Multivariate Miscellany

Lecture 11.2 Clustering

Lauren Sullivan

Module: Multivariate Models

Readings

Required for class:

- NA

Optional:

- \textcolor{teal}{Holmes, S. and Huber, W. Modern Statistics for Modern Biology - Chapter 5: Clustering.
- Pathak, M. (2018) Hierarchcial Clustering in R

Clustering

An important part of multivarite statistics is to find groups within our data, so we can group our data that are similar to try to find patterns. We have already talked about a few ways to do this through ordination, where you have treatment variables and you want to see if the data cluster by those treatment variables. However what do you do if you have less prior information about where the clusters should be forming?

We will be talking about **unsupervised** clustering, where groups are allowed to form.

1. k-means, k-medoids
2. Hierarchical clustering
3. Graph clustering

k-means, k-medoid Clustering

An iterative process that attempts to find the shortest distance from all points to the number of group centers (k).

Step 1: Start with your $\mathbf{Y}$ data matrix with only the columns of data you are interested in.

Step 2: Randomly pick k cluster centers out of the number of observations (or cells) in your dataset.

Step 3: Assign the rest of the data observations to a group with the closest center.

Step 4: For each group, replace the selected center with the arithmetic mean of the cluster.

- If using k-medoid, then choose a center of the cluster so distance to all points are minimized.

Repeat steps 3 and 4 until the clusters stabilize.

Data Example

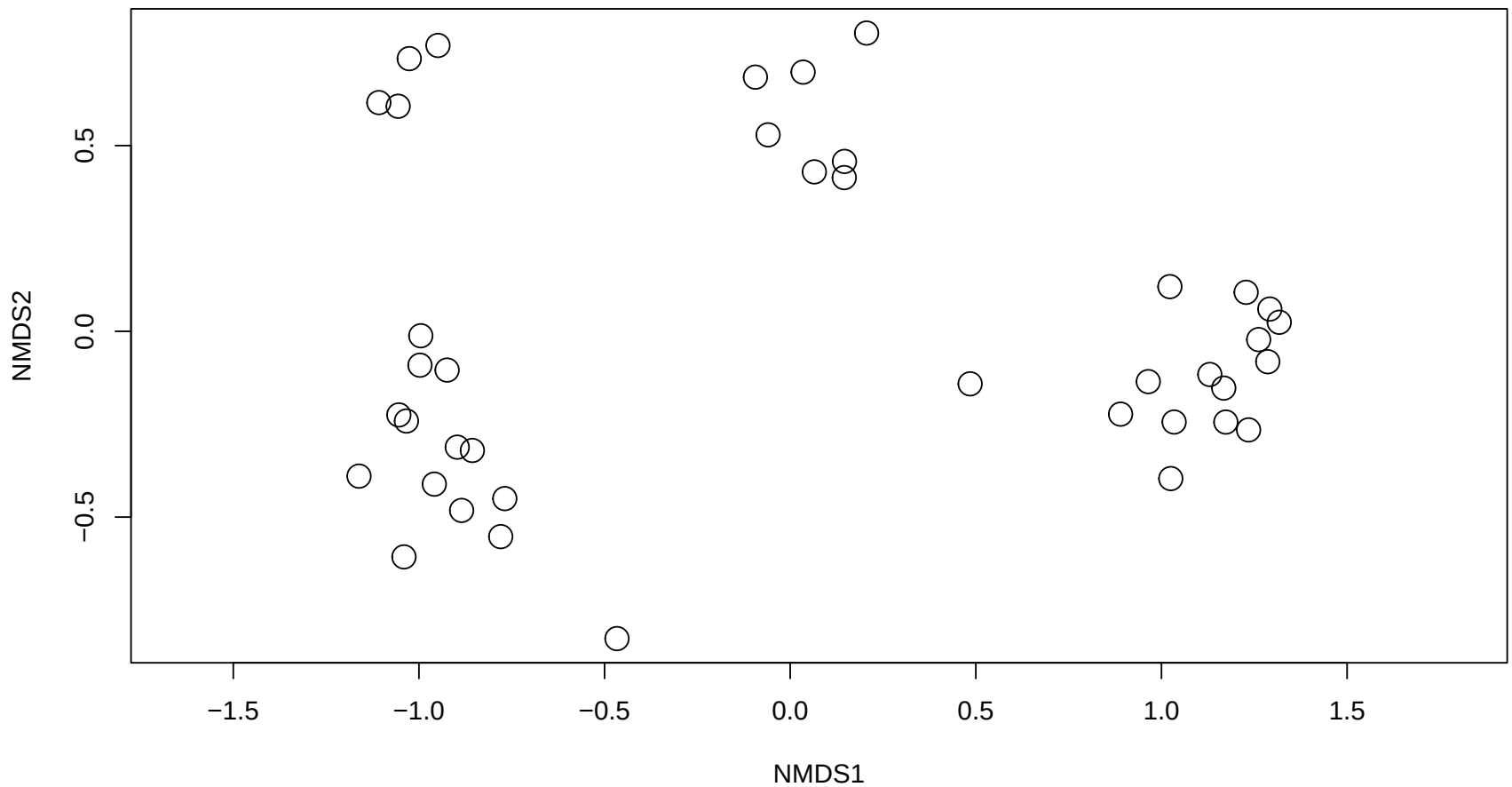
Let's look back at our metabolomics dataset, where rows are accessions and columns are metabolites within the tissues of each accession.

```
## # A tibble: 40 x 6
##    access    unknown_A1 unknown_1 amide_1 amide_12  amide_2
##    <chr>          <dbl>     <dbl>   <dbl>    <dbl>    <dbl>
##  1 Ang267             0     0     0.00754   0.0689 0.0314
##  2 AngAng272          0     0     0.0192    0.0666 0.0568
##  3 AngAng285          0     0     0.0292    0.212  0.0749
##  4 AngAng318          0     0.168 0         0.219  0.0314
##  5 AngStr266          0     0     0.0340    0.267  0.0802
##  6 AngStr320          0     0     0.00425   0      0.133
##  7 Atr255             0     0     0.275     0.312  0.00832
##  8 Atr260             0     0     0.148     0.355  0.164
##  9 Atr262             0     0     0.0638    0.161  0.000876
## 10 Atr299             0     0     0.0707    0.286  0.00179
## # ... with 30 more rows
```

k-means Clustering

This method requires coordinate data, so you need to create your distance matrix, and then pull out the points.

