Economics 217 - Clustering

- Topics covered in this lecture
  - K-Means
  - Hierarchical Clustering
  - DBScan
- Tons of resources exist on these topics, but I've used sample chapters from this book:

  https://www-users.cs.umn.edu/ kumar001/dmbook/index.php

Clustering

- Clustering is a method for arranging objects into like groups
  - Unsupervised Machine Learning
- Clustering can be used to uncover the nature of the data
  - Eg. Correlations in price, neighborhoods of villages
- Can also be used to summarize the data
  - Eg. Mean of similar objects
- Can also be used to speed up other techniques
  - Eg. Applying KNN within clusters

Types of clustering and clusters

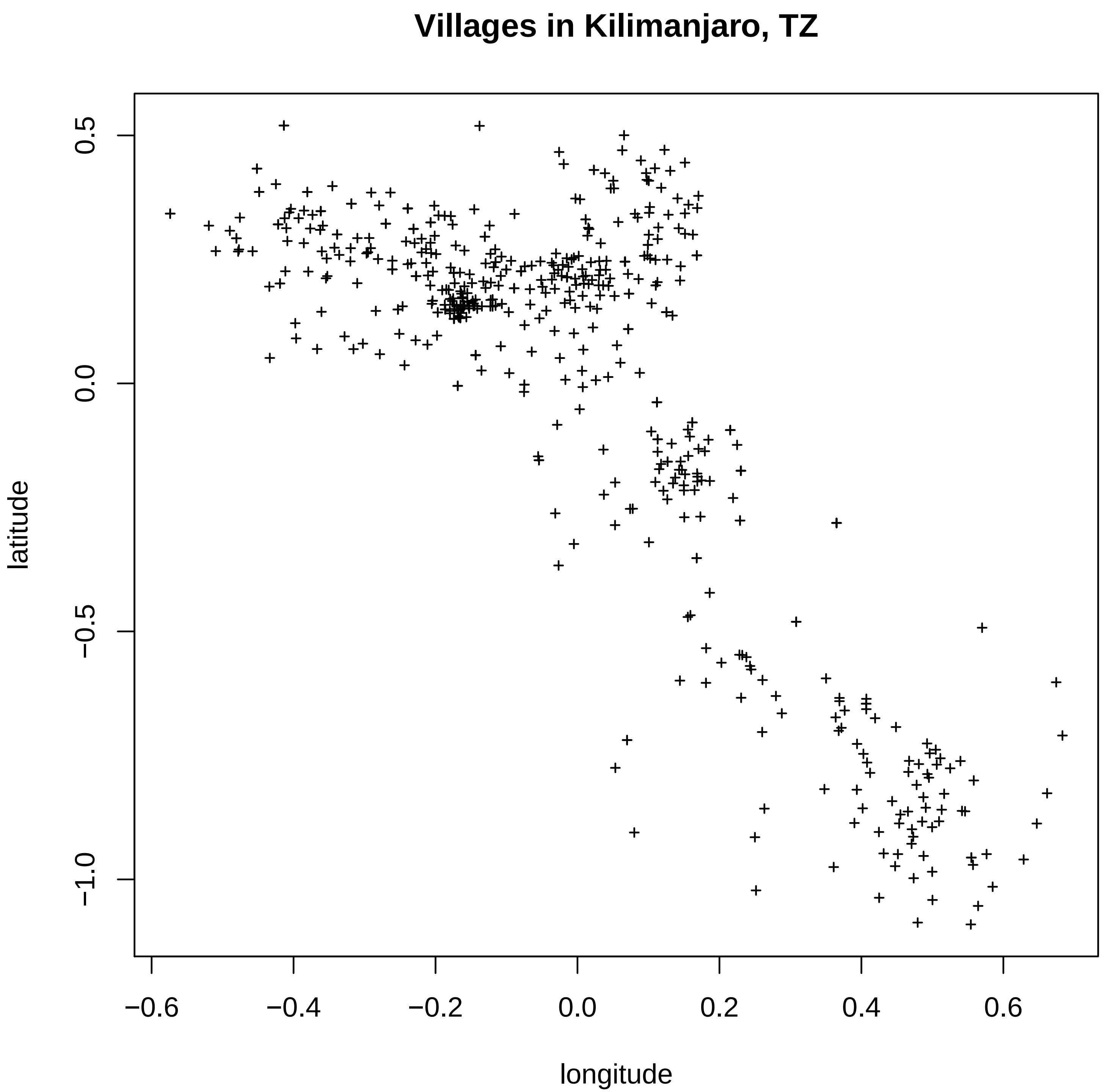
- Clustering
  - Partitioned: objects assigned to non-overlapping groups
  - Hierarchical: clusters defined within larger clusters
- Prototype-based clusters
  - A "prototype" defines the cluster - this is often the centroid of a group of points (or some other central tendency)
  - Objects within a cluster are closer to its prototype than a different prototype.
- Graph-based clusters
  - Objects are defined as connections to one another
  - The type of connection defines the shape of the clusters
- Density-based clusters
  - Regions of high density are surrounded by regions of low density
  - Not all points will be within a cluster

