Cluster analysis:

We can only do some of the basics in here. A thorough understanding would require a discussion of matrix algebra and computer algorithms.

But we should be able to do enough so that you understand what's going on.

Introduction.

Suppose we had a collection of objects and we wanted to sort them.

We might want to sort animals into taxonomic groups.

Sort proteins based on structural similarities.

Sort DNA based on similar sequences.

Lots of things we might want to do.

In essence, cluster analysis tries to sort and arrange a group of objects in such a way so that we gain understanding of what's happening.

Let's do a simple example. We measure three species of iris. For each species we measure the length and width of the sepal and petals.

In other words, we have four measurements for each individual:

sepal length, sepal width, petal length, petal width.

This is actually a well known data set that has been used extensively over the years to illustrate "multivariate" statistical techniques.

Multivariate statistics - statistics that deal with things where you have more than one measurement of each object.

Up until now we measured either length or width, but not both.

Multivariate statistics let's you analyze data using all the measurements at once.

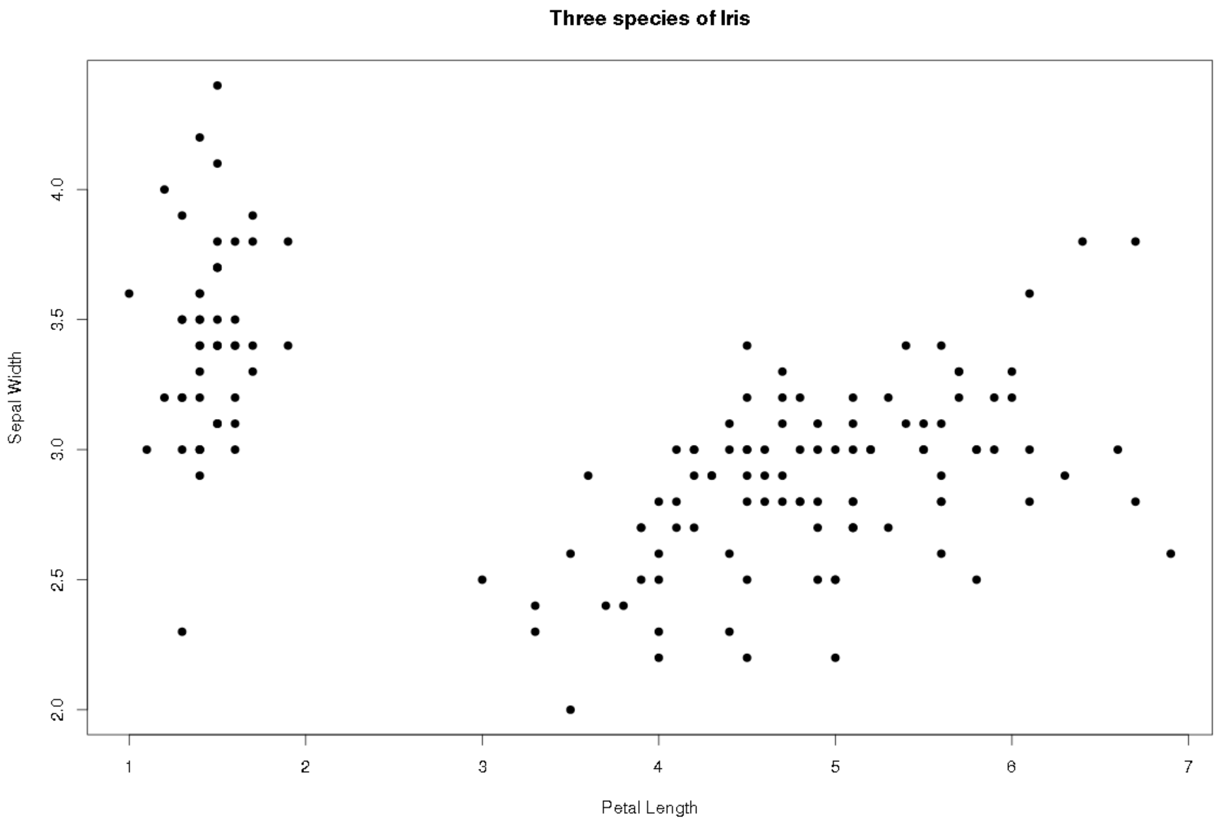
(For more details see the notes on MANOVA, that we won't cover this year).

Back to the iris data.