- I will do this with nMDS as an option, but this can be done with any ordination method.



k-means Clustering

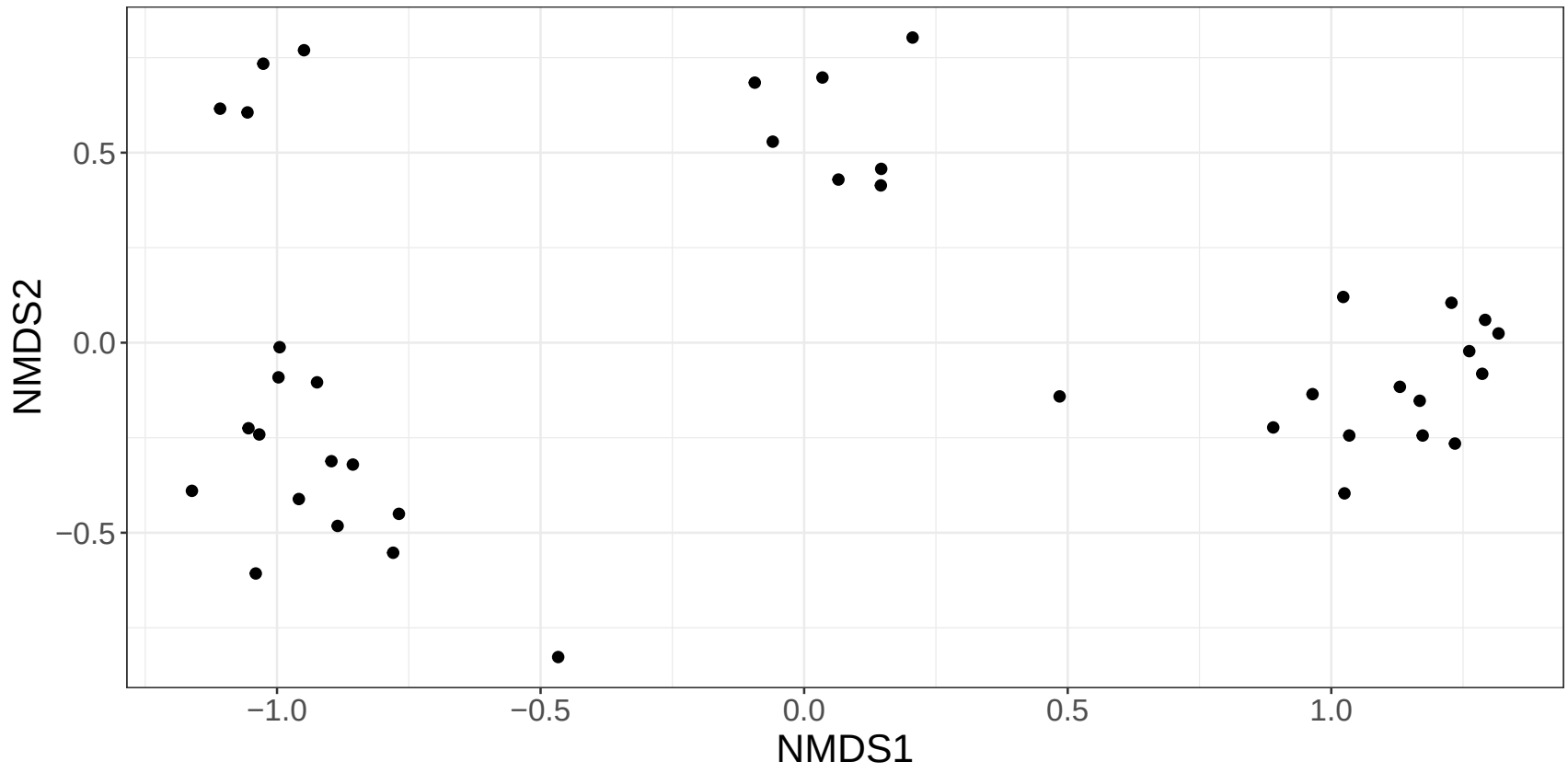
You can also get the nMDS scores using `scores()` and plot by hand.

```
scores <- as_tibble(scores(lipo.mds2, choices, display = "sites"))
scores
```

```
## # A tibble: 40 x 2
##      NMDS1  NMDS2
##      <dbl>  <dbl>
##  1 -0.780 -0.553
##  2 -0.769 -0.450
##  3 -0.959 -0.411
##  4 -1.16  -0.390
##  5 -0.885 -0.482
##  6  1.17  -0.153
##  7 -0.897 -0.312
##  8 -0.856 -0.321
##  9 -1.05  -0.225
## 10 -1.03  -0.241
## # ... with 30 more rows
```

k-means Clustering

```
ggplot(scores, aes(x = NMDS1, y = NMDS2))+
  geom_point(cex = 2)+
  theme_bw()+
  theme(text = element_text(size=18))+
  labs(x = "NMDS1", y = "NMDS2")
```



