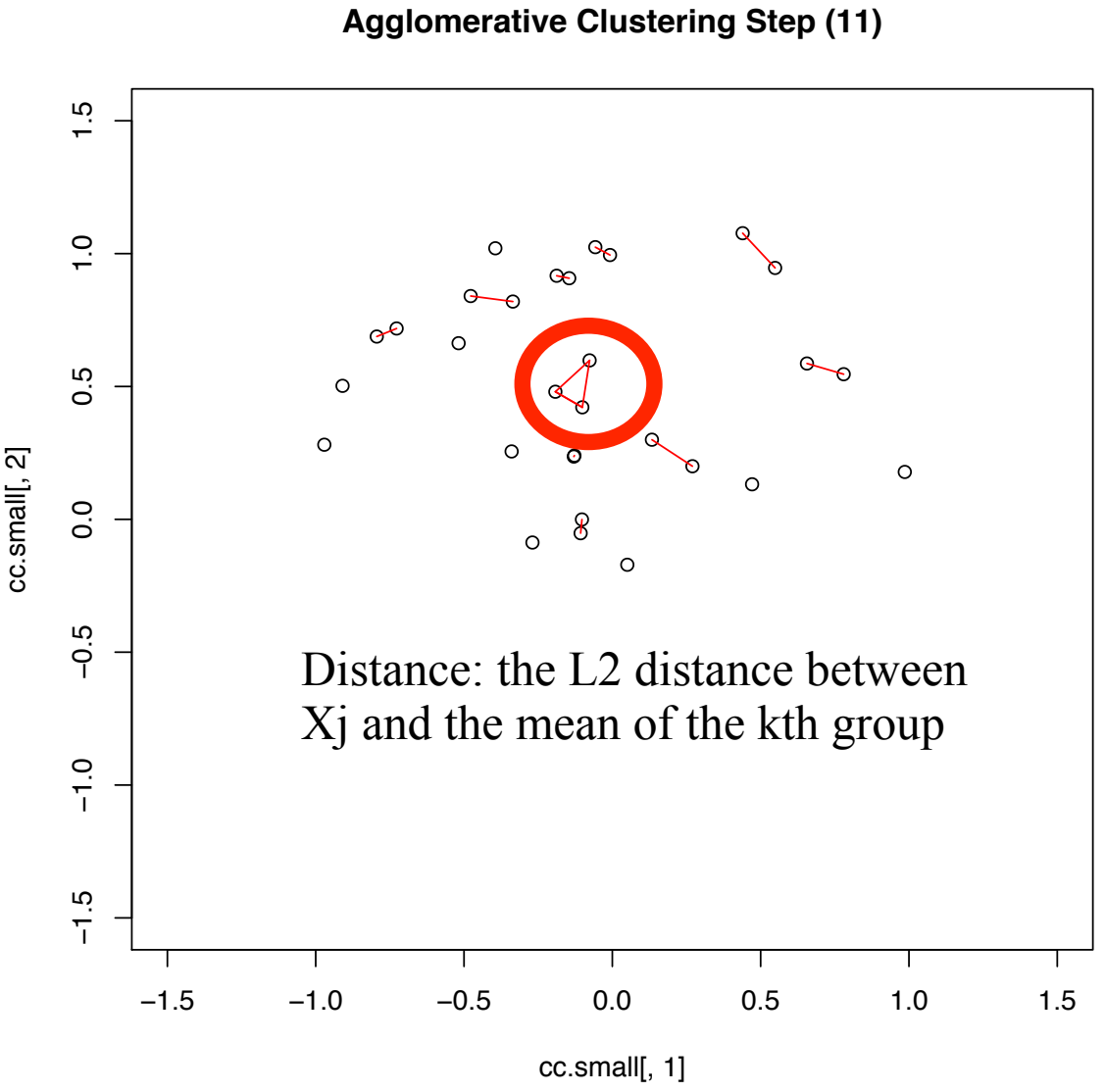
Agglomerative Clustering



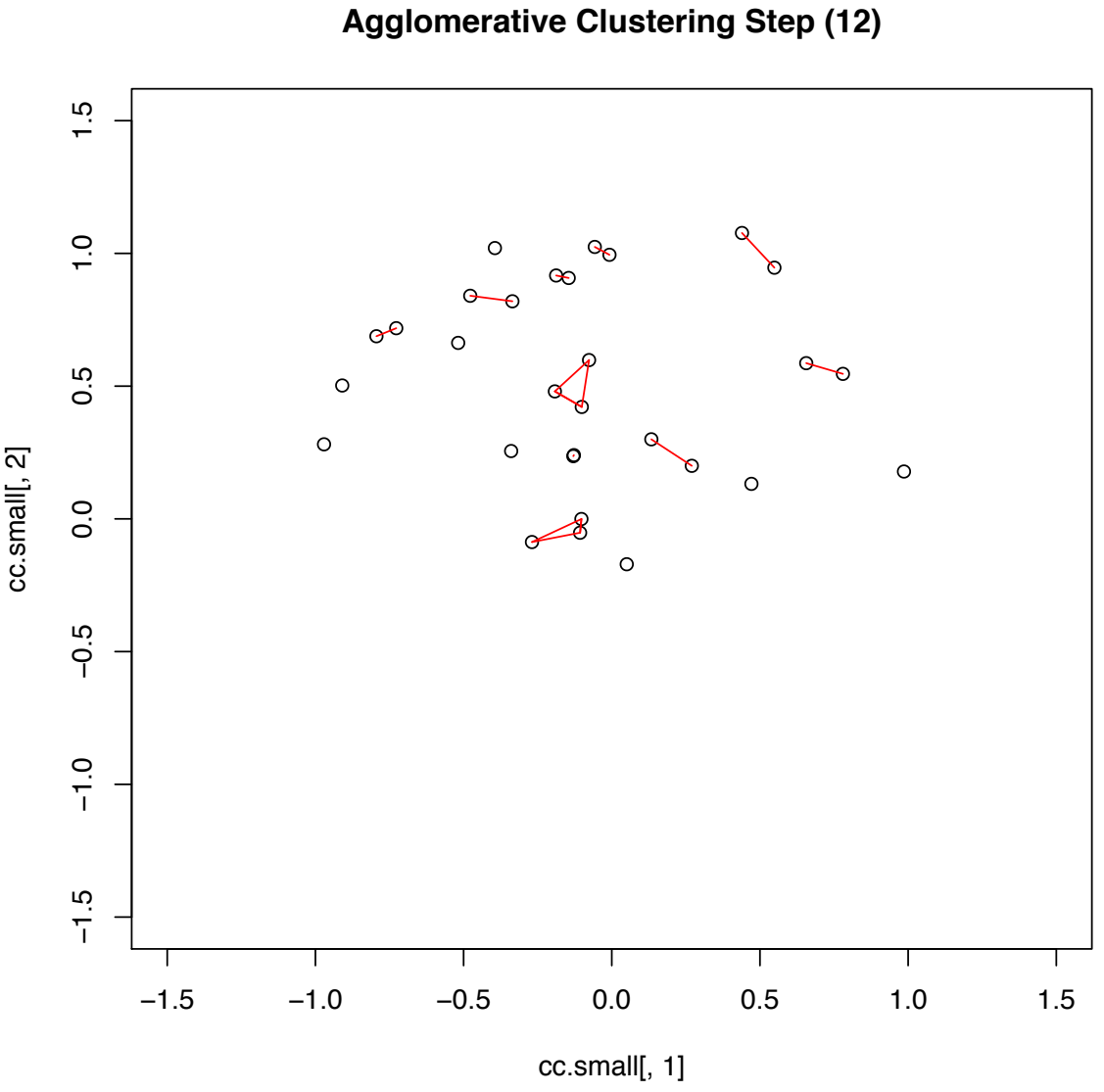
Agglomerative Clustering



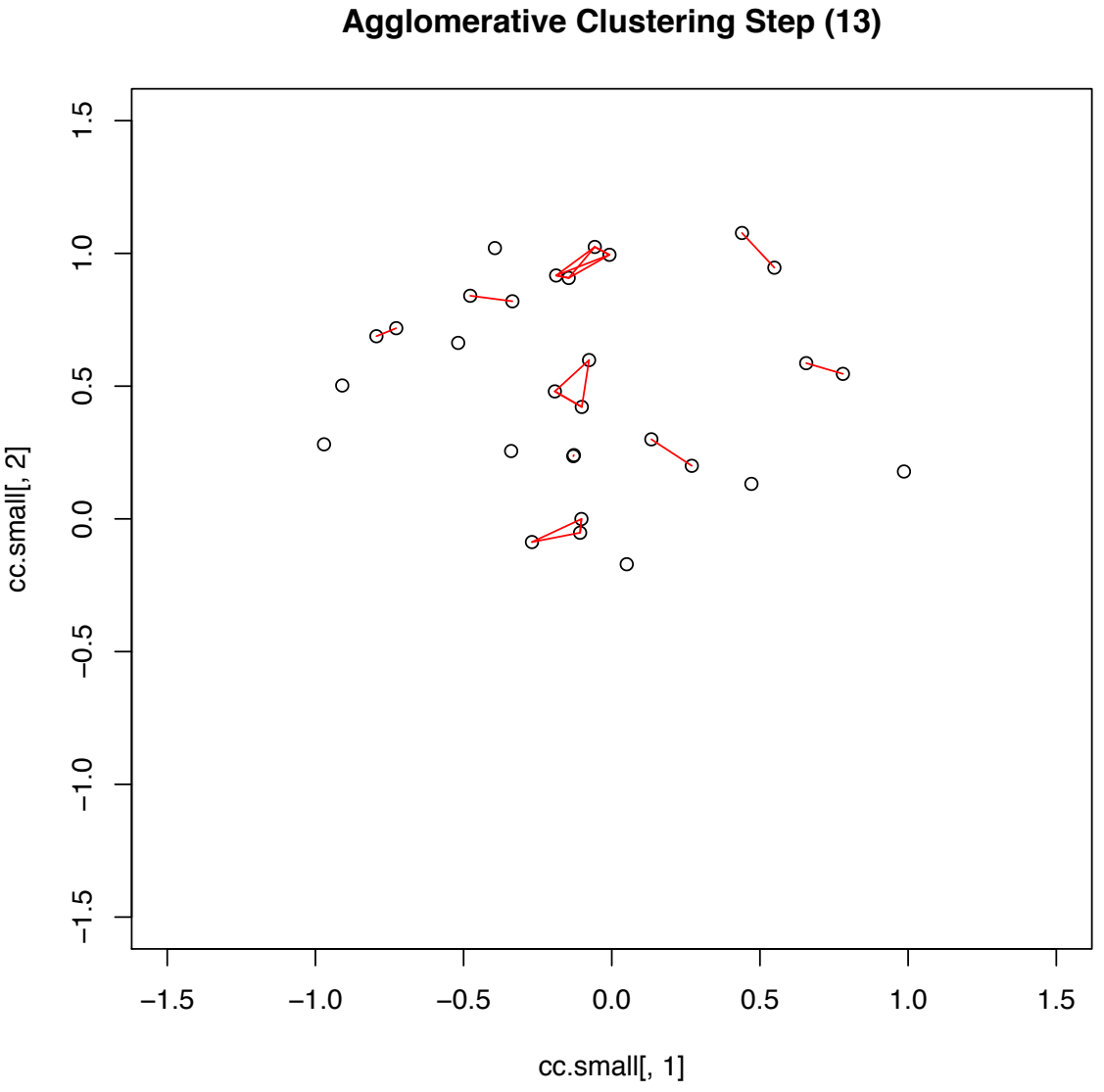
Agglomerative Clustering



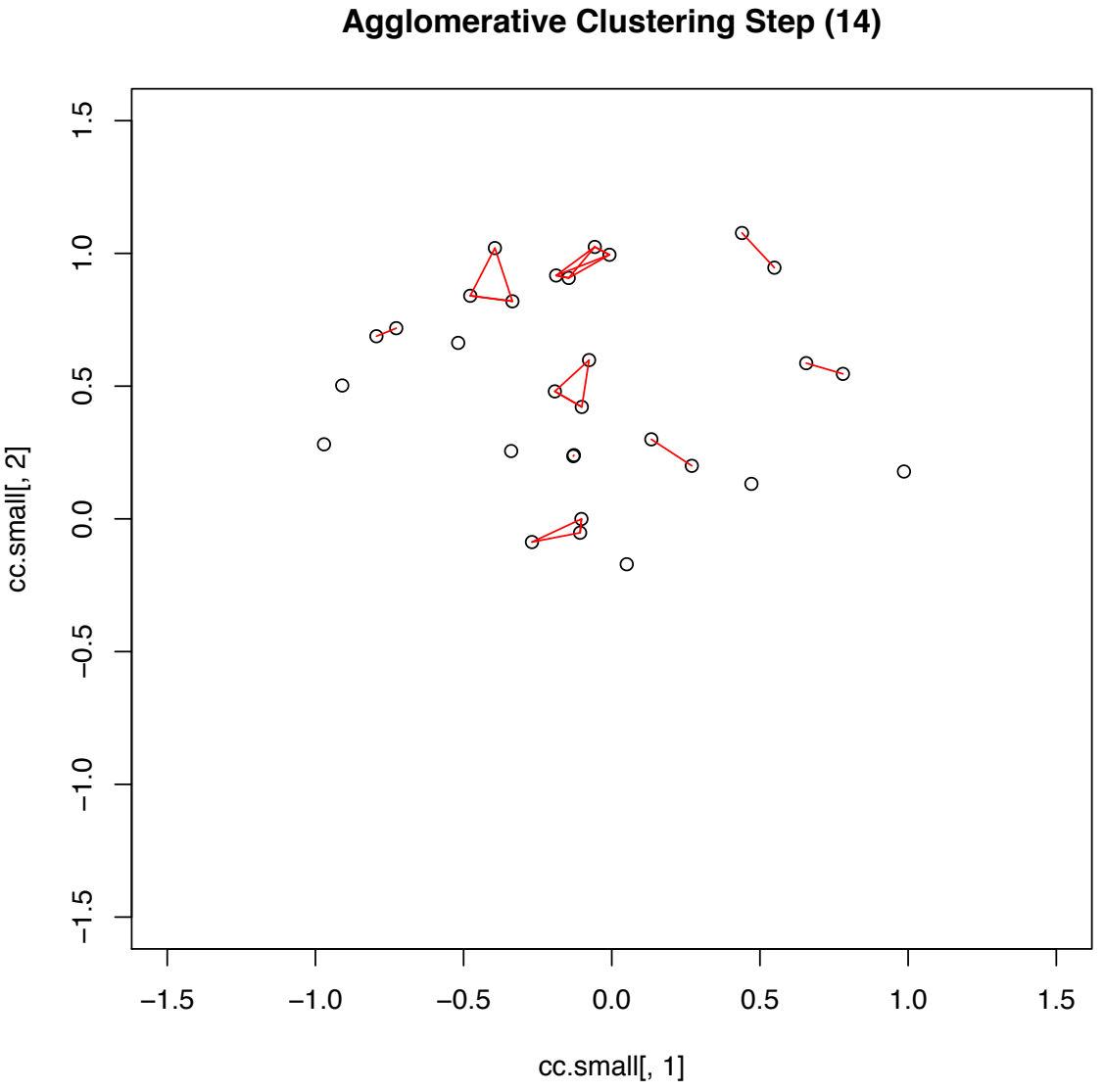
Agglomerative Clustering



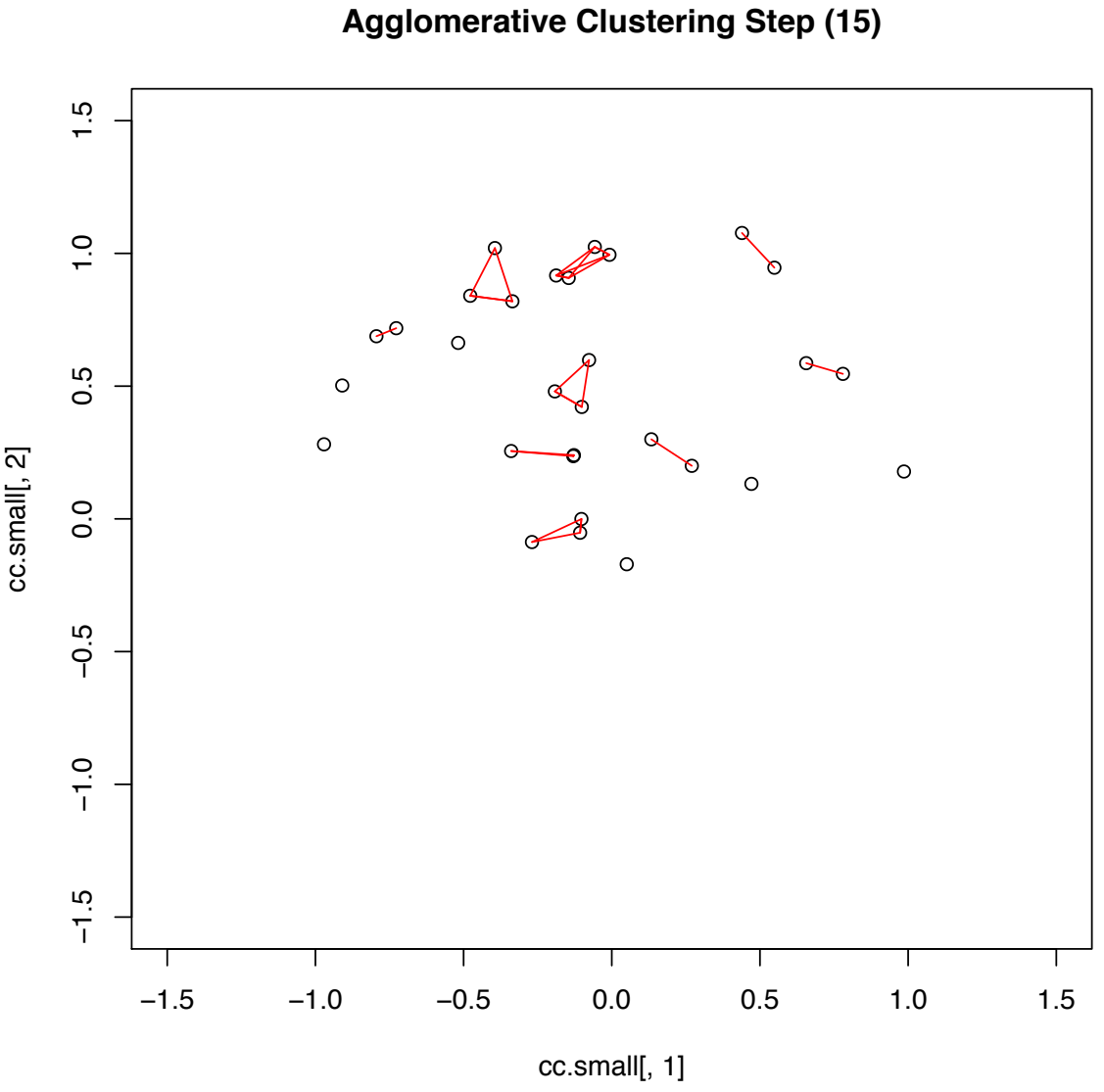
Agglomerative Clustering



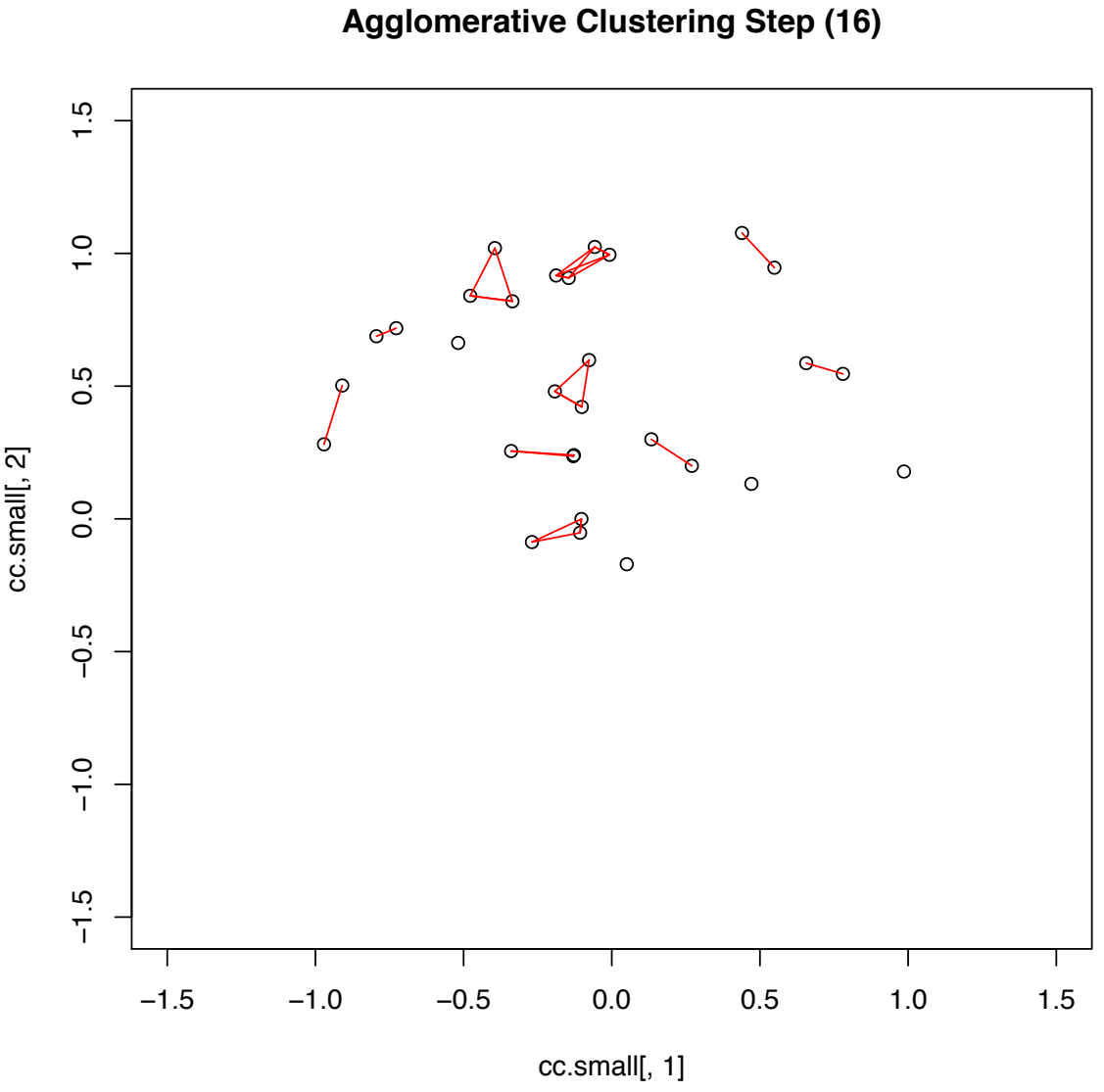
Agglomerative Clustering



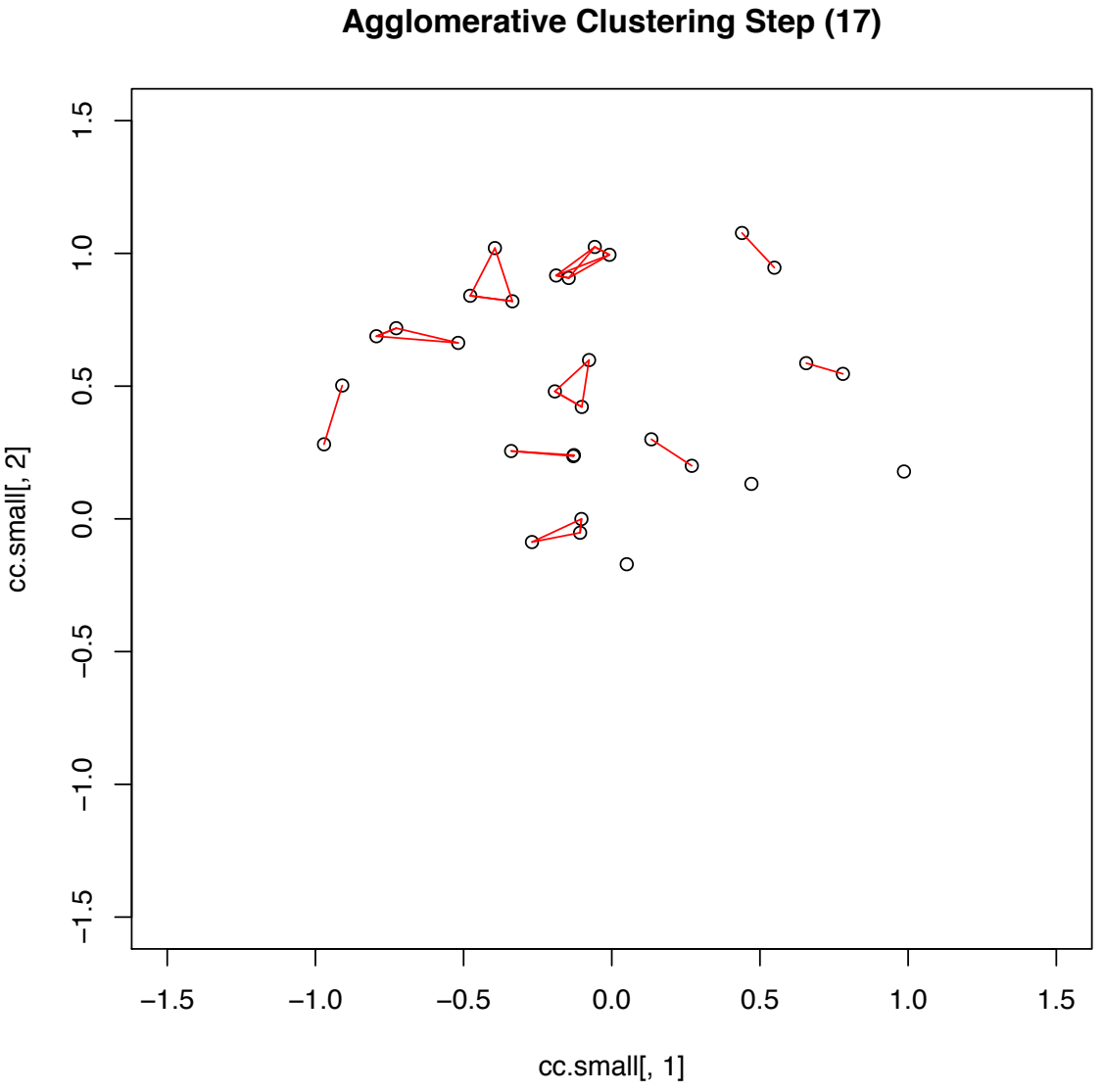