Three techniques

- K-Means
  - Prototype-based, partitional clustering
  - User-specified number of clusters (K), with clusters represented by their centroids
- Agglomerative Hierarchical Clustering
  - Starts with each point as a singleton cluster and then iteratively merging the two closest clusters until a single exhaustive cluster remains.
  - Can be represented with a "Dendrogram", which is an illustration of how clusters are merged. Solution easily determined after choosing the number of clusters.
- DBSCAN
  - Density-based partitioned clustering with noise.
  - Number of clusters is automatically determined by the algorithm parameters and the structure of the data.
  - Points in low-density regions are classified as noise and omitted.

Data: Village locations in Kilimanjaro, TZ

- How could village clustering matter?
  - Administrative governence
  - Market catchment areas
  - Agro-climactic variation
- Data is from a service that provides GPS locations of cities, towns, and villages
  - Anonymized version on the website
  - For now, we will use the clustering algorithms to evaluate clusters in space
- Questions:
  - How to clusters change with each method?
  - What happens when we use an optimal cluster size algorithm?

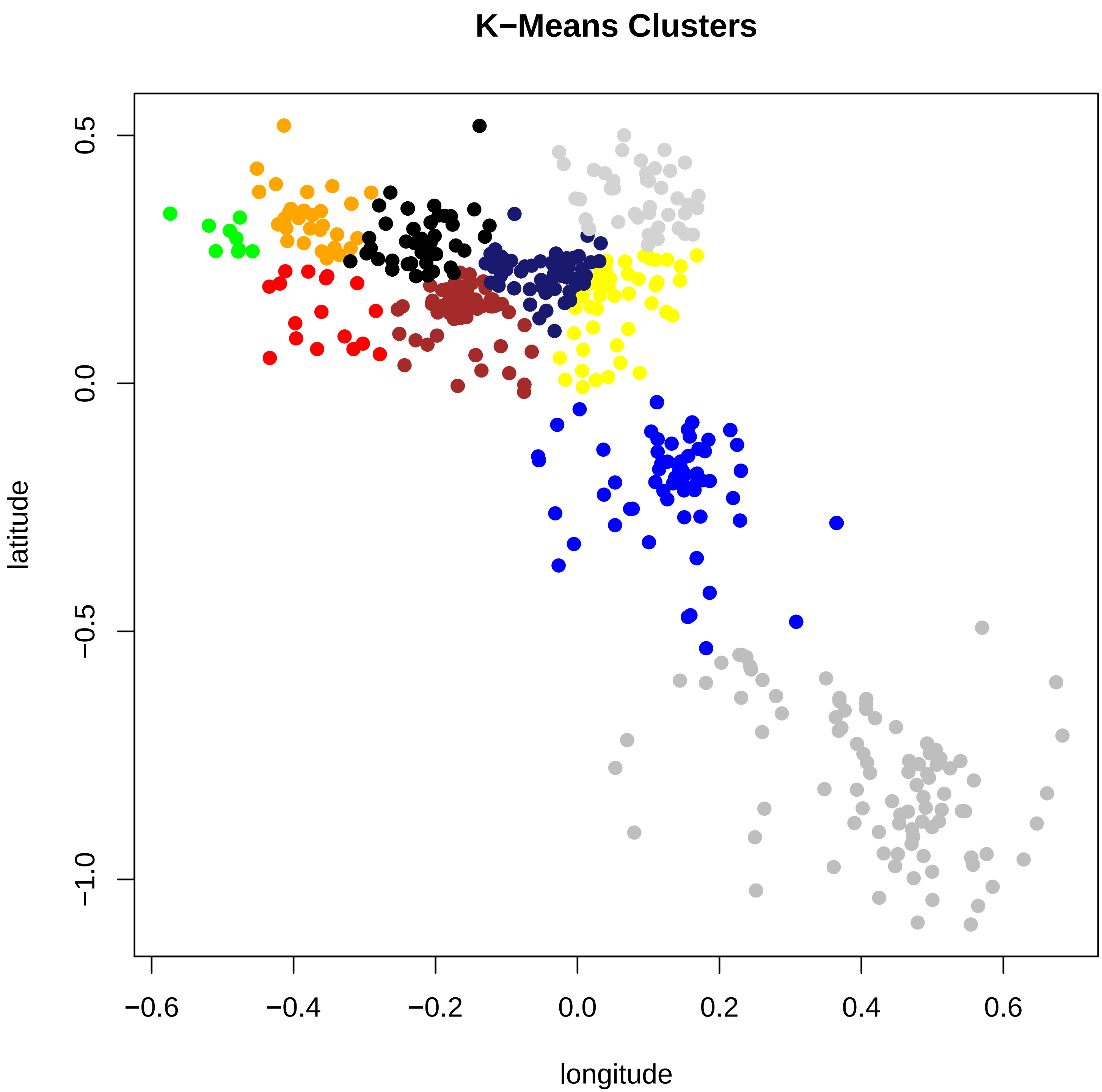

Villages in Kilimanjaro, TZ
latitude
0.5
0.0
−0.5
−1.0
longitude
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

K-Means: The approach

- K-Means
  - Partition clusters based on their proximity to an iterative set of centroids
- Basic algorithm
  - Choose number of clusters, K
  - Start with random cluster centroids
  - Assign points to closest centroid
  - Re-calculate centroids