k-means Clustering with `kmeans()`

Now run your k-means clustering.

```
lipo <- cbind(lipo, scores)
set.seed(100)
lipo.c.kmns <- kmeans(lipo[,45:46], centers = 5, nstart = 10)
str(lipo.c.kmns)
```

```
## List of 9
##  $ cluster     : int [1:40] 4 4 2 2 4 3 2 2 2 2 ...
##  $ centers     : num [1:5, 1:2] 0.0634 -0.9863 1.1008 -0.7881 -1.0347 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:2] "NMDS1" "NMDS2"
##  $ totss       : num 42.6
##  $ withinss    : num [1:5] 0.2166 0.2233 0.9603 0.2657 0.0338
##  $ tot.withinss: num 1.7
##  $ betweenss   : num 40.9
##  $ size        : int [1:5] 7 9 15 5 4
##  $ iter        : int 4
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

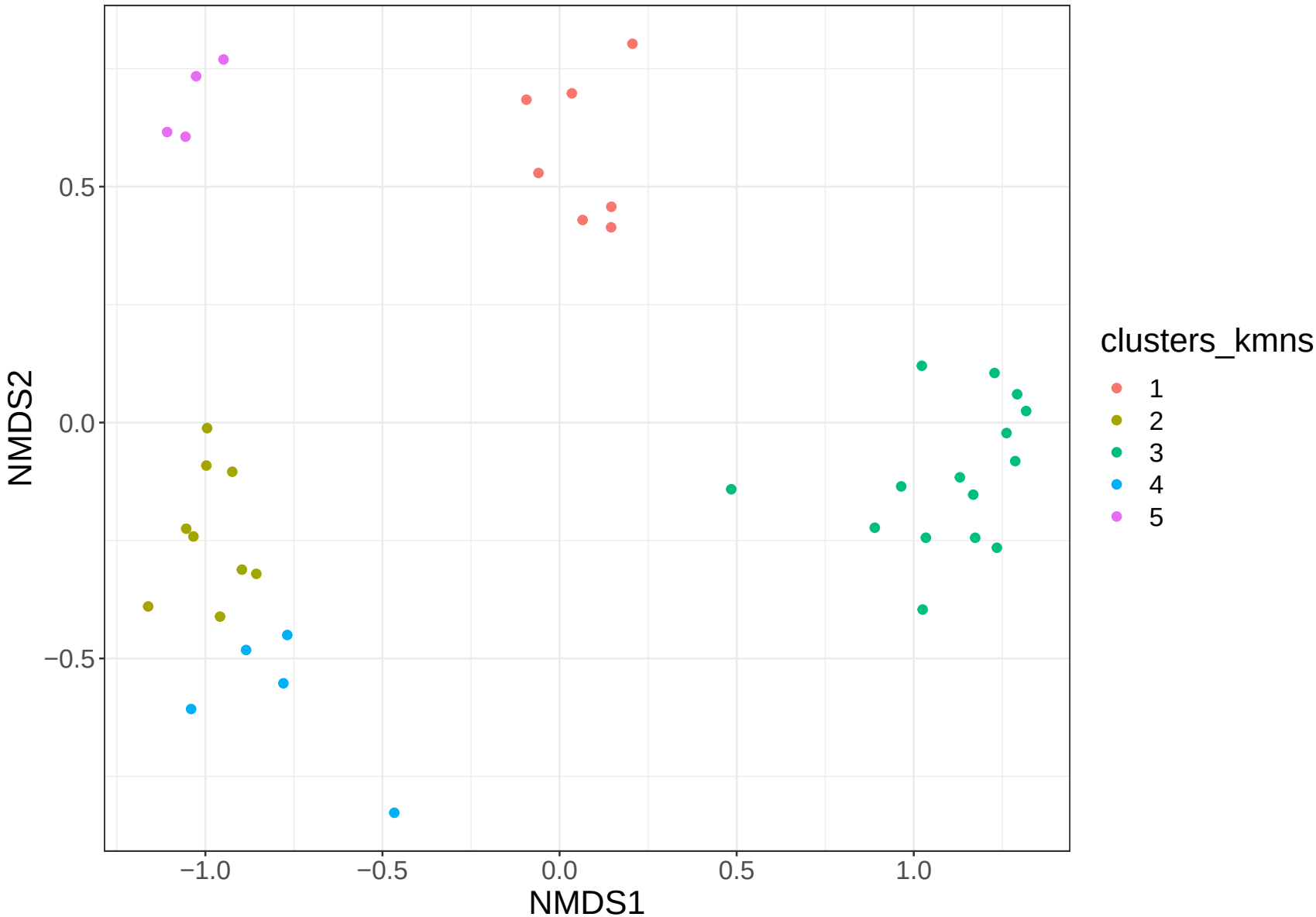
k-means Clustering with `kmeans()`

```
lipo.c.kmns
```

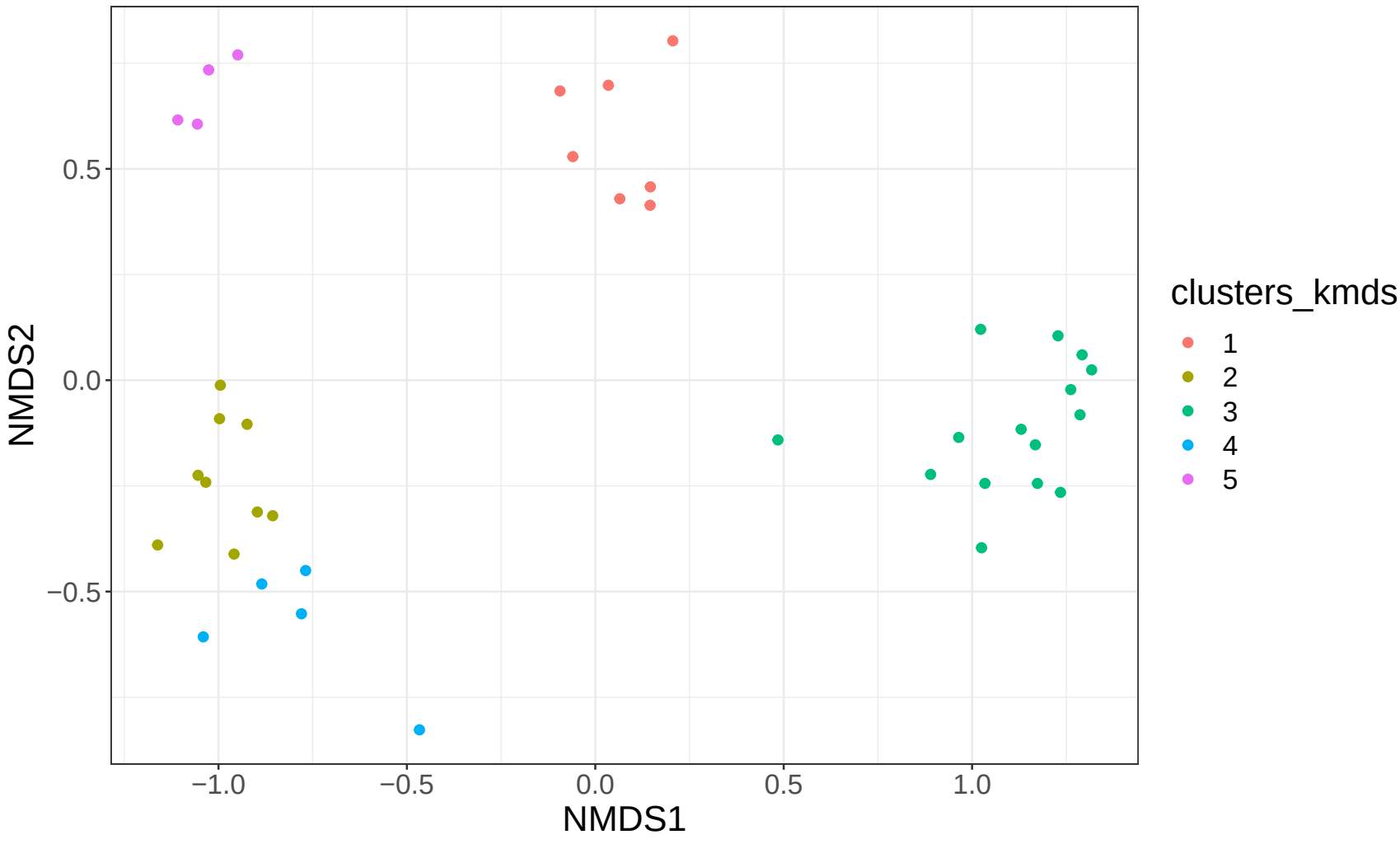
```
## K-means clustering with 5 clusters of sizes 7, 9, 15, 5, 4
##
## Cluster means:
##        NMDS1      NMDS2
## 1  0.0634216  0.5734437
## 2 -0.9863332 -0.2341355
## 3  1.1008092 -0.1141920
## 4 -0.7880718 -0.5838253
## 5 -1.0346827  0.6812801
##
## Clustering vector:
##  [1] 4 4 2 2 4 3 2 2 2 2 3 3 1 1 1 1 4 4 3 3 3 3 3 3 3 3 1 1 1 2 2 2 3 3 3 5 5
## [39] 5 5
##
## Within cluster sum of squares by cluster:
## [1] 0.21655558 0.22329086 0.96031639 0.26567646 0.03383573
##  (between_SS / total_SS =  96.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Plotting k-means Clusters