Agglomerative Clustering



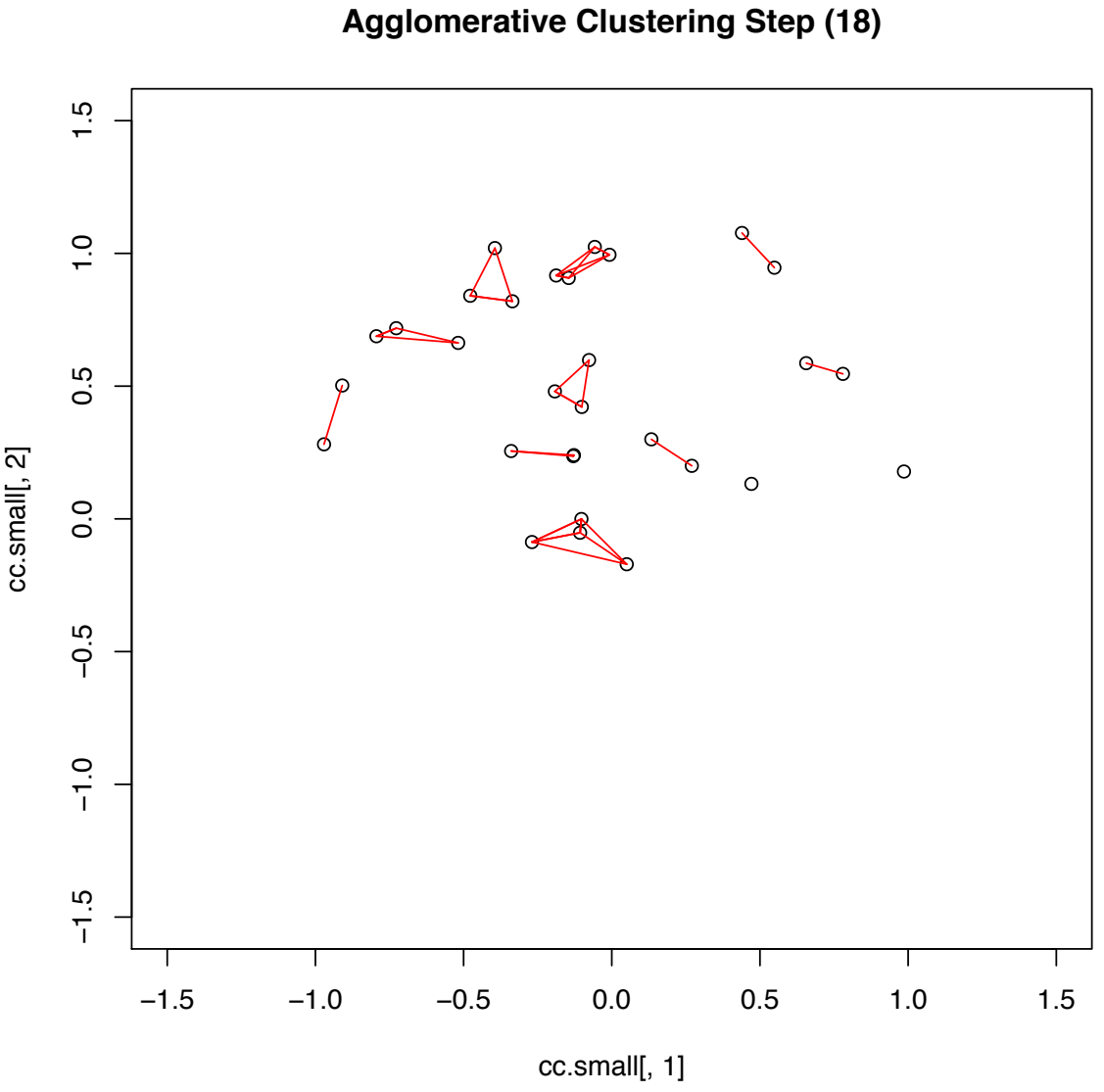
Agglomerative Clustering



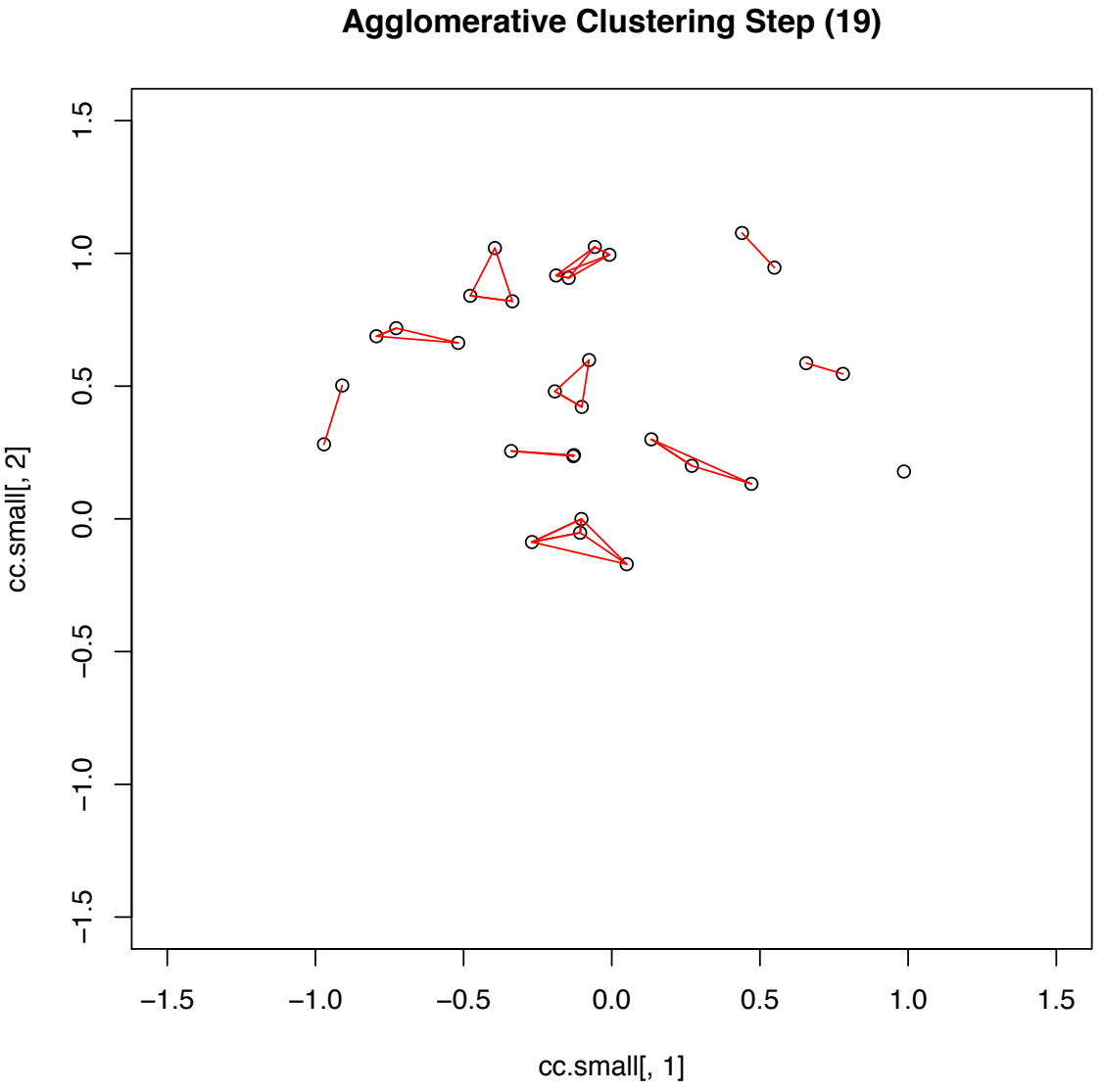
Agglomerative Clustering



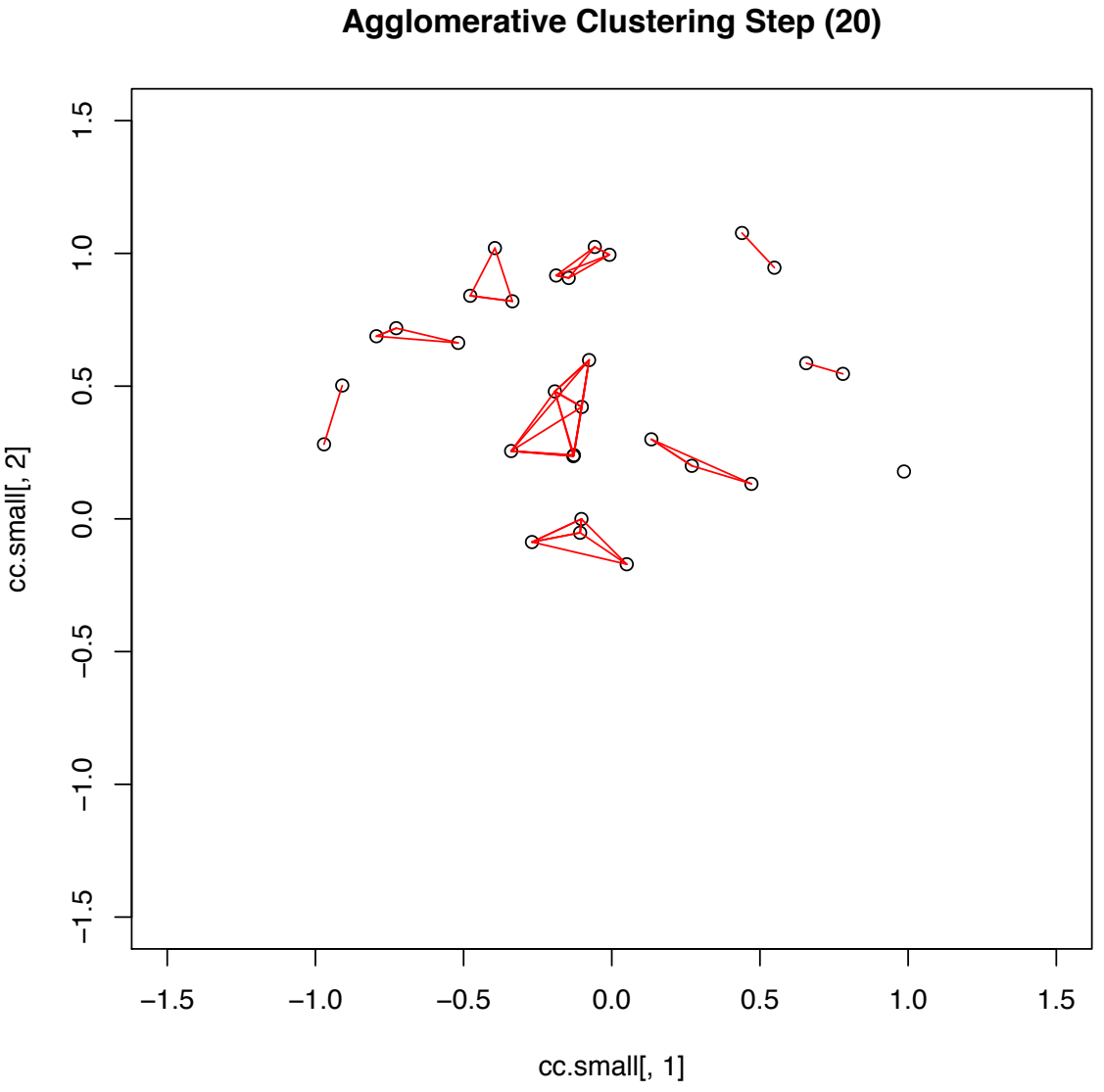
Agglomerative Clustering



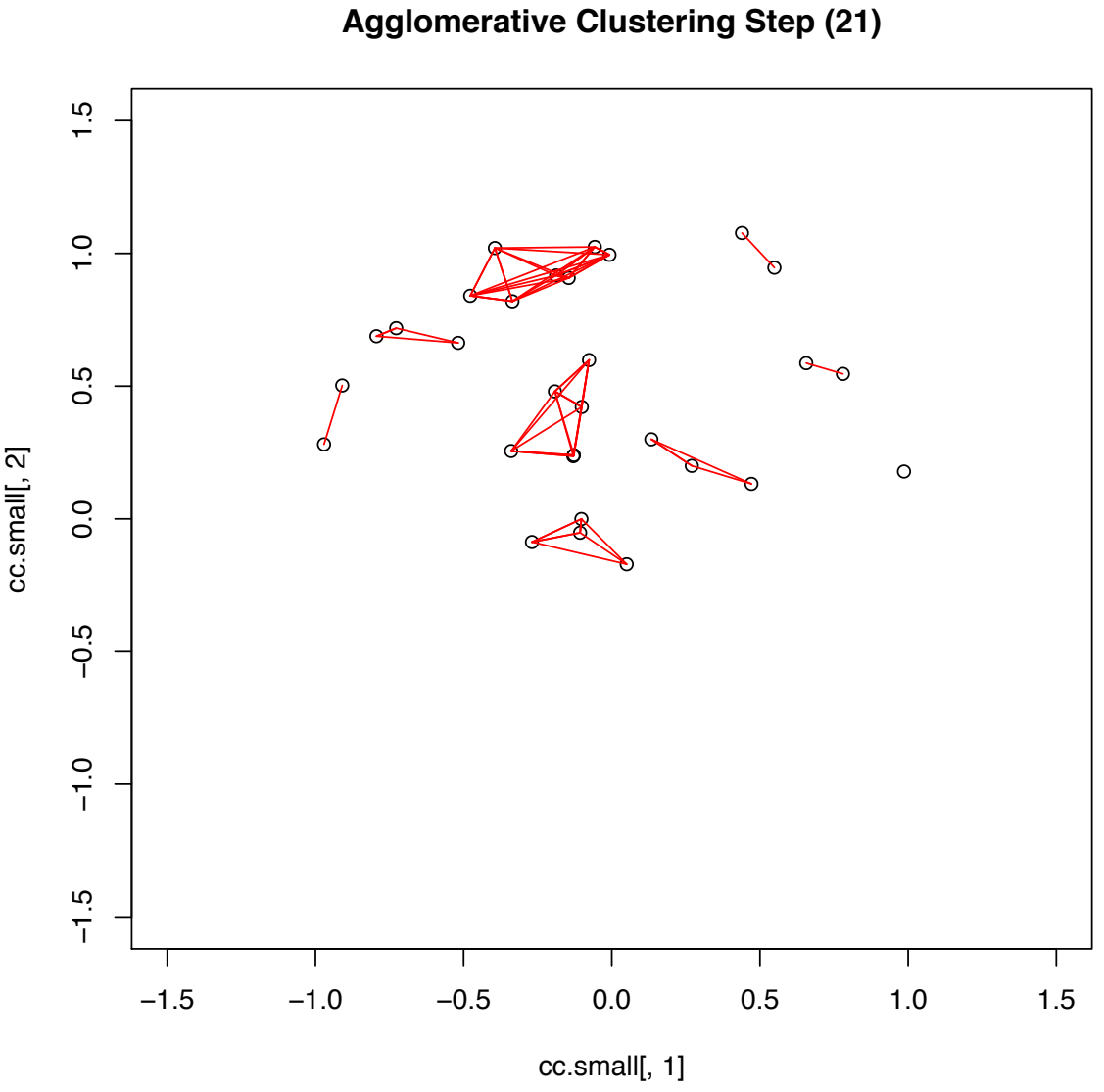
Agglomerative Clustering



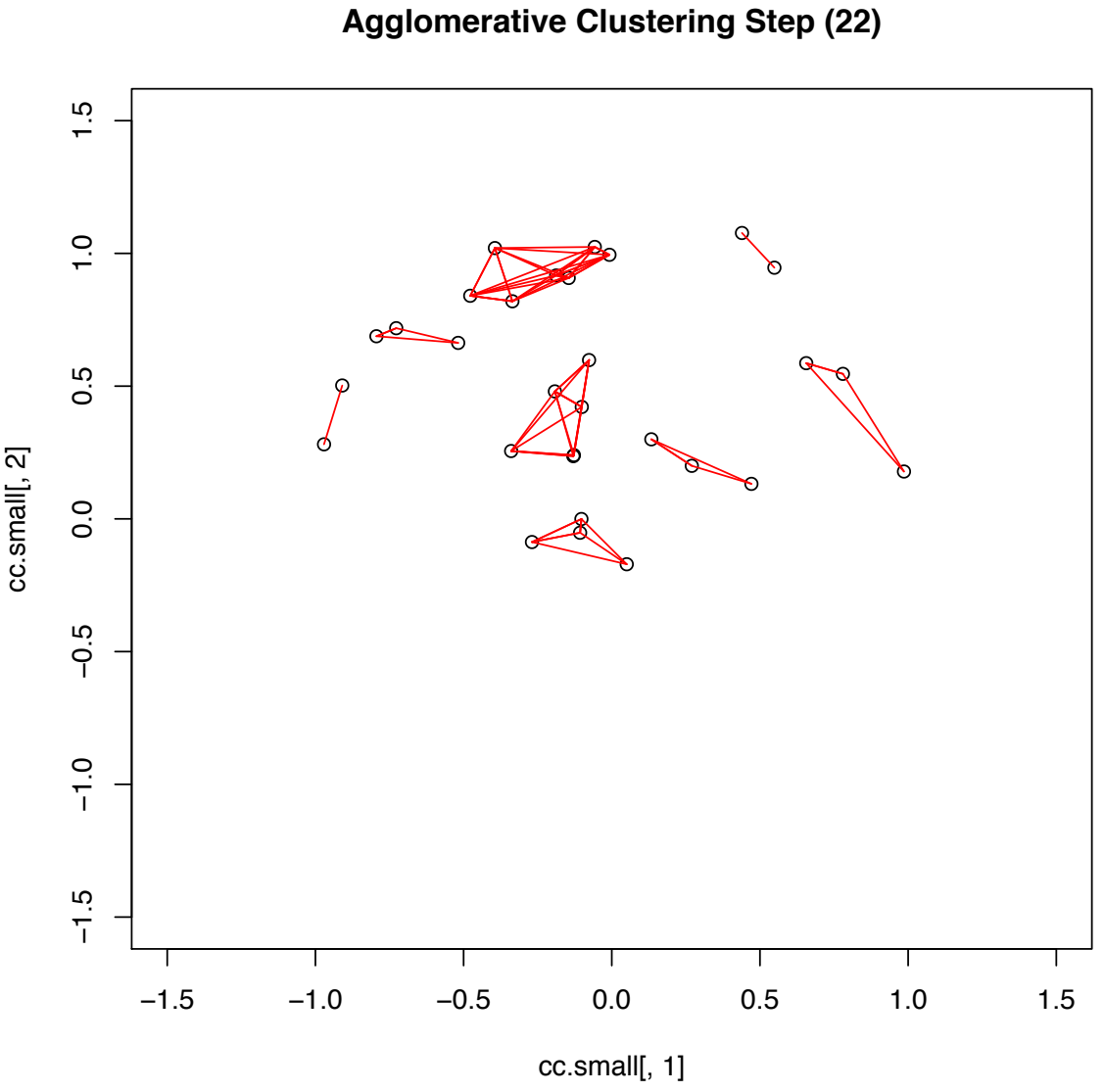
Agglomerative Clustering



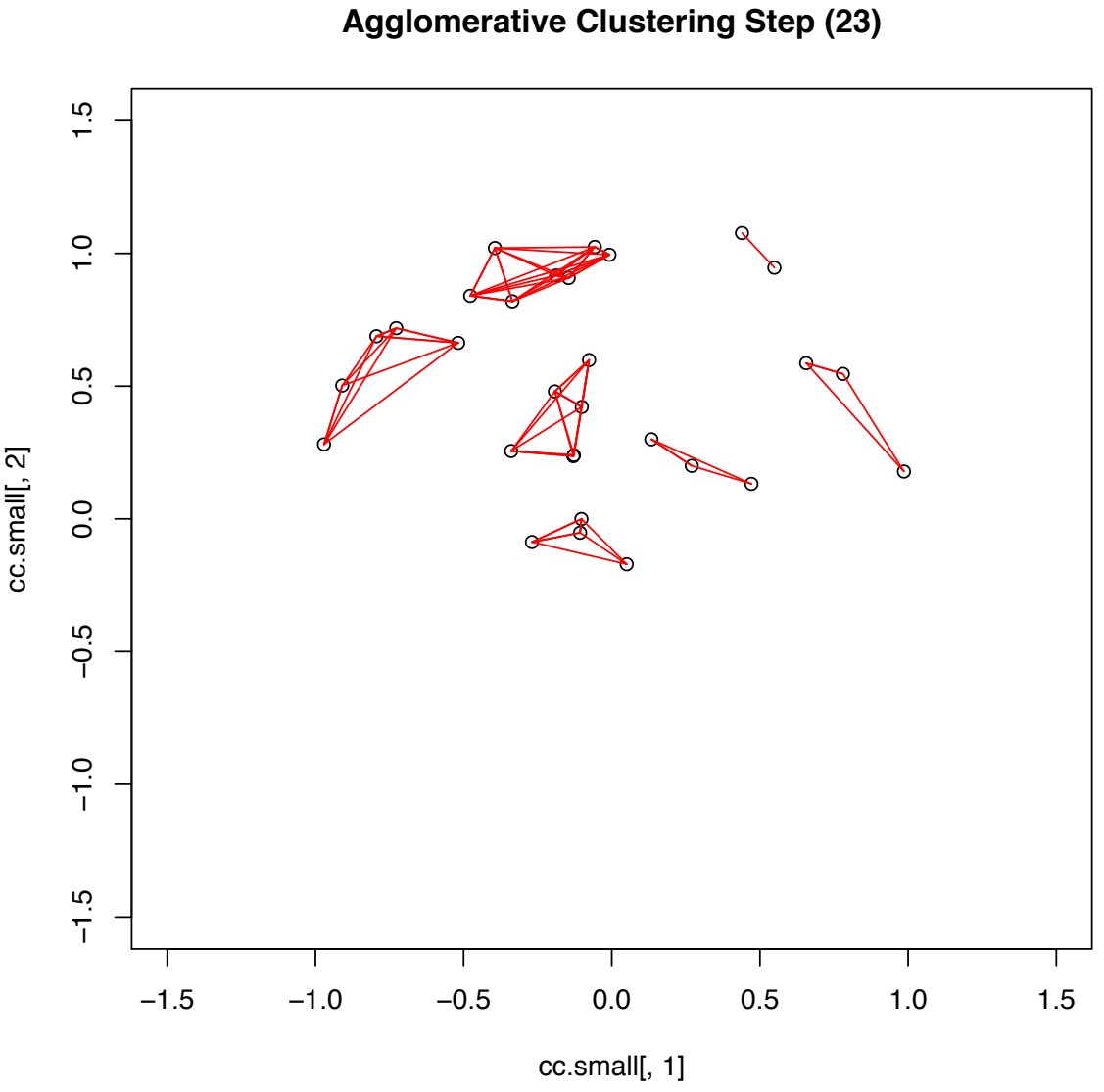
Agglomerative Clustering



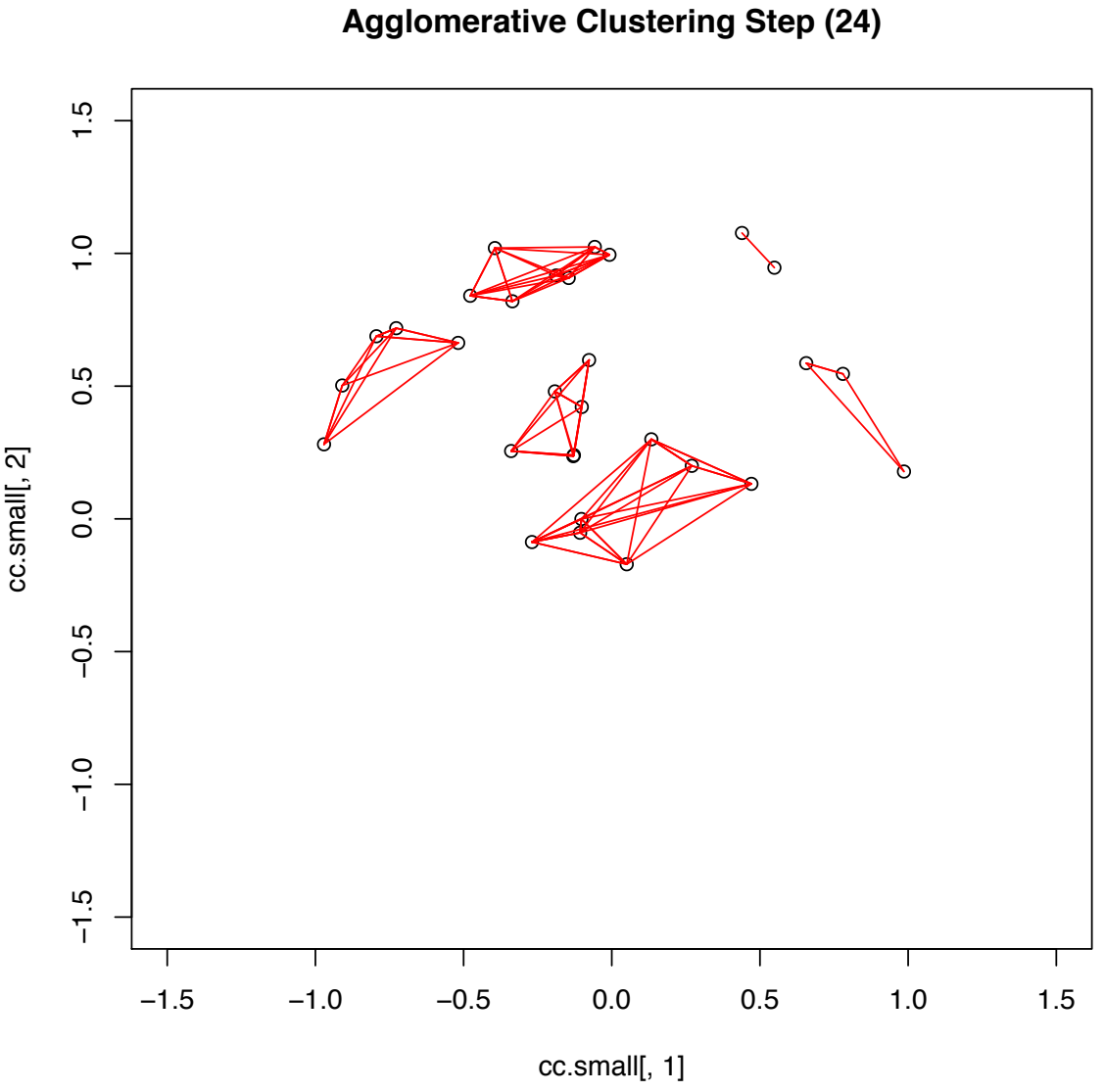