  - Repeat until convergence
- Upside: Very quick, straightforward
- Downside: May be sensitive to starting values, number of clusters

K-Means: Code

- Load Data
```
sx<-read.csv("https://people.ucsc.edu/~aspearot/Econ_217/TZ.csv")
```
- Create a color pallete to use in all examples
```
color<-c("red","blue","green","gray","yellow","orange",
         "brown","midnightblue","black","lightgray")
```
- Create an empty plot with a title
```
plot(sx,pch=3,cex=0.5,col="white",main="K-Means Clusters")
```
- Calculate the K-means clusters, assuming K=10
```
results10<-kmeans(sx,10)
```
- Illustrate the results, using different colors for clusters, on a plot
```
sx2<-sx
sx2$res<-results10$cluster
for(i in 1:10){
  sx3<-subset(sx2,res==i)[,c("longitude","latitude")]
  points(sx3,pch=19,col=color[i])
  points(results10$centers[i,],pch=19,col=color[i])
}
```



K−Means Clusters
latitude
longitude
0.5
0.0
−0.5
−1.0
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

Hierarchical Clustering: The approach(es)

- Start with all points as singleton clusters, and join observations based on proximity of existing clusters
- Joining methods:
  - Complete: Smallest maximum pairwise distance between clusters. Tends to create agglomerative clusters
  - Single: Smallest minimum pairwise distance between clusters. Tends to create long, connected clusters
- Basic algorithm
  - Choose number of clusters, K
  - Compute a distance matrix between points
  - Merge the two closest clusters
  - Update the distance matrix according to method and repeat
  - Stop when there is only one cluster
- As with K-Means, results will depend on the number of clusters.