Add the clusters to your dataset and plot.



The k-medoids Clustering option

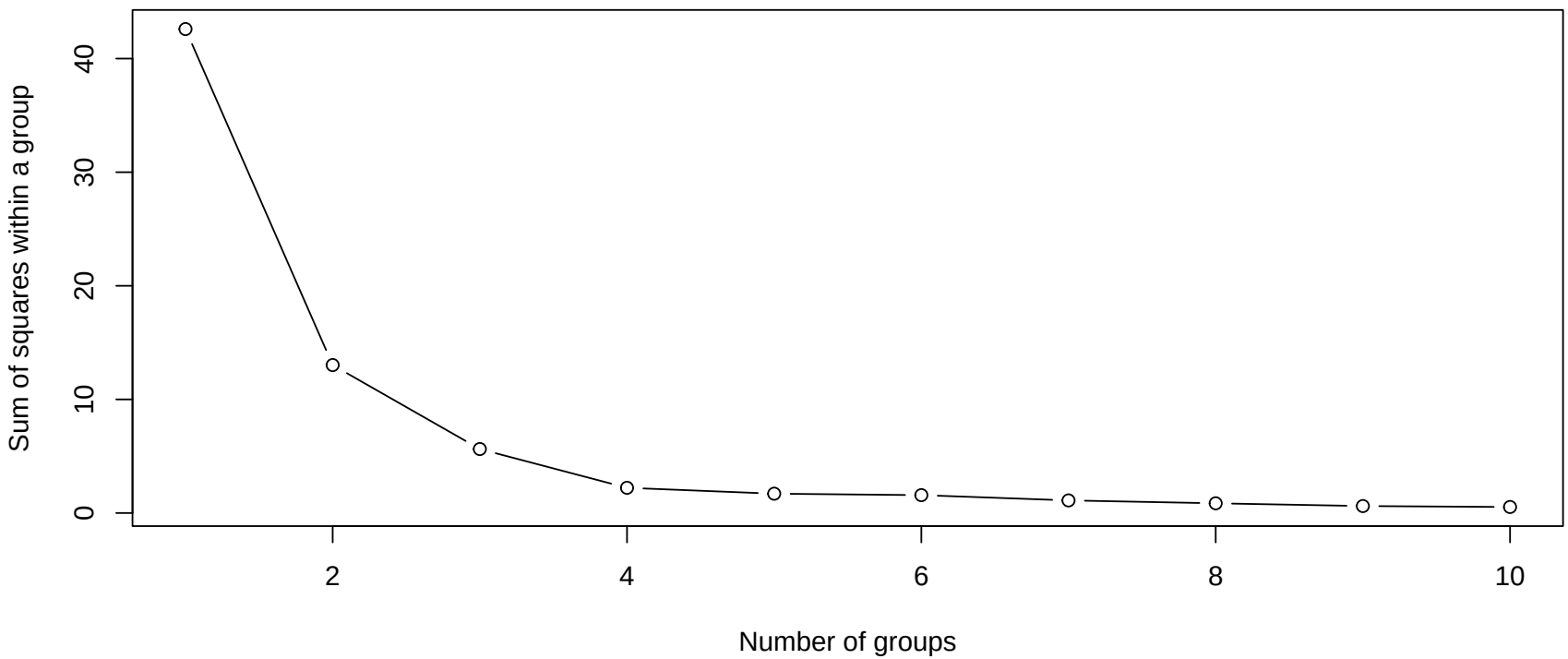


Deciding your k

Here's a function from Luiz Fonseca to help you determine what k to use based on sums of squares within groups.

```
wssplot <- function(data, nc=15, seed=123){
               wss <- (nrow(data)-1)*sum(apply(data,2,var))
               for (i in 2:nc){
                    set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
                plot(1:nc, wss, type="b", xlab="Number of groups",
                     ylab="Sum of squares within a group")}

wssplot(lipo[,45:46], nc = 10)
```



Hierarchical Clustering

Helps you cluster points into hierarchical groups. This algorithm repeatedly tries to combine the two nearest clusters into a larger cluster.

Step 1: Calculate the distance between each pair of points.