Agglomerative Clustering



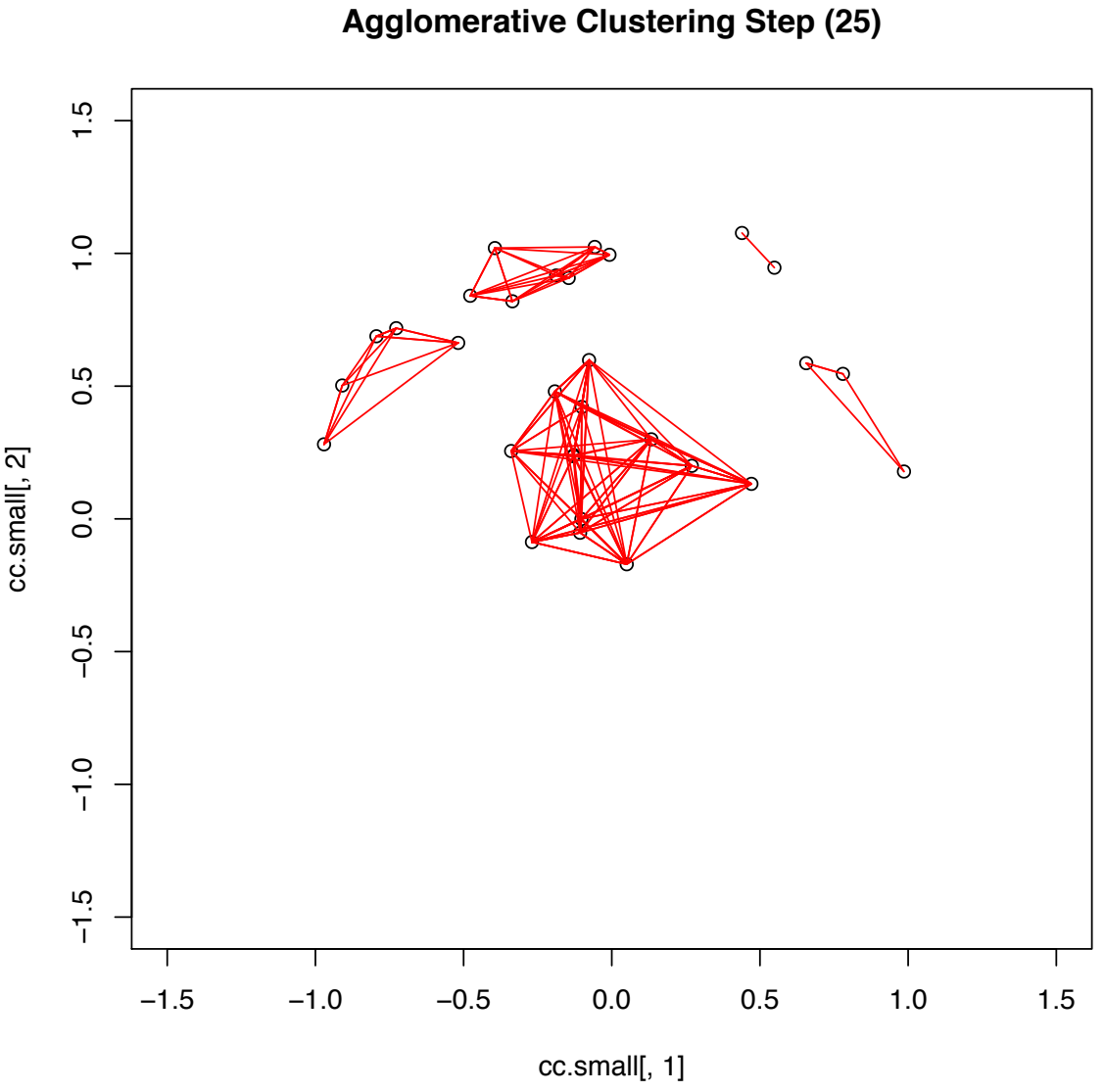
Agglomerative Clustering



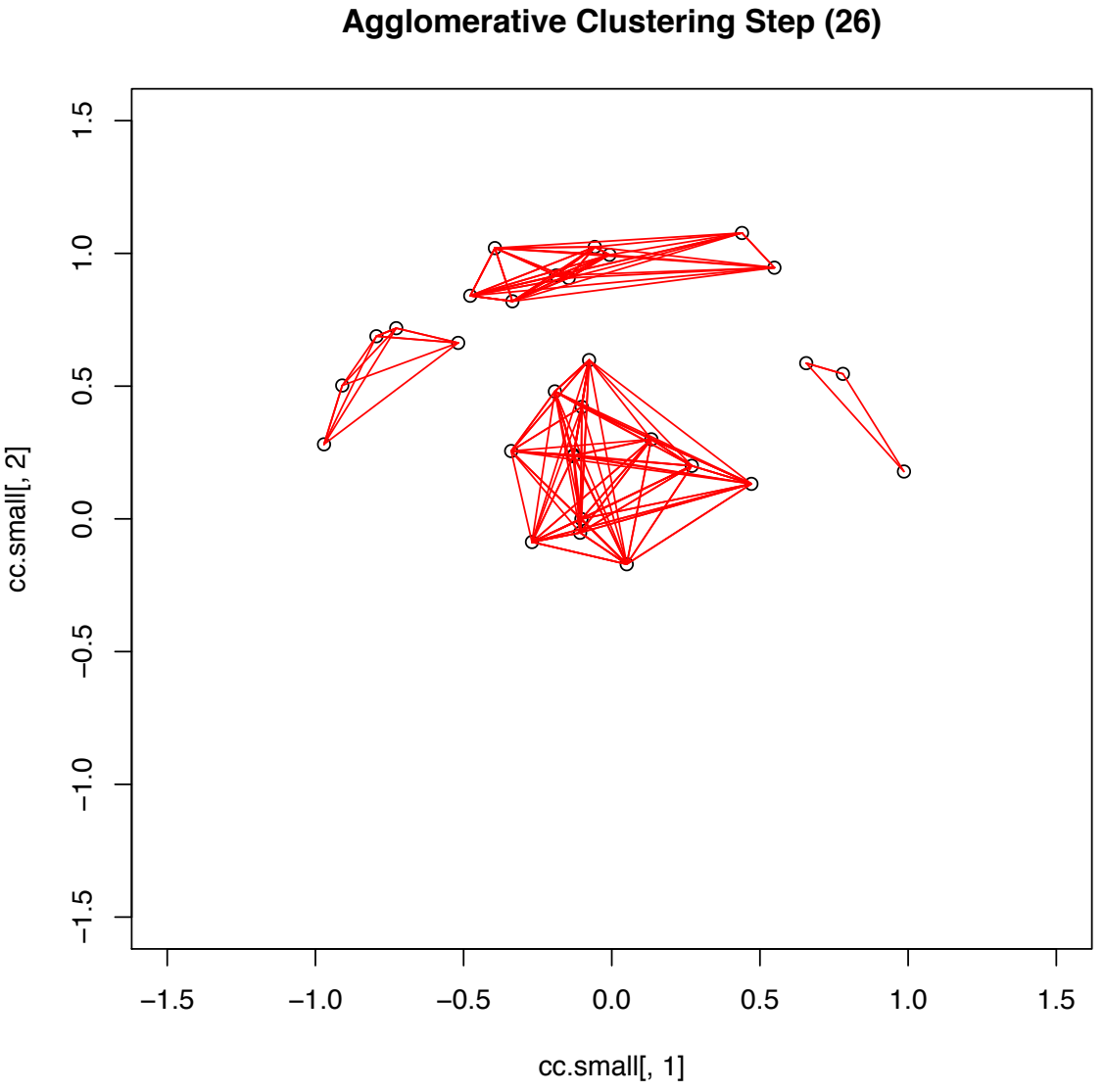
Agglomerative Clustering



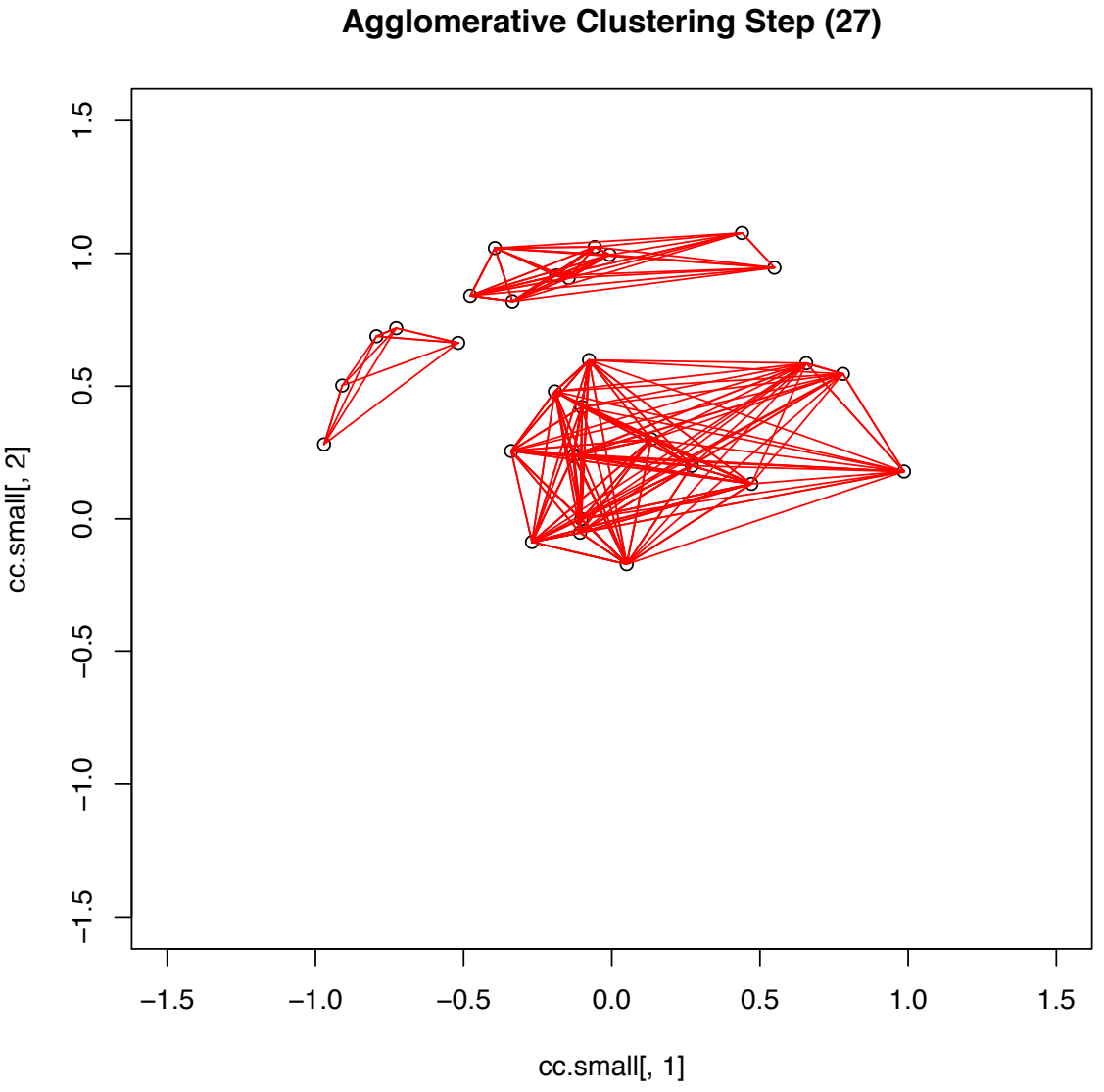
Agglomerative Clustering



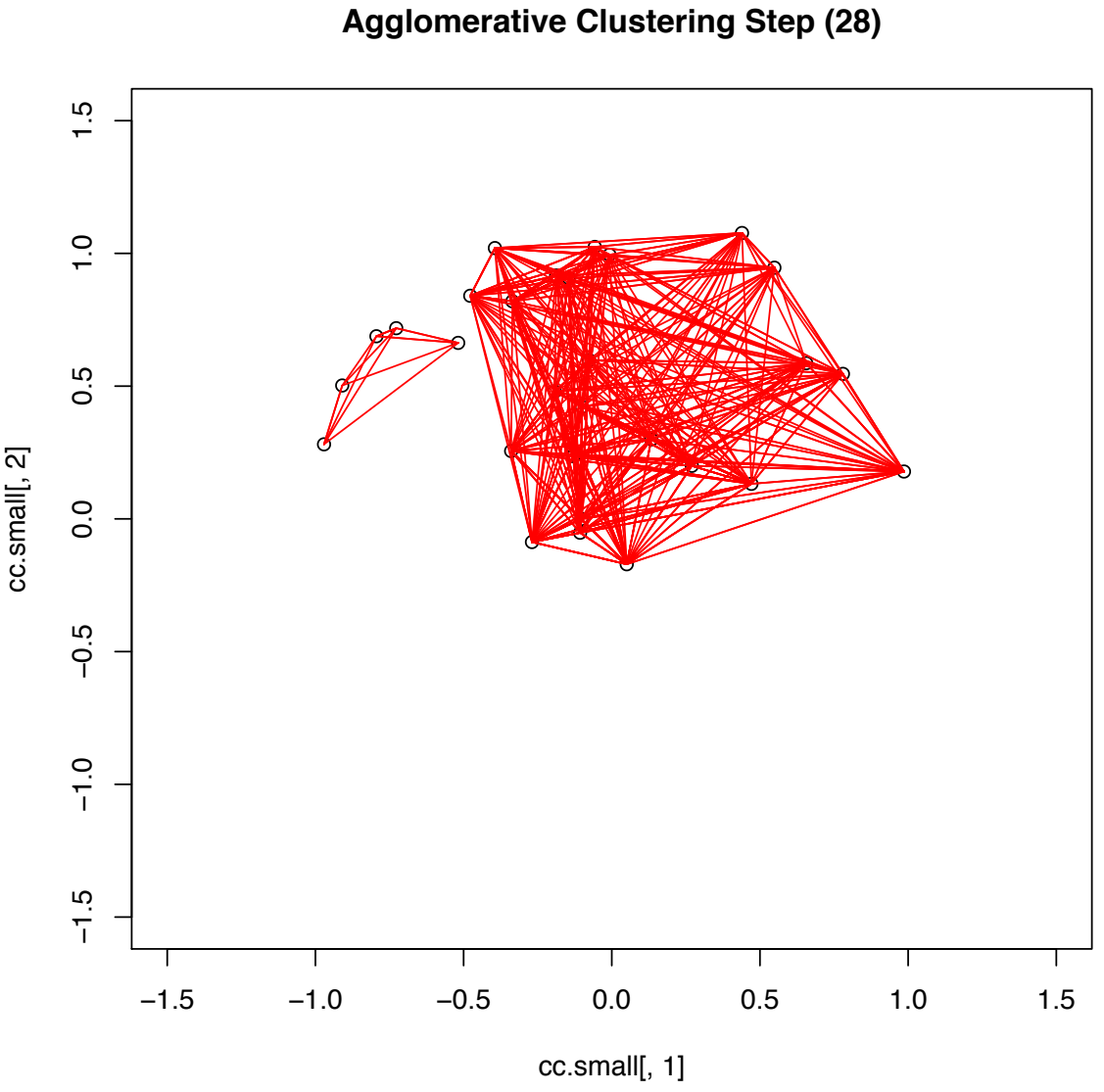
Agglomerative Clustering



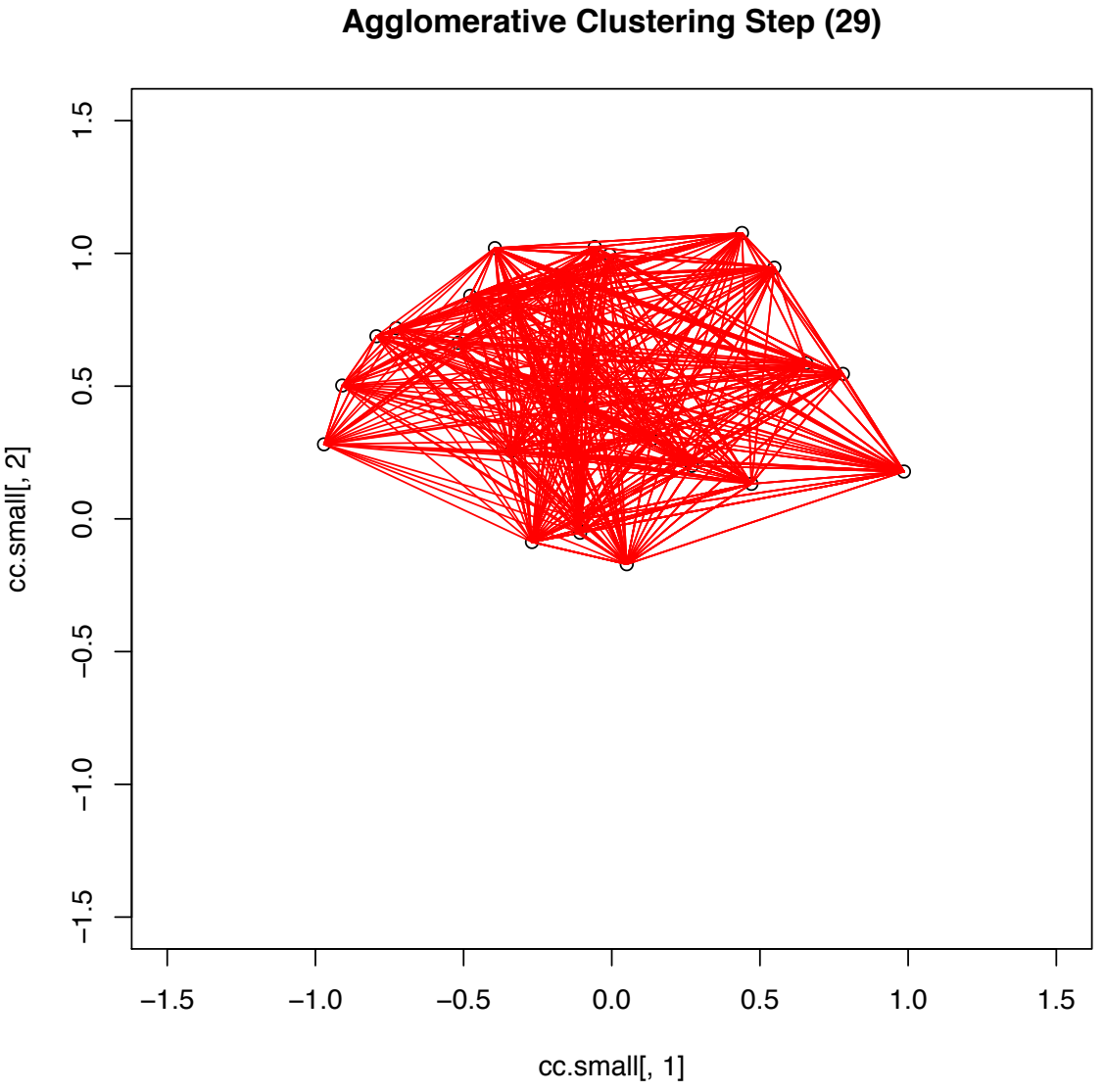
Agglomerative Clustering



Agglomerative Clustering

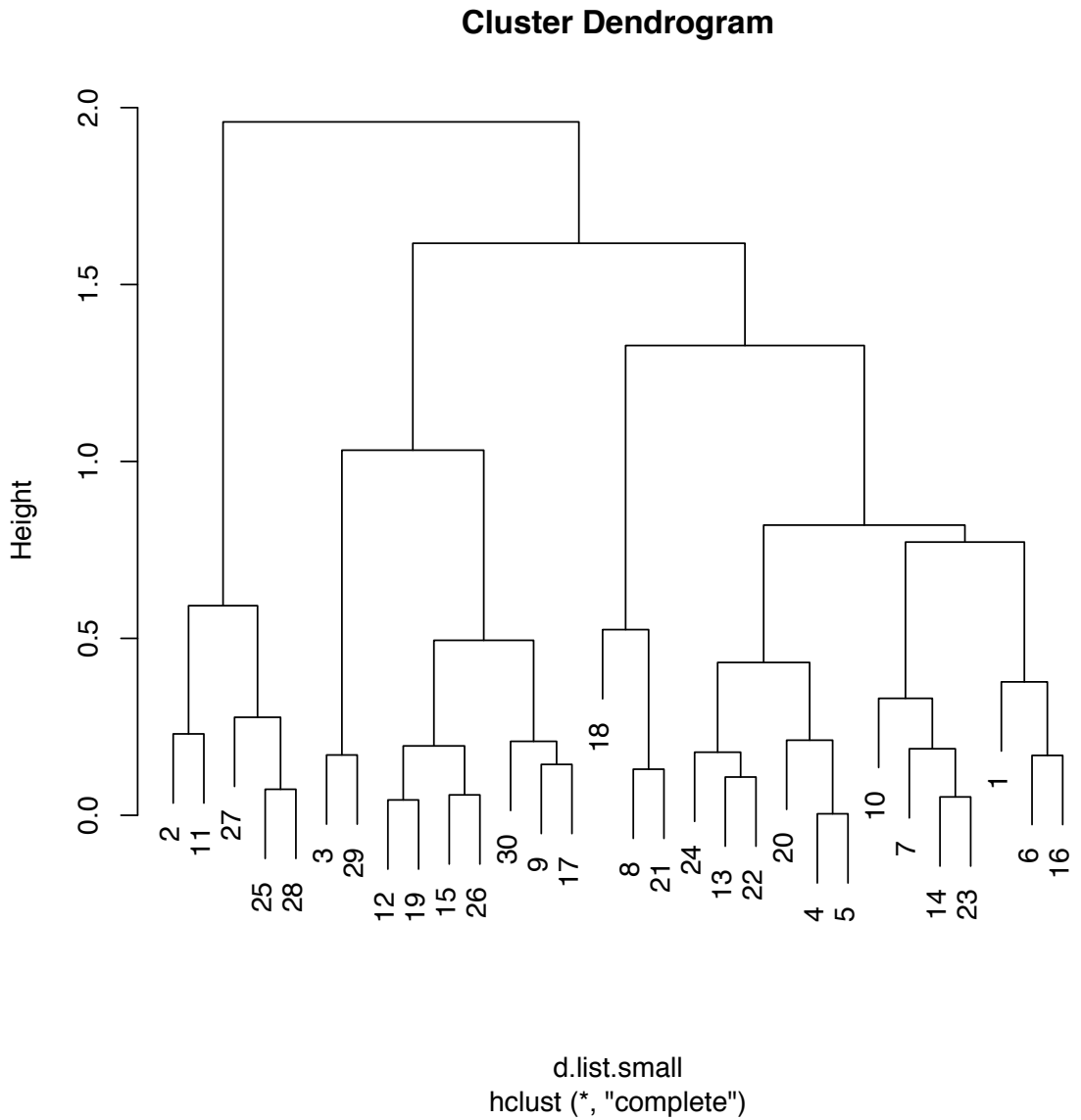