Hierarchical Clustering: Code

- Create a distance matrix between villages

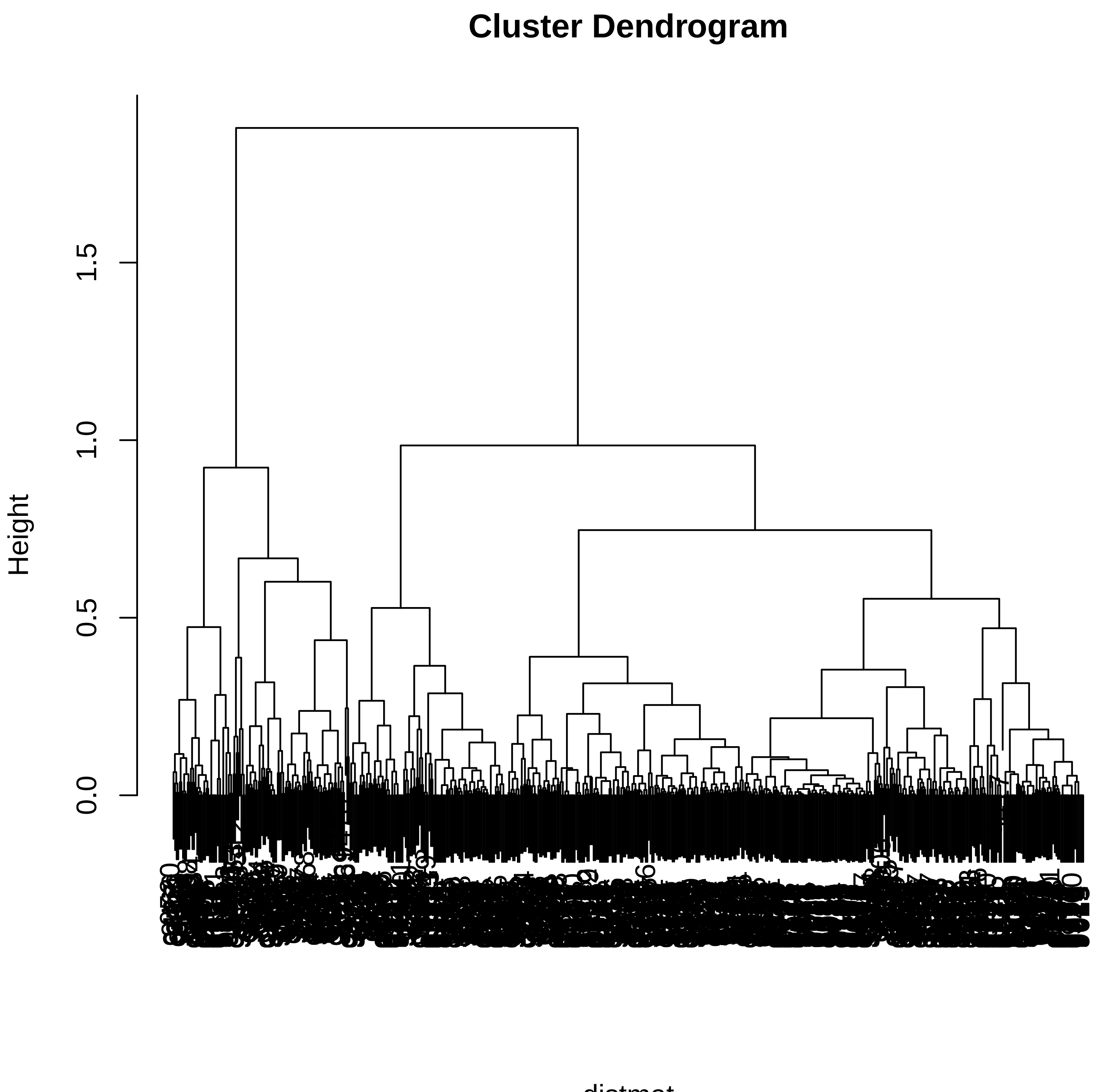
```
distmat<-dist(sx)
```

- Run HC using complete and single linkages

```
results.complete<-hclust(distmat,method="complete")
results.single<-hclust(distmat,method="single")
```

- Results can be represented as a dendrogram
  - Essentially like a tree diagram, starting from singleton clusters, aggregated to one cluster
- Sample of a Dendrogram for the complete linkage

```
plot(results.complete)
```



Cluster Dendrogram
Height
0.0
0.5
1.0
1.5
distmat
hclust (*, "complete")