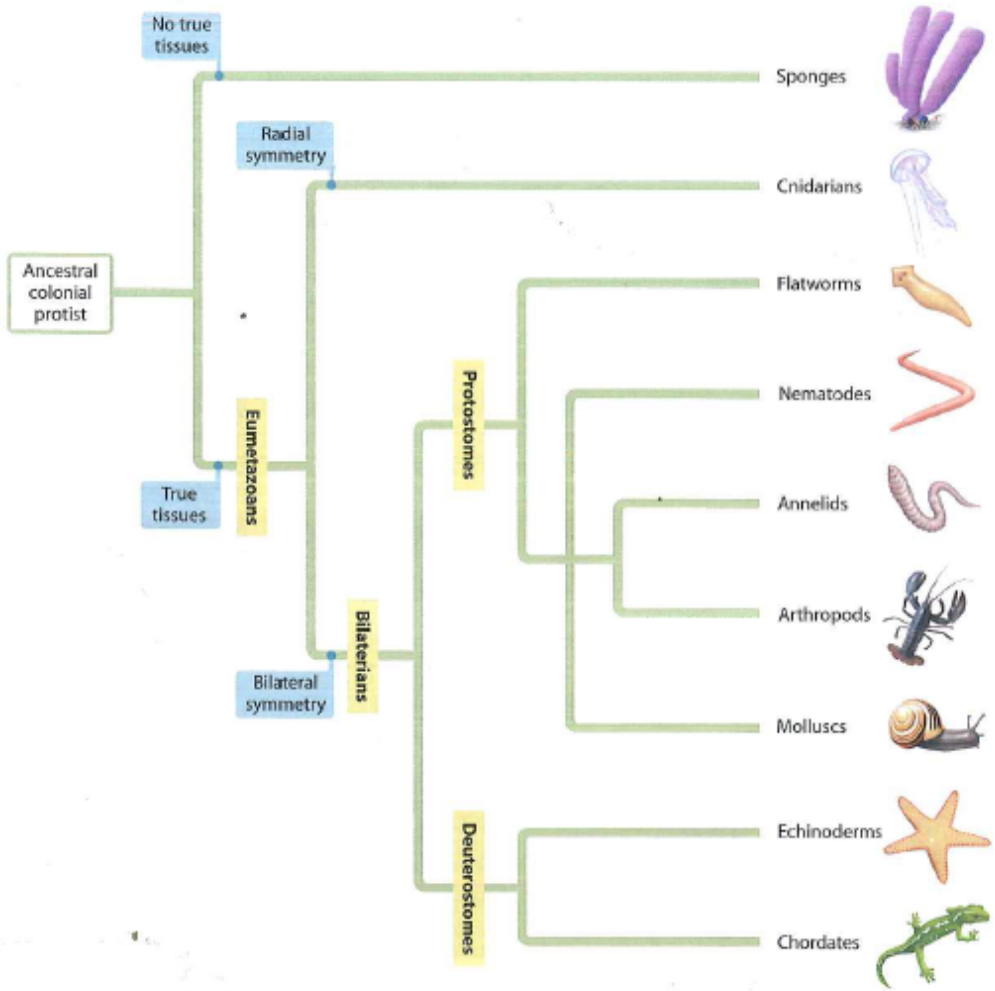
Step 2: Put all points in their own cluster.

Step 3: Merge closest pairs based on distances until your number of clusters goes down by 1.

Step 4: Repeat steps 2 and 3 until everything is merged into a single cluster.

Hierarchical Clustering

It can be helpful to visualize this type of clustering as a dendrogram or tree. Each smaller cluster merges until there is a single cluster at the base of the tree.



▲ Figure 18.4 A hypothesis of animal phylogeny based on morphological comparisons

Hierarchical Clustering with `hclust()`

You want to use a distance matrix for hierarchical clustering, and you want to make sure your data is scaled appropriately.

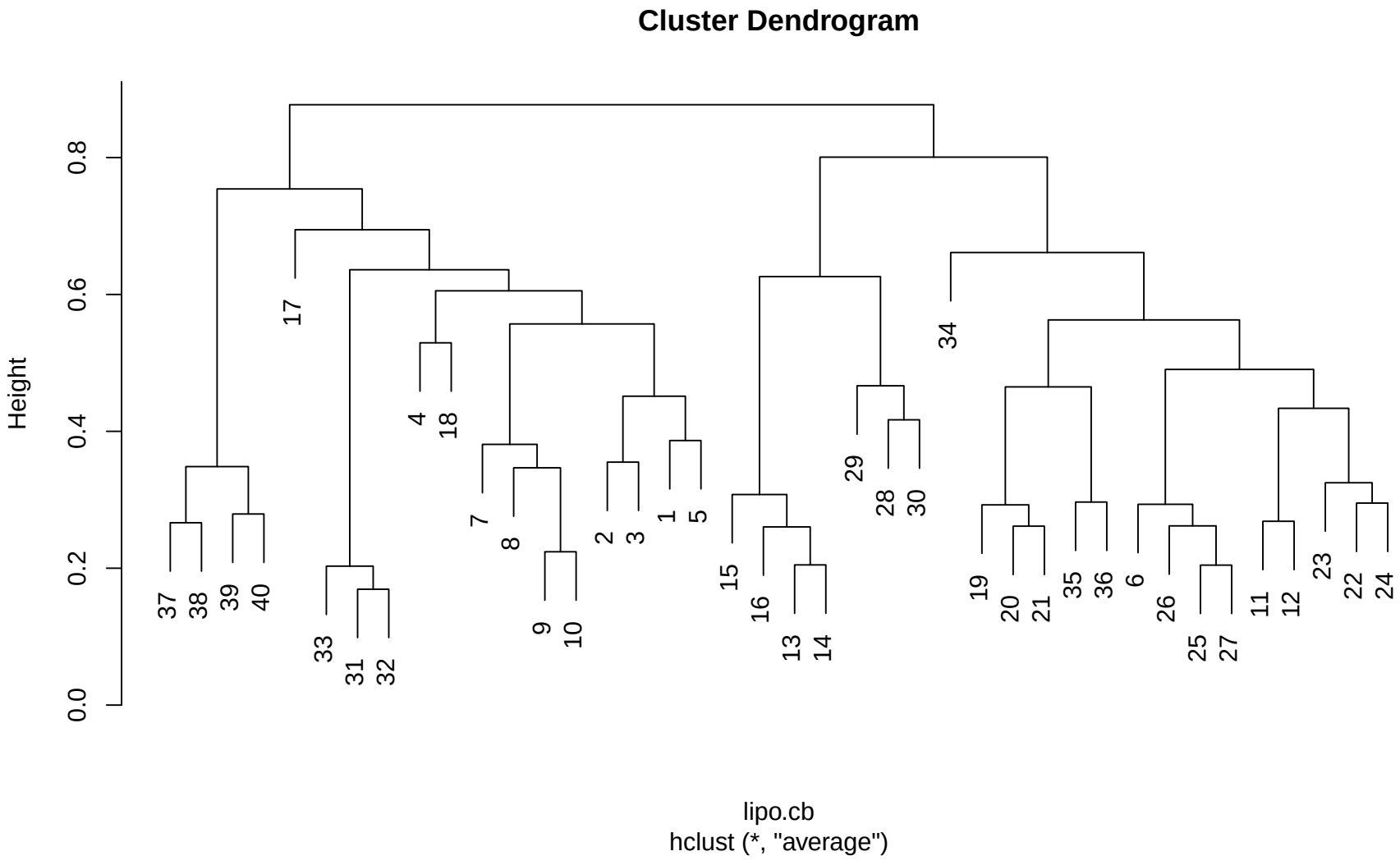
- Here, we have already standardized our data by site totals.
- You need to specify the linkage method with `method=`.

```
lipo.cb <- vegdist(lipo.tot, method = "canberra")
set.seed(143)
lipo.hclust <- hclust(lipo.cb, method = "average")
lipo.hclust
```

```
##
## Call:
## hclust(d = lipo.cb, method = "average")
##
## Cluster method   : average
## Distance         : canberra
## Number of objects: 40
```