Agglomerative Clustering



Agglomerative Clustering

This can be viewed as a tree:

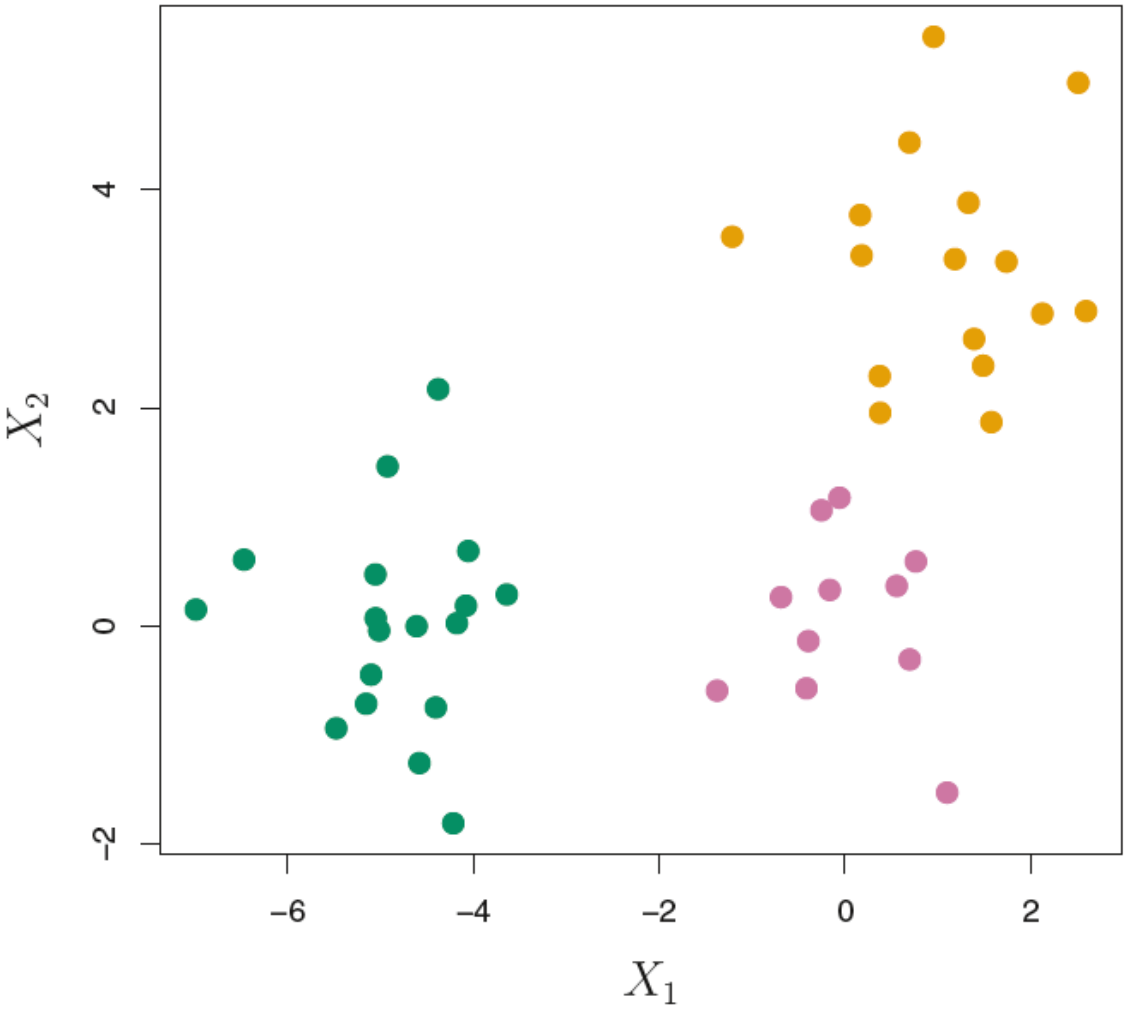


Agglomerative Clustering

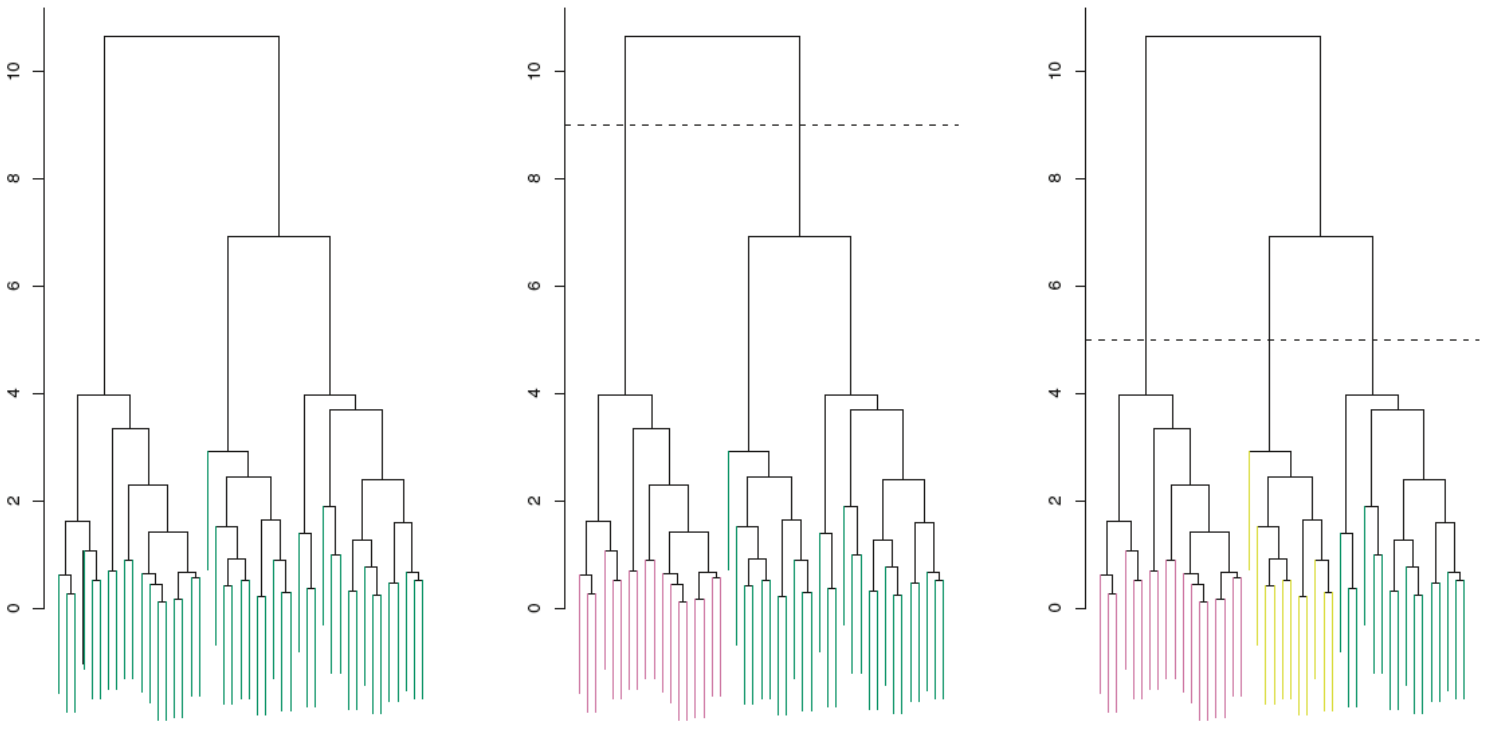
Representing clustering with a tree:

- each level represents a segmentation of the data
- the tree represents a sequence of clusterings
- height represents the negative similarity between merged groups
- user must choose best grouping
- great for summarizing algorithm and data