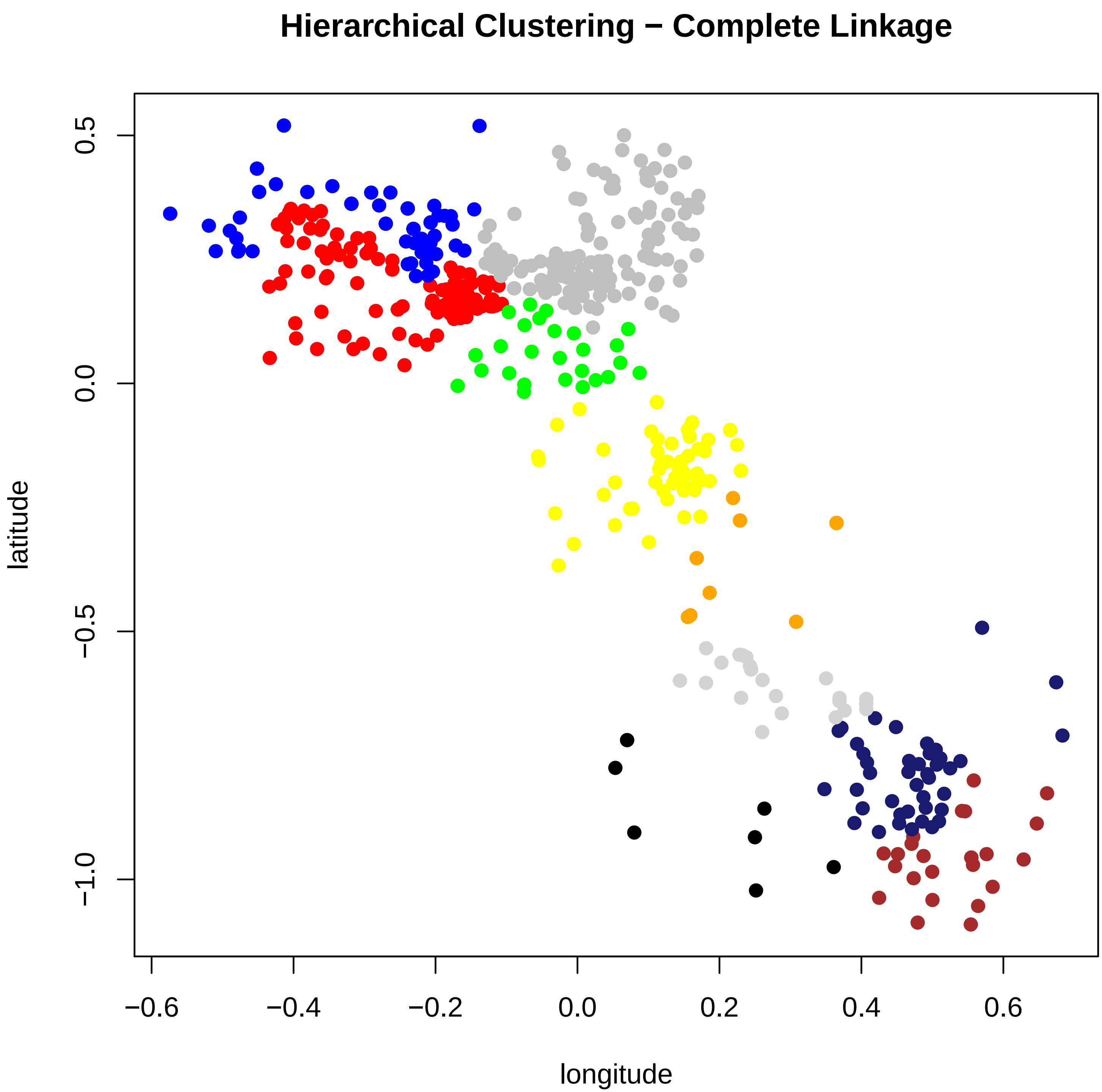
Hierarchical Clustering: Code (cont.)