Plotting the Hierarchical Clusters

```
plot(lipo.hclust)
```



Cutting the Tree

You can cut the tree to get the desired number of clusters.

```
lipo.cut.5 <- cutree(lipo.hclust, k = 5)
lipo.cut.5
```

```
##  [1] 1 1 1 1 1 2 1 1 1 1 2 2 3 3 3 3 4 1 2 2 2 2 2 2 2 2 3 3 3 1 1 1 2 2 2
## [39] 5 5
```

```
lipo.cut.3 <- cutree(lipo.hclust, k = 3)
lipo.cut.3
```

```
##  [1] 1 1 1 1 1 2 1 1 1 1 2 2 3 3 3 3 1 1 2 2 2 2 2 2 2 2 3 3 3 1 1 1 2 2 2
## [39] 1 1
```

Graph Clustering

Graph clustering uses a graph theoretic framework to select points that are similar based on various criteria

- nearest neighbors
- distance cutoff

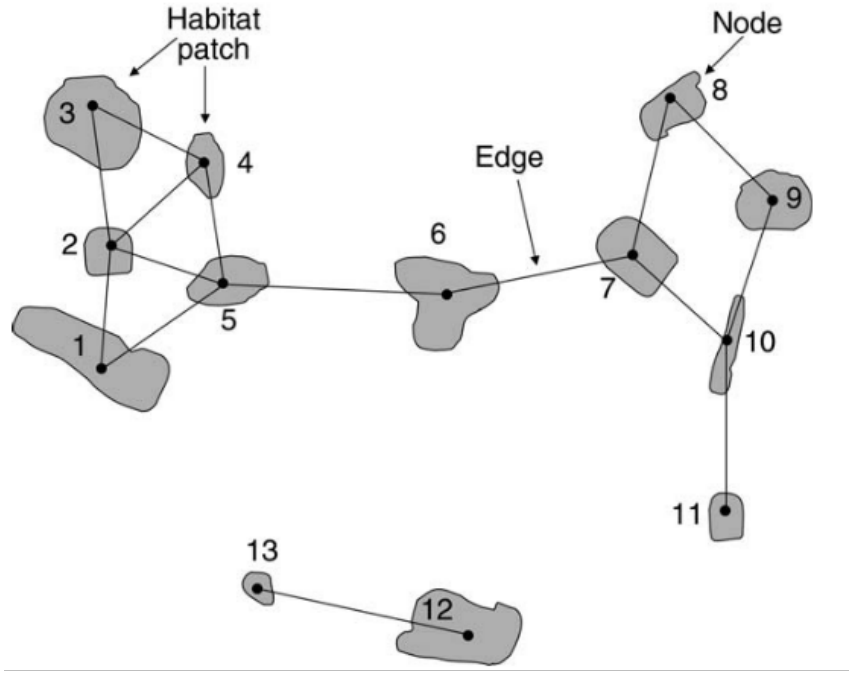
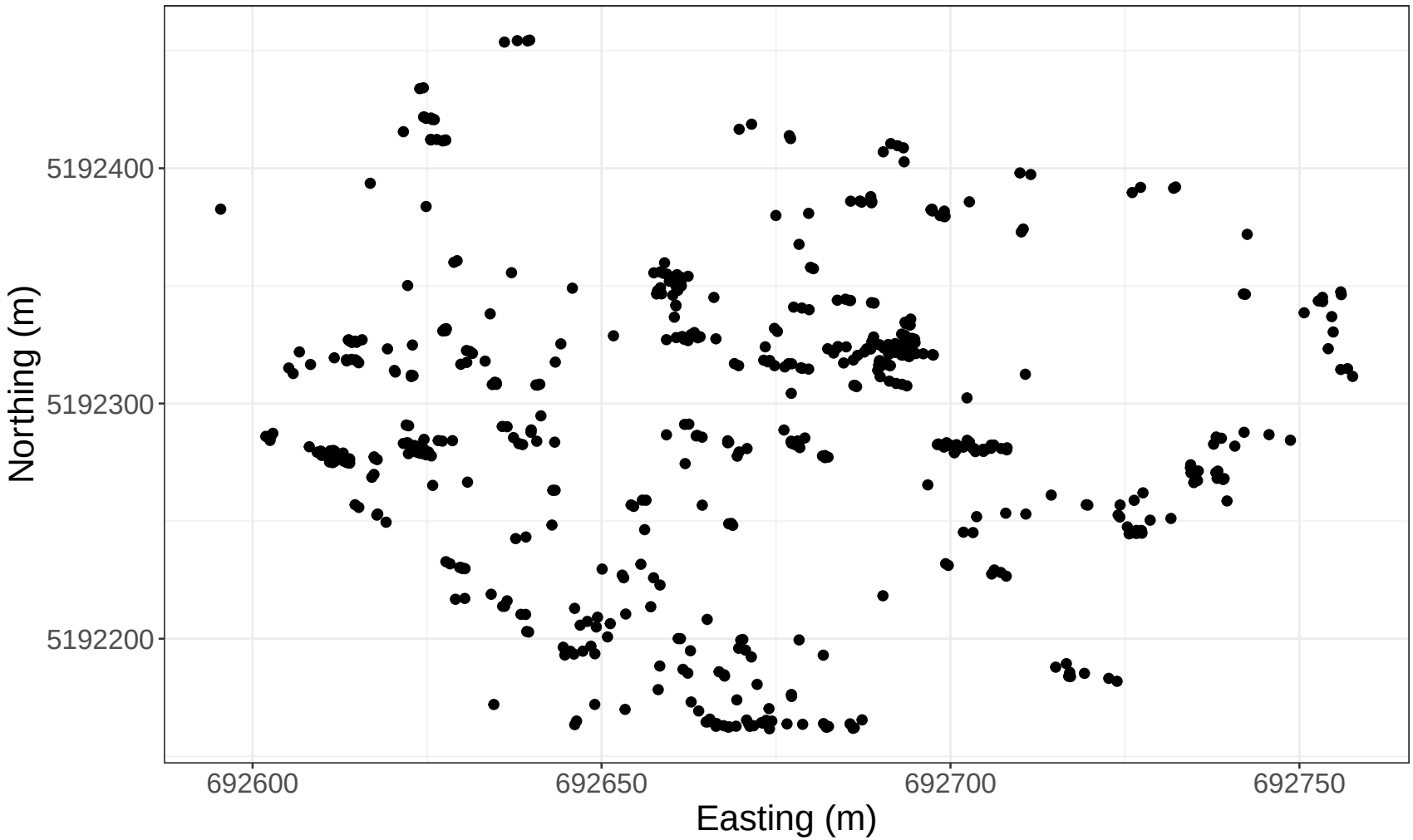