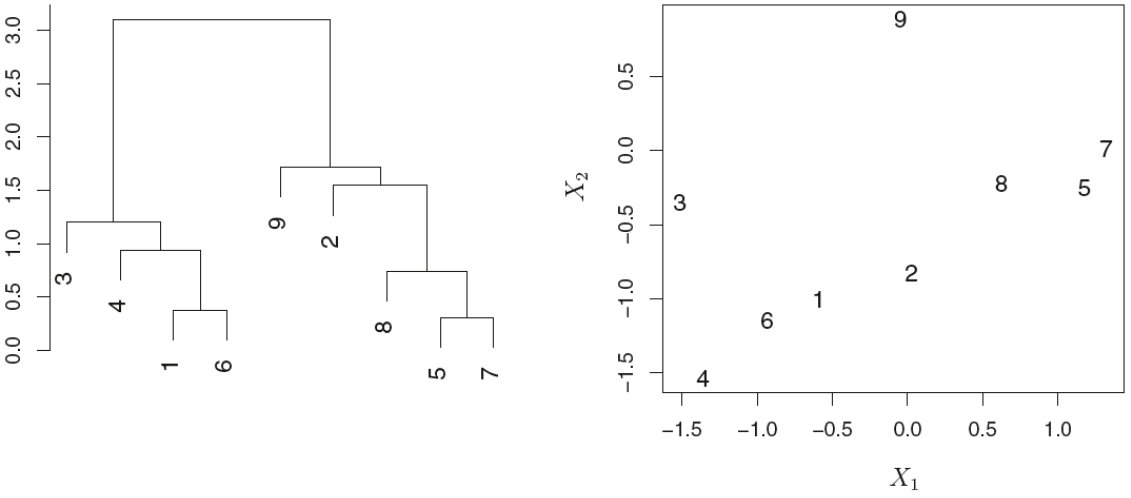
Agglomerative Clustering: data simulated from a 3-class model



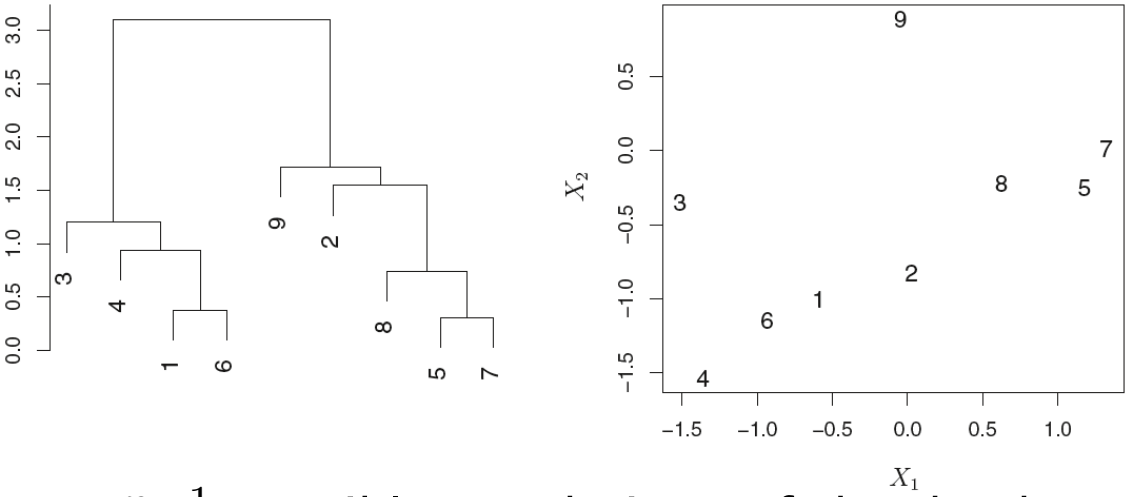
Dendogram obtained from hierarchical clustering



Agglomerative Clustering: interpretation of a dendogram



Agglomerative Clustering: interpretation of a dendogram



- There are 2^{n-1} possible reorderings of the dendrogram, where n is the number of leaves.
- At each of the $n-1$ points where fusions occur, the positions of the two fused branches could be swapped without affecting the meaning of the dendrogram.
- DO NOT draw conclusions about the similarity of two obs based on their proximity along the *horizontal* axis.
- Draw conclusions about the similarity of two observations based on the location on the *vertical* axis where branches containing those two observations first are fused.

Outline

Group Similarity

How do we define similarity between groups?