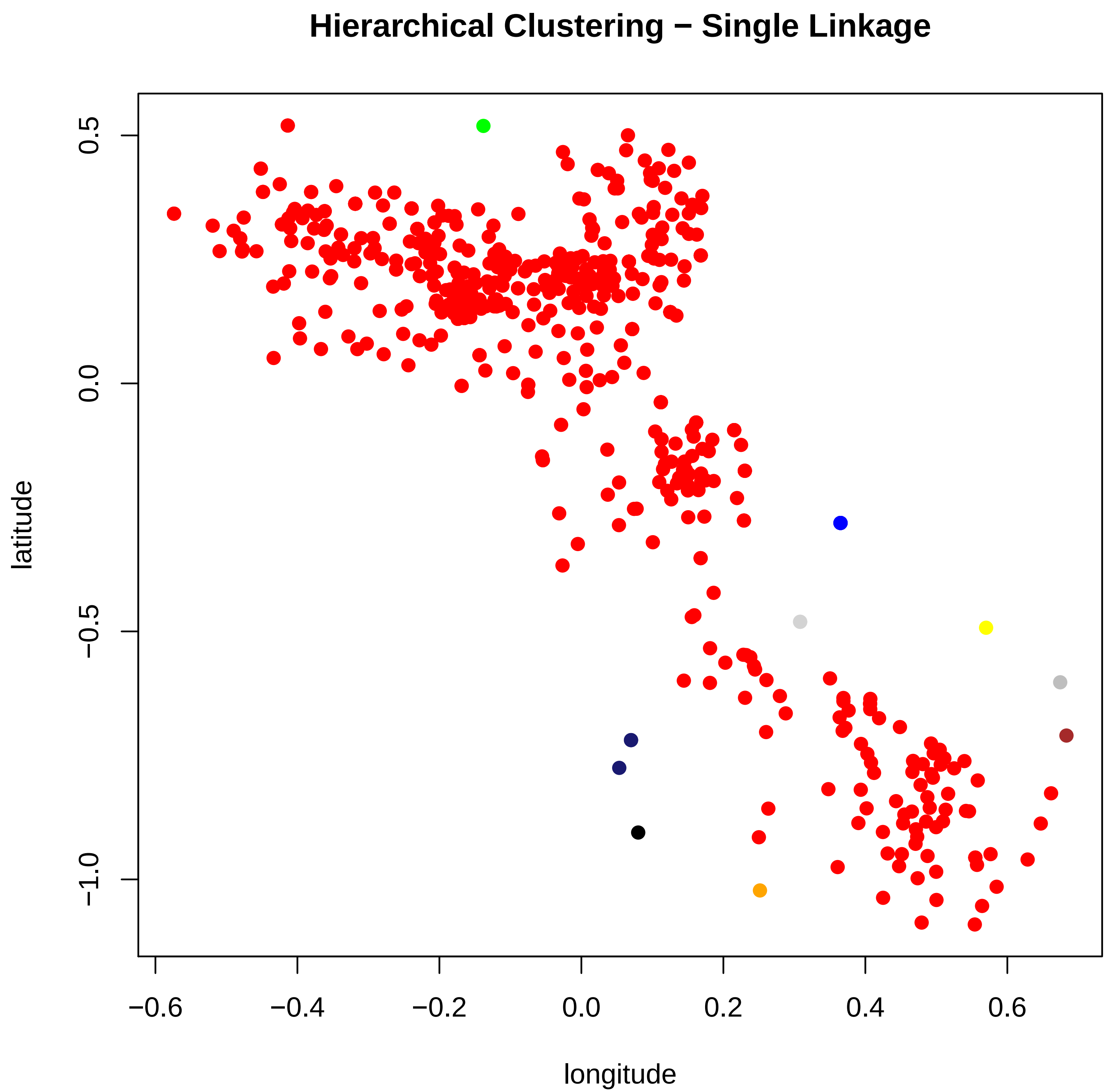
- Illustrate the "complete" results, using different colors for clusters, on a plot

```
plot(sx,pch=3,cex=0.5,col="white",main="Hierarchical
Clustering - Complete Linkage")
sx2<-sx
sx2$res<-as.numeric(cluster.complete)
for(i in 1:10){
   sx3<-subset(sx2,res==i)[,c("longitude","latitude")]
   points(sx3,pch=19,col=color[i])
}
```

- Illustrate the "single" results, using different colors for clusters, on a plot

```
plot(sx,pch=3,cex=0.5,col="white",main="Hierarchical
Clustering - Single Linkage")
sx2<-sx
sx2$res<-as.numeric(cluster.single)
for(i in 1:10){
   sx3<-subset(sx2,res==i)[,c("longitude","latitude")]
   points(sx3,pch=19,col=color[i])
}
```


Hierarchical Clustering − Complete Linkage
latitude
longitude
0.5
0.0
−0.5
−1.0
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


Hierarchical Clustering − Single Linkage
latitude
0.5
0.0
−0.5
−1.0
longitude
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

DBSCAN: The approach

- Group observations into high density areas, with outliers considered noise. Density determined by:
  - *eps* is the radius to determine density.
  - *MinPoints* determines a minimum number of points within a radius
- Three types of points:
  - Core: The interior. A point is a core point if there are at least *MinPoints* within a distance of *eps*.
  - Border: A border point is not a core point, but falls within *eps* of a core point.
  - Noise: Neither a core point nor a border point.
- Basic algorithm
  - Eliminate noise points.
  - Group of connected core points into a separate cluster.
  - Assign each border point to one of the clusters of its associated core points.