Figure 1: Minor and Urban 2008 Conservation Biology

Graph Clustering

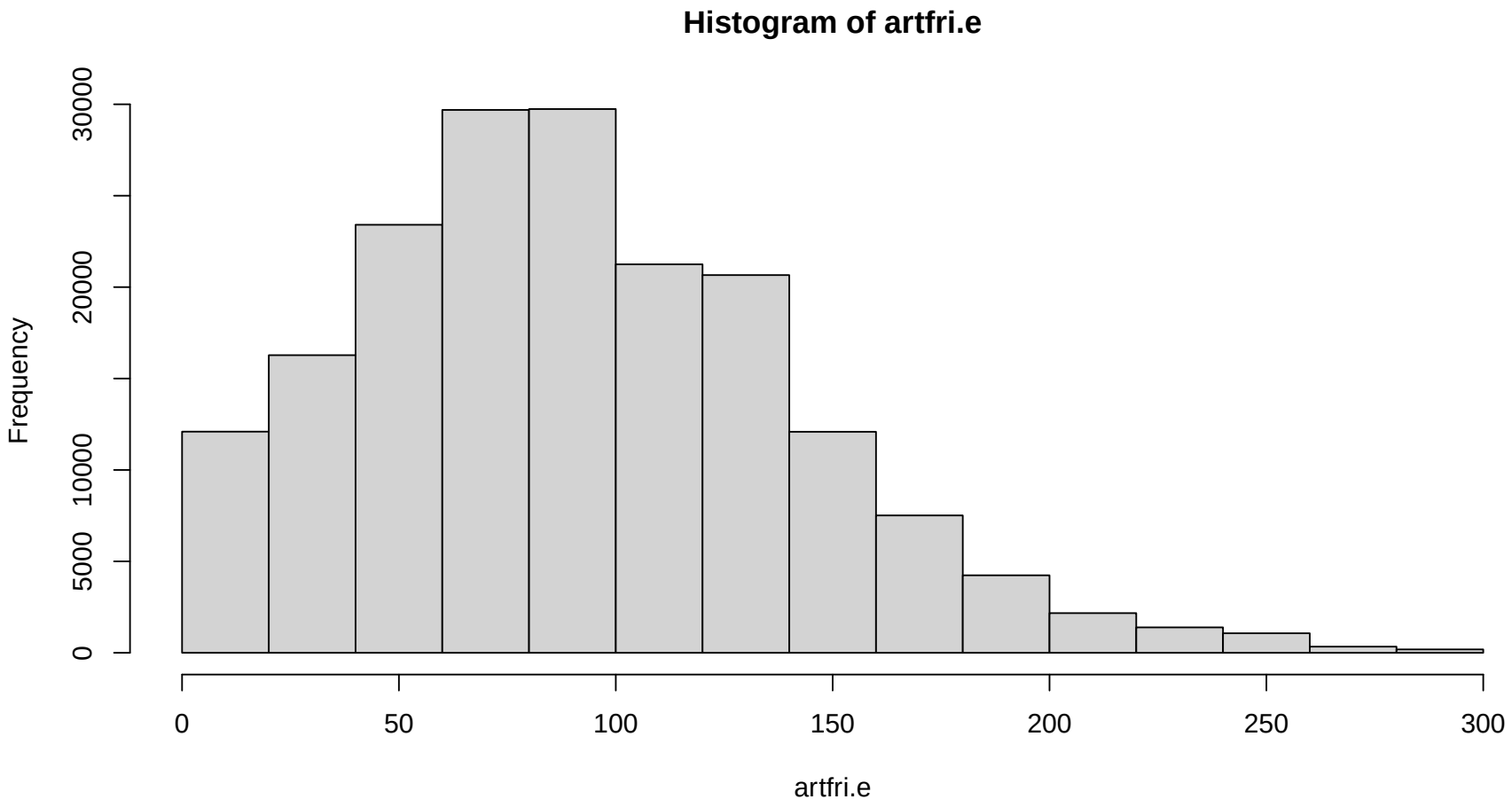
I had plant locations in a prairie, and I was trying to sample seeds from mother plants across the prairie but I wanted to group them in an un-biased way and select from those groups.