The most popular choices:

- *Single-linkage*: the distance between the closest pair

$$d_{SL}(G, H) = \min_{i \in G, j \in H} d(x_i, x_j)$$

- *Complete-linkage*: the distance of the furthest pair

$$d_{CL}(G, H) = \max_{i \in G, j \in H} d(x_i, x_j)$$

- *Group-average*: the average distance between groups

$$d_{GA}(G, H) = \frac{1}{N_G N_H} \sum_{i \in G,\, j \in H} d(x_i, x_j)$$

- *Metroid:* the distance between the means or metroids of groups

$$d_{ME}(G, H) = d(\mu_G, \mu_H)$$

Group Similarity

- single linkage can produce "chaining", where a few close observations can cause early merger of two groups
- complete linkage can cause groups not to merge if there are a few distant observations
- group averaging is a compromise, although it can require scaling of the data/similarities
- Metroids are similar to k-means. Major drawback that an *inversion* can occur: two clusters are fused at a height **below** either of the individual clusters in the dendrogram. This can lead to difficulties in visualization as well as in interpretation of the dendrogram.

Agglomerative Clustering: interpretation of a dendogram

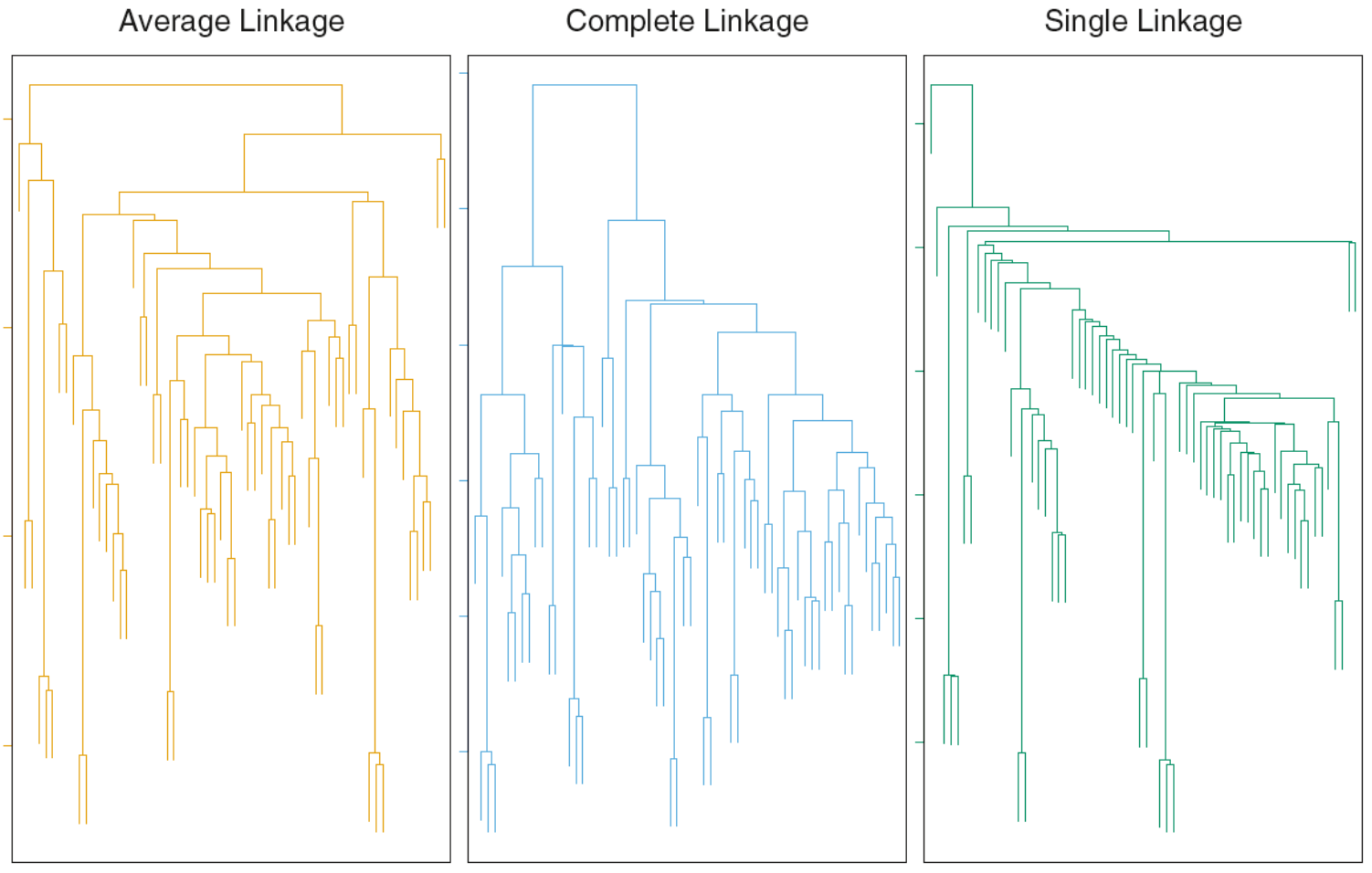
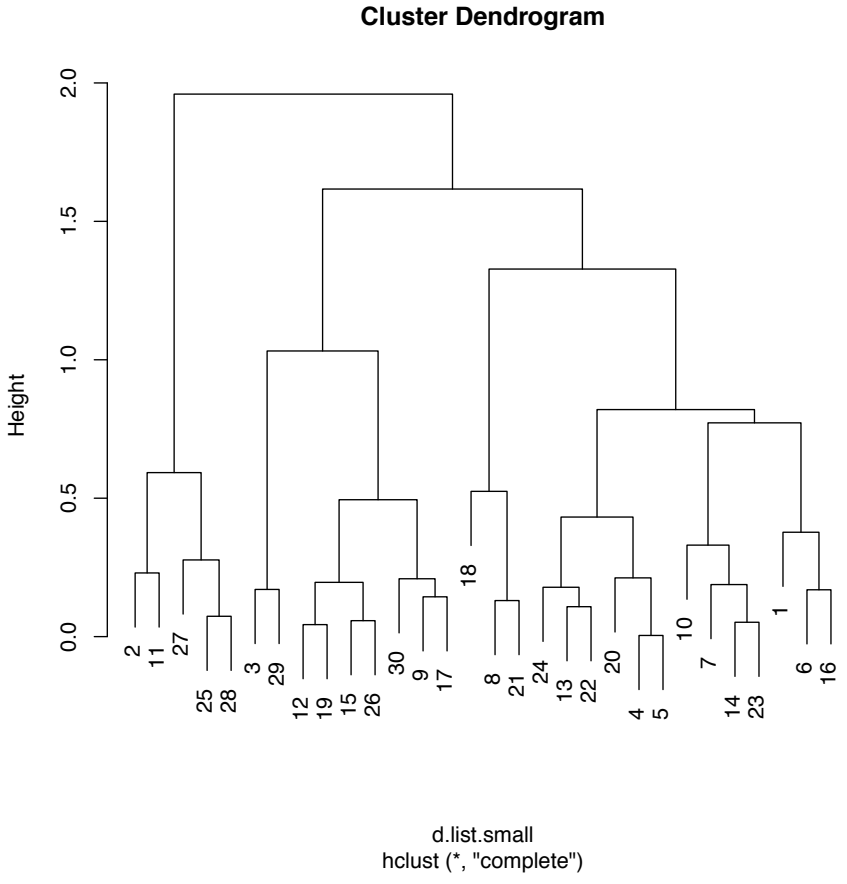
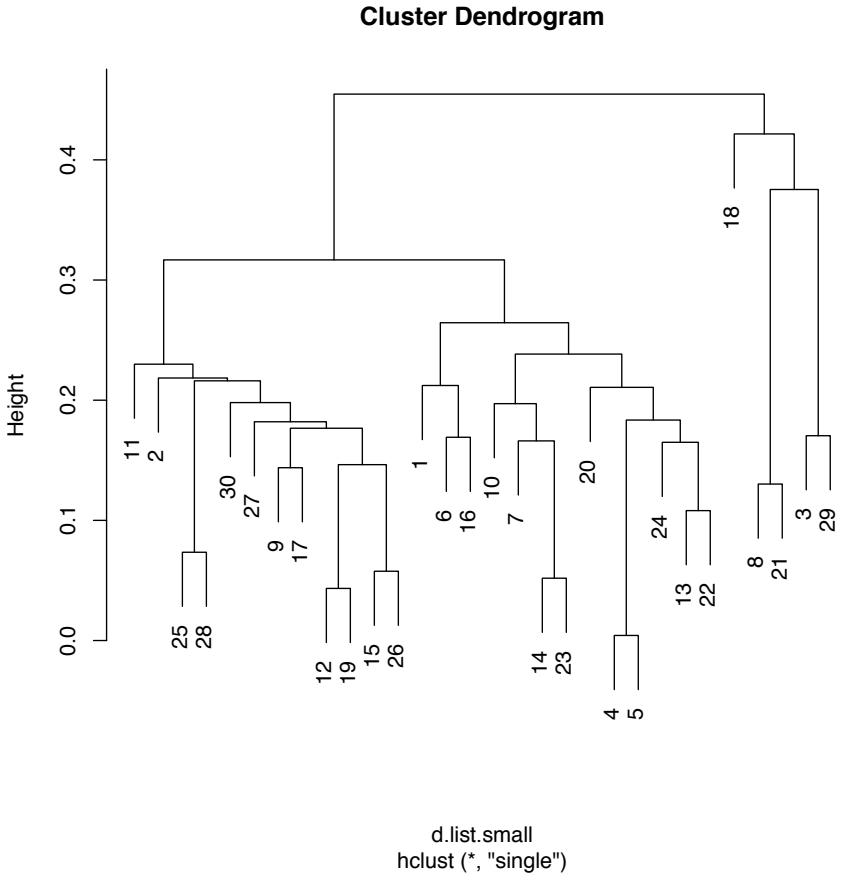


FIGURE 10.12. *Average, complete, and single linkage applied to an example data set. Average and complete linkage tend to yield more balanced clusters.*

Caveats





(Left) Complete linkage dendrogram, (Right) Single linkage

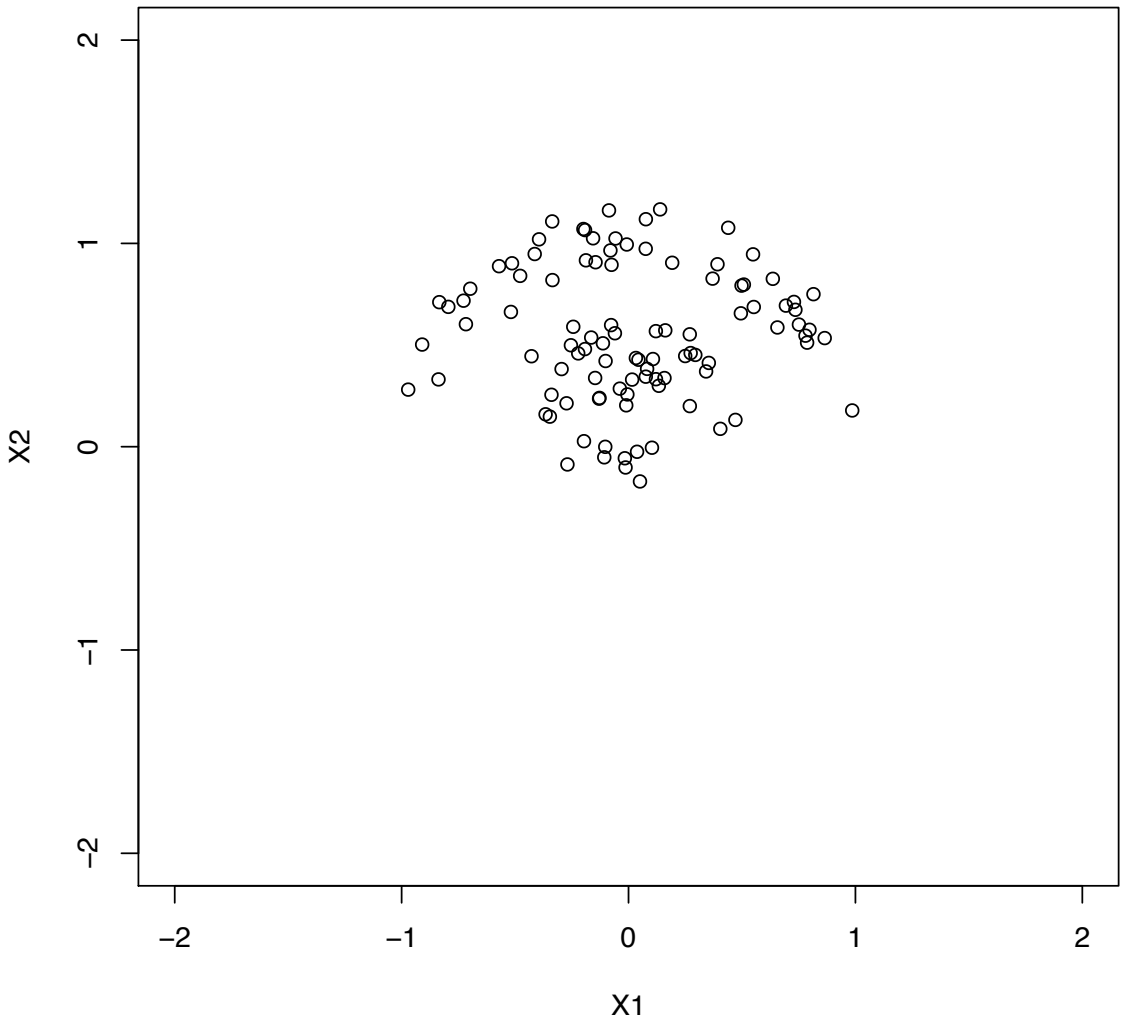
Caveats

Hierarchical clustering has a few issues:

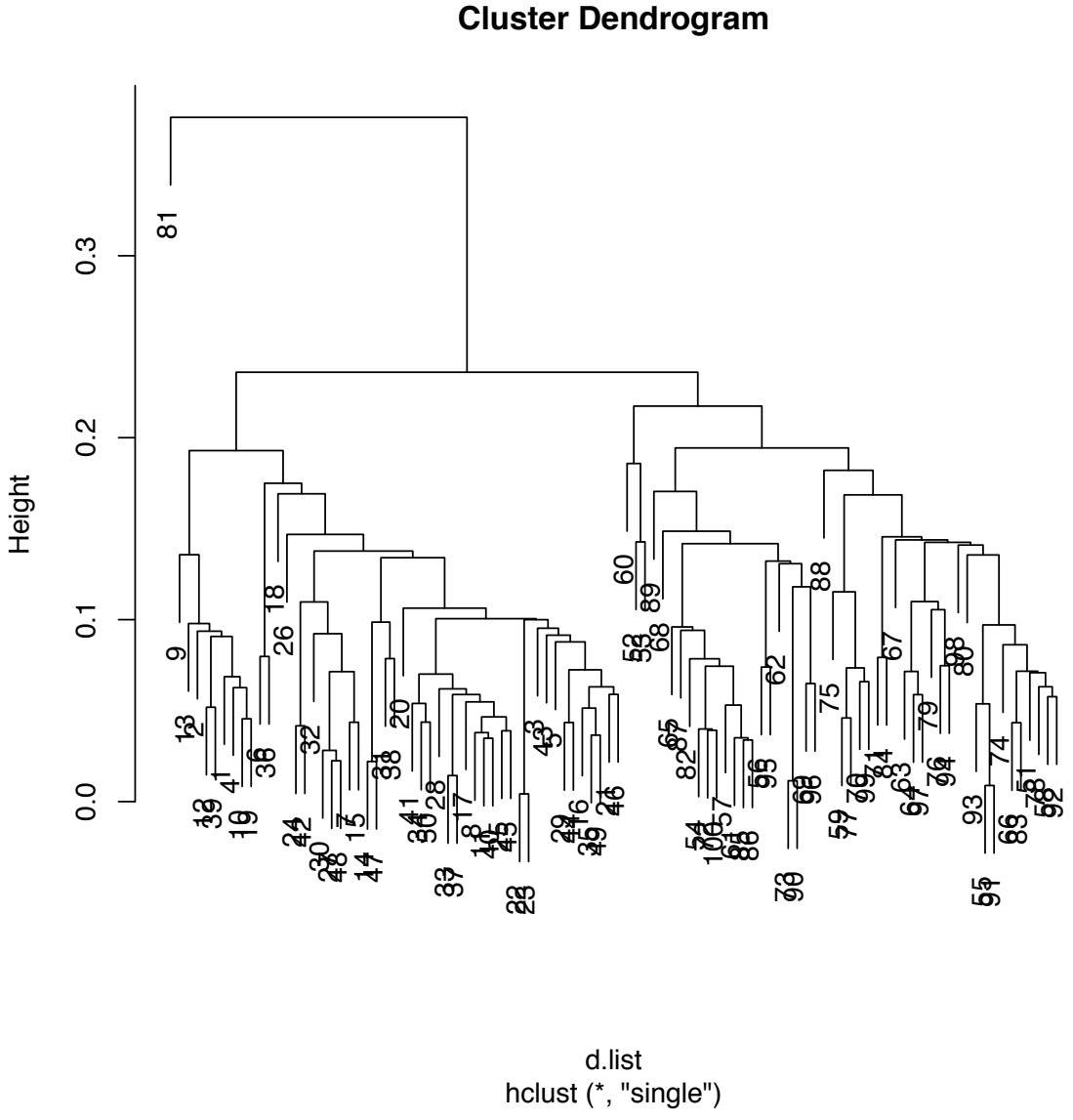
- different similarity metrics can make very different dendrograms (remind you of anything?)
- this method imposes a hierarchical structure on data, even if it does not exist
- minimal complexity is $\mathcal{O}(n^2)$, not good for large datasets

Back to Original Problem

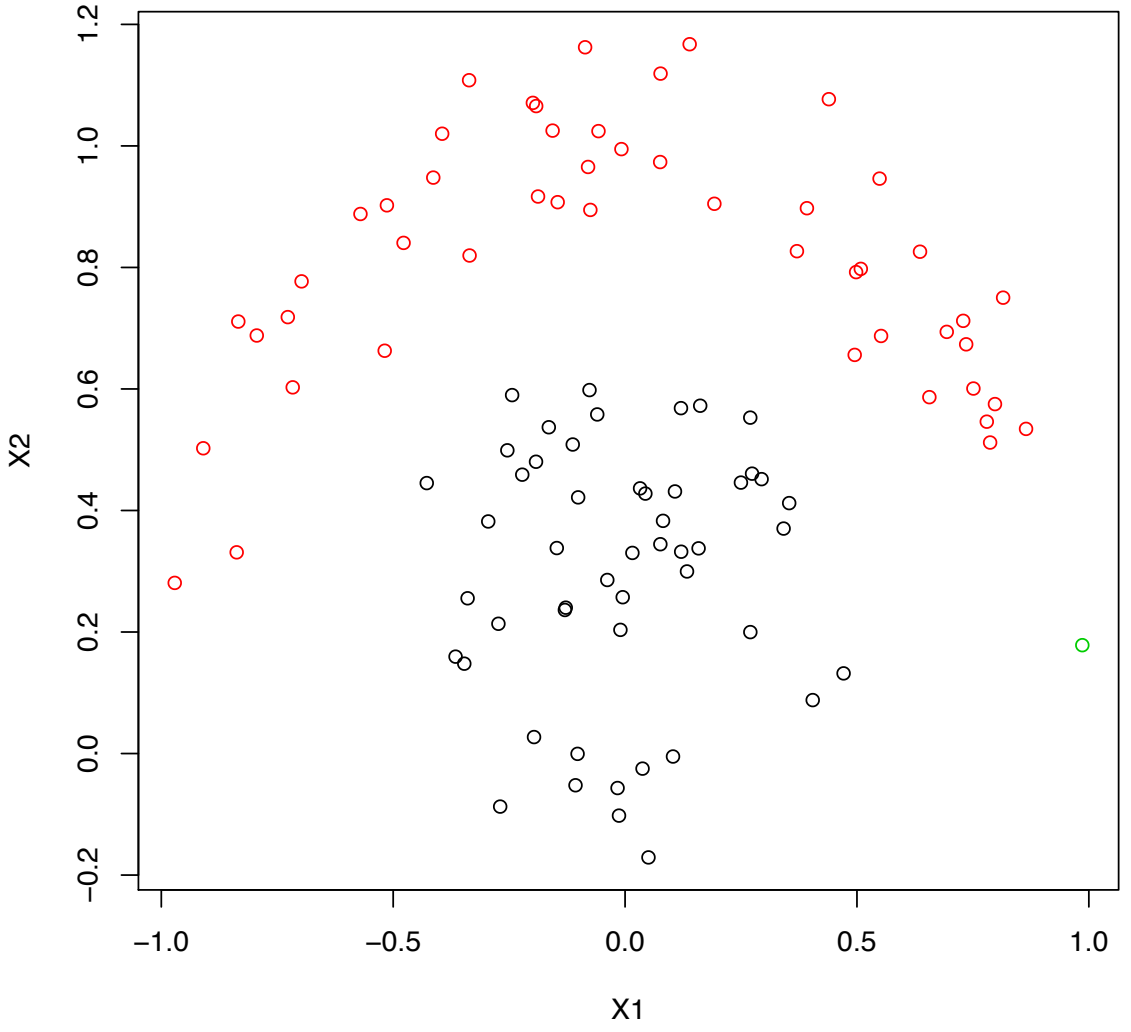
What sort of clustering should we use?



Back to Original Problem



Back to Original Problem



Outline

Divisive Clustering

Now on to *divisive clustering*

Basic algorithm:

- start with all data points in one single group
- repeat: split one cluster
- stop: when all groups only contain individuals

Divisive Clustering

How can we split groups?

Use cuts!

There are many cut-based algorithms, but we will look at one.

Cuts are not the only way to divide data, but they are the simplest.

Divisive Clustering

Inter/Intra Cluster Costs:

Given:

- $U = \{x_1, \ldots, x_n\}$ set of all observations
- A partitioning $C_1, \ldots, C_k$ of the objects

Set:

- $cutcost(C_p) = \sum_{i \in C_p,\ j \notin C_p} d(x_i, x_j)$
- $intracost(C_p) = \sum_{i,j \in C_p} d(x_i, x_j)$
- the contribution of each cluster is the ratio of external similarity to internal similarity

$$cost(C_1, \ldots, C_k) = \sum_{p=1}^{k} \frac{cutcost(C_p)}{intracost(C_p)}$$

Want to find clustering $C_1, \ldots, C_k$ that minimizes $cost(C_1, \ldots, C_k)$

Divisive Clustering

Heuristic way to find $\min cost(C_1, \ldots, C_k)$

1. choose initial partition $C_1, \ldots, C_k$
2. do until change in cost is less than ϵ for outer loop:
   - active set is $\{1, \ldots, n\}$
   - for $i = 1, \ldots, n$:
     - sample j from active set; remove j
     - move x_j into cluster that minimizes cost

Note: more computationally demanding than agglomerative clustering

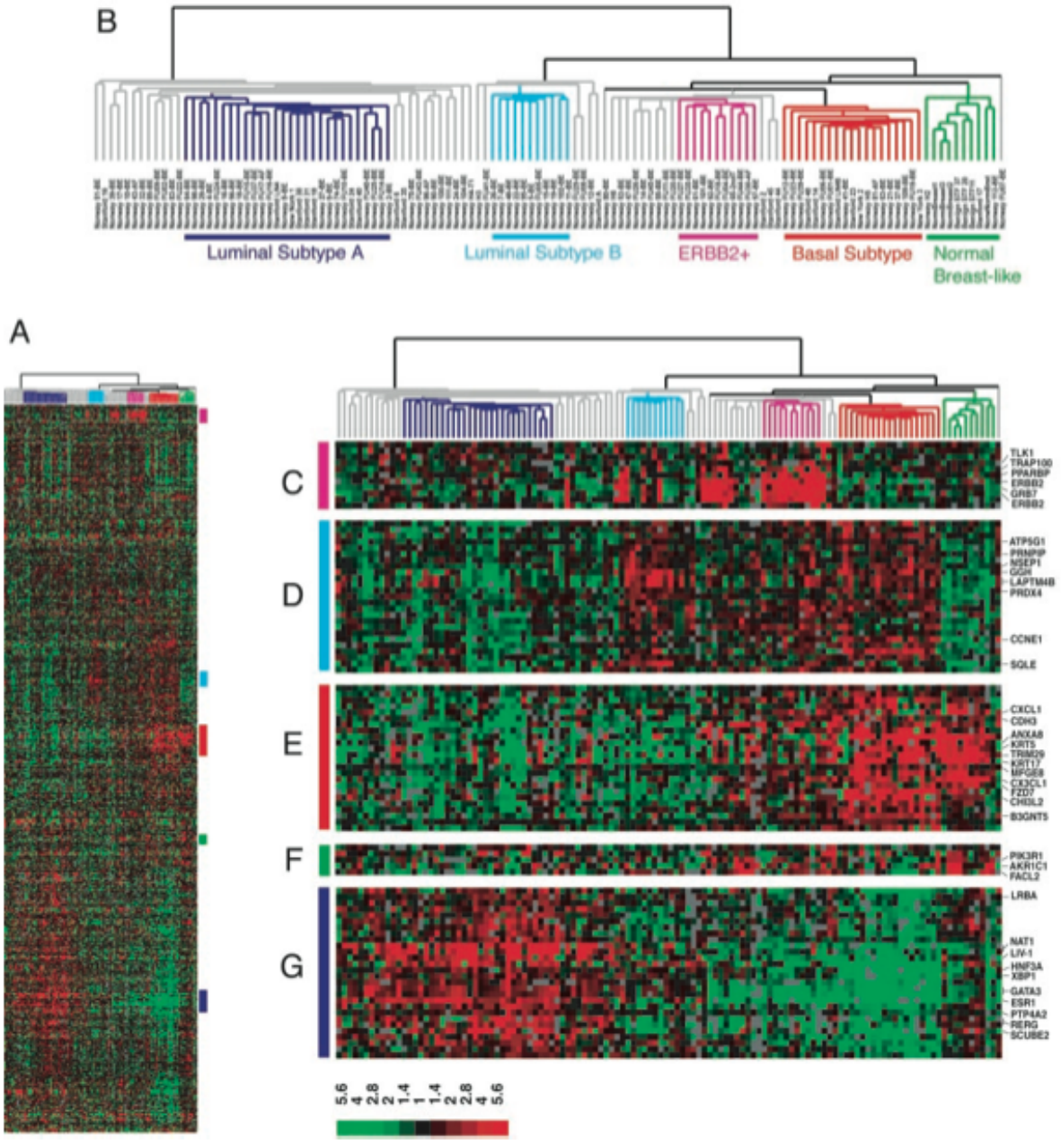
Outline

Applications of Hierarchical Clustering

Microarray gene expression data:

- "Repeated Observation of Breast Tumor Subtypes in Independent Gene Expression Data Sets" (Sorlie et al., 2003)
- hierarchical clustering is used to create new theories (discover tumor subtypes)
- ...which are then tested in the lab
- hypothesis: "breast tumor subtypes represent biologically distinct disease entities"

Applications of Hierarchical Clustering



Outline

Hierarchical Clustering in R

We will use the `hclust` function in the `stats` package (which is likely automatically loaded)

- let's make a function that plots the clusters as connections between points
- (the output of the following code is precisely the set of agglomerative clustering plots we previously viewed)

```
> # Get data
> library(datasets)
> n.faithful <- length(faithful[,1])
> cc.small <- faithful[sample(1:n.faithful,30),]
> n.small <- length(cc.small[,1])
> d.list.small <- dist(cc.small)
> h.list.small <- hclust(d.list.small, method="single")
> plot(h.list.small)
```

Hierarchical Clustering in R

```
> # Make clusters
> cluster.list <- list()
> merge <- h.list.small$merge
> for (i in 1:(n.small-1)){
   temp.vals <- merge[i,]
   if (i==1){
   cluster.list[[i]] <- c(-temp.vals)}
   else{
    # Check to see if negative or positive
    c.temp.1 <- mat.or.vec(1,1)
    if (temp.vals[1] < 0){
      # if neg, include positive version of index
      c.temp.1 <- -temp.vals[1]}
    else{
      # if positive, include indices from cluster
      c.temp.1 <- cluster.list[[temp.vals[1]]]}
    c.temp.2 <- mat.or.vec(1,1)
    if (temp.vals[2] < 0){
      c.temp.2 <- -temp.vals[2]}
    else{
      c.temp.2 <- cluster.list[[temp.vals[2]]]}
    cluster.list[[i]] <- c(c.temp.1,c.temp.2)}}
```

Hierarchical Clustering in R

Plot the results:

```
for (i in 1:(n.small-1)){
  pdf(paste("AgClustS",i,".pdf",sep=""))
  plot(cc.small[,1],cc.small[,2],ylim=c(-1.5,1.5),xlim=c(-1.5,1.5),
      main=paste("Agglomerative Clustering Step (",i,")",sep=""))
  for (j in 1:i){# plot lines
    temp.vec <- cluster.list[[j]]
    temp.n <- length(temp.vec)
    for (k in 1:(temp.n-1)){
      ind.1 <- temp.vec[k]
      for (ell in (k+1):temp.n){
        ind.2 <- temp.vec[ell]
        lines(c(cc.small[ind.1,1],cc.small[ind.2,1]),c(cc.small[ind.1,2],
          cc.small[ind.2,2]),col=3,xlab="X1",ylab="X2")
      }
    }
  }
dev.off()
}
```

Final-esque question

Use hierarchical agglomerative clustering to cluster the following data points with (a) single linkage, then (b) average linkage, and then (c) complete linkage:

x_1	x_2
-1.0	-1.2
-1.2	-1.8
-2.1	-2.4
1.1	1.5
1.5	1.6
1.3	0.7