DBSCAN: Code

- Load dbscan library

```
install.packages("dbscan")
library(dbscan)
```

- Calculate distance matrix

```
  distmat<-dist(sx)
```

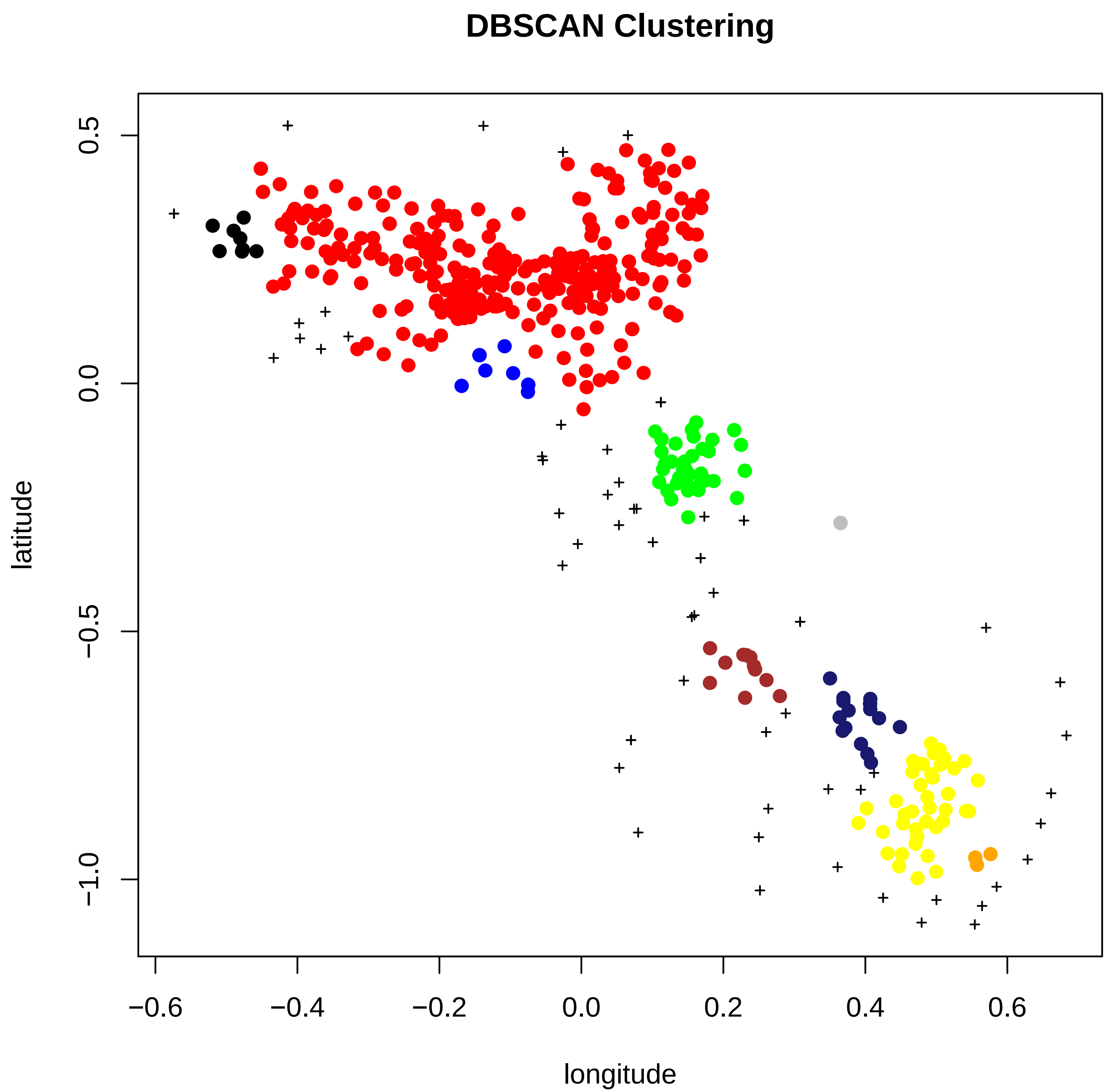
- Create plot with all points

```
  plot(sx,pch=3,cex=0.5,main="DBSCAN Clustering")
```

  - Some of these points will not be assigned to a cluster

- Illustrate the results, using different colors for clusters, on a plot

```
sx2<-sx
sx2$res<-results.dbscan$cluster
for(i in 1:10){
  sx3<-subset(sx2,res==i)[,c("longitude","latitude")]
  points(sx3,pch=19,col=color[i])
}
```


DBSCAN Clustering
latitude
longitude
0.5
0.0
−0.5
−1.0
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6