Graph Clustering

First I made a distance matrix with Euclidean distance (because it was physical distance between plants).

```
artfri.e <- vegdist(as.matrix(artfri[,7:8]), method = "euclidian")
hist(artfri.e)
```



Graph Clustering

I picked a quantile cutoff from the histogram of neighbor distances of 10%, to create network elements that are all within the closest 10% of distances to each other.

```
quantile(artfri.e, 0.1)
```

```
##       10%
## 29.01604
```

Graph Clustering

`igraph()` can do a lot of things, and here I will use an edge list (a dataframe that describes the distance between each set of points) to create the graph clusters.

```
#first making a dataframe out of the distance matrix
df<-data.frame(as.matrix(artfri.e))

#making sure that rows and columns are labeled the same with a new
#     variable called "id"
names(df)<-artfri$id
df$id<-artfri$id

#melting it so it looks like an edge list (where the connections are)
df_melt<-melt(df, id="id")
```

Graph Clustering with `igraph()`

Next create your graph using the distance cutoff we determined before (the nearest 10% of neighbors).

```
#making a network based on a neighborhood based on a quantile distance
g<-graph.edgelist(as.matrix(subset(df_melt, value <= quantile(artfri.e, .1))[,1:2]),
                  directed=FALSE )

#removing self-loops
g<-simplify(g)
g
```

```
## IGRAPH 18f9129 UN-- 604 18211 --
## + attr: name (v/c)
## + edges from 18f9129 (vertex names):
##  [1] 1--2   1--3   1--4   1--5   1--15  1--54  1--e28 1--76  1--77  1--79
## [11] 1--82  1--75  1--74  1--73  2--3   2--4   2--5   2--15  2--54  2--e28
## [21] 2--77  2--82  2--73  2--6   2--8   2--9   2--11  2--12  2--13  2--14
## [31] 2--16  2--18  2--19  2--20  2--21  2--22  2--23  2--25  2--26  2--27
## [41] 2--28  2--29  2--30  2--31  2--32  2--51  2--56  2--57  2--58  2--61
## [51] 3--4   3--5   3--15  3--54  3--e28 3--6   3--8   3--9   3--11  3--12
## [61] 3--13  3--14  3--16  3--18  3--19  3--20  3--21  3--22  3--23  3--25
## [71] 3--26  3--27  3--28  3--29  3--30  3--31  3--32  3--51  3--58  3--61
## + ... omitted several edges
```

Graph Clustering

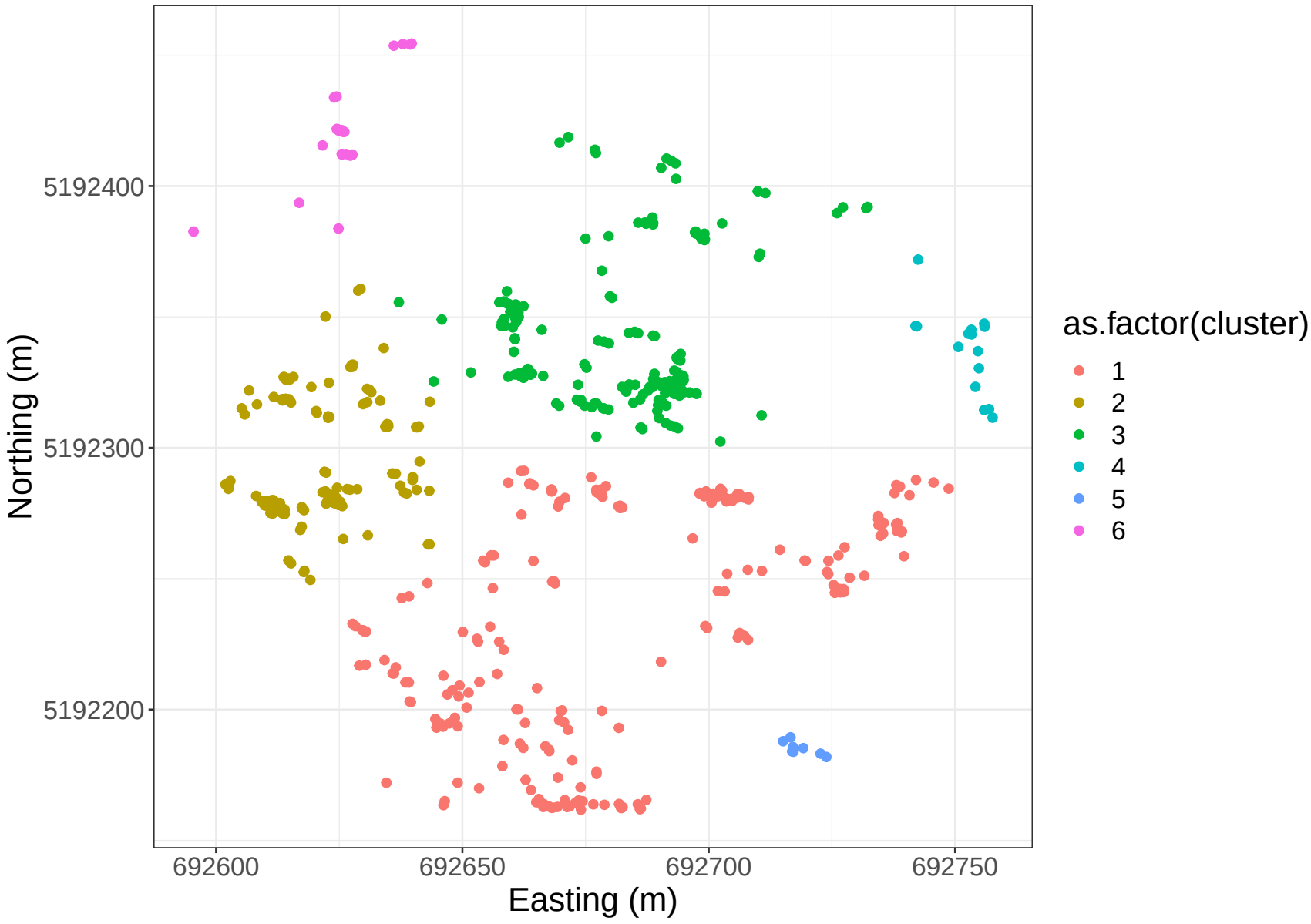
Then create your clusters based on this graph.

```
#create clusters
clusters<-cluster_fast_greedy(g)
network_clusters<-data.frame(as.matrix(membership(clusters)))

#organize your naming
names(network_clusters)<-"cluster"
network_clusters$id<-rownames(network_clusters)
head(network_clusters)
```

```
##    cluster id
## 1        1  1
## 2        1  2
## 3        1  3
## 4        1  4
## 5        1  5
## 15       1 15
```

Graph Clustering



Quiz

Please complete quiz 11.2 on Canvas. Then continue on to the next lecture.