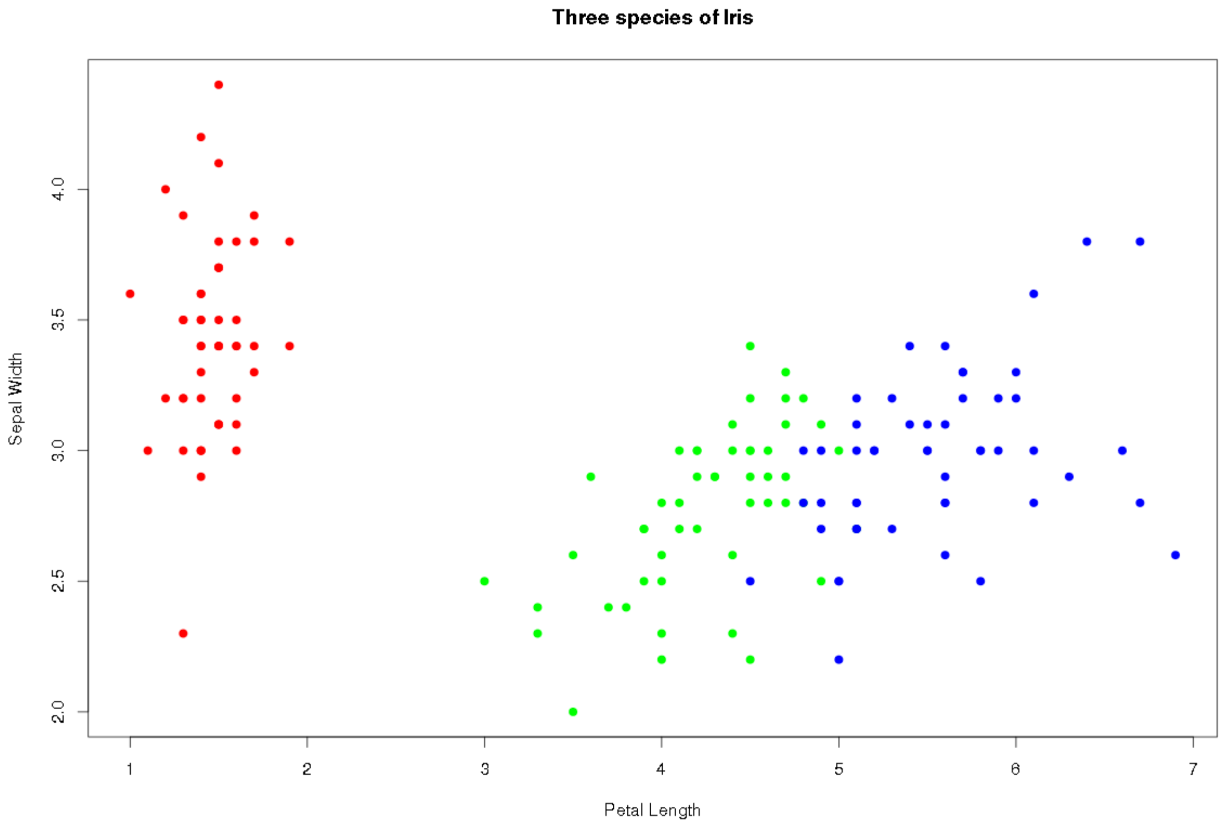
We can't plot all four measurements at once, so let's just stick with two of them:

Sepal width and petal length.

If we plot petal length (x) vs. sepal width (y), we see that there appear to be at least two clusters:



But now if we add color to identify the species, it almost seems like there are three obvious clusters:



(This is an artifact of adding color. Compare again with the previous plot to see that).

But it's also possible that we know nothing about flowers. We might not even know how many species we're dealing with (maybe we're dealing with 150 species!?).

This leaves us with two approaches to clustering or sorting things (his is grossly simplified, but it'll do for now):

1) K-means clustering.

We use this if we have some idea of how many clusters to expect.

2) Hierarchical clustering.

We use this if we are just trying to sort everything based on similarities. This type of clustering is often followed up by generating a “tree” as well.

Let's do an example of each.

K-means clustering.

Suppose we knew how many clusters we wanted. With the flower example we might know we're dealing with three species.

How can we sort our clusters into species (suppose we didn't know which set of measurements go with which species)?

We start by picking three spots at random.

For each spot, we then calculate which of the remaining spots are closest to that spot.

(We'll illustrate this on the board)

We then take the “average” of these sorted spots as a new starting point.

Re-calculate which of the remaining spots are now closest to this “new” starting point.

(Again, we'll illustrate this on the board)

We repeat this until the points don't change groups anymore.

Note that the above is something you can easily program into a computer (it's actually an algorithm, or outline, telling the computer what to do).

Let's try this in R:

```
# Note that the “iris” data set is always present in R.  Just type
# “iris” at the command prompt to see it.

iris

# But we're only interested in Sepal Width and Petal Length, so
# let's get rid of the rest of the data:

iris2 <- iris[c(2,3)]
# this tells R to keep only the 2nd and 3rd columns from the iris
# dataset and put the rest into “iris2”

# now we just use the “kmeans” command and tell R how may clusters
# we want:

irisfit <- kmeans(iris2, 3)
# Note that we need to put this into a new variable (“irisfit” in
# this case).

# irisfit contains the classifications we're interested in.  For
# each observation kmeans has assigned our observation to one of
# three clusters.  If you want to see it, just type “irisfit”:

irisfit

# but this isn't that informative, so let's do a little more work:

iris3 <- data.frame(iris2, irisfit$cluster)
# this tells R to combine iris2 (which has our measurements) with
# the classification results of our kmeans procedure.

# Now we have one data set giving the sepal width, petal length,
# and the cluster that R thinks each observation belongs to.

iris3

# we can plot all this to see what it looks like:

plot(iris3$Petal.Length,iris3$Sepal.Width,pch = 19,col =
c("green","red","blue") [unclass(iris3$irisfit.cluster)],ylab =
 "Sepal Width",xlab = "Petal Length",main = "Three species of Iris
classified by K-means clustering")

# We don't have the time to explain this command in detail, but
# here's a brief summary:

# basically it this plots petal length vs. sepal width:
# “pch” is the size of the dots
# “col” tells R which colors to use for our dots (this SHOULD
# match the number of different classes you have or it may give
# you weird results).
# “unclass” tells R what to use to assign colors (i.e., it will
# assign colors based on the classification variable.
# “ylab”, “xlab” and “main” should be self explanatory.
```

Hierarchical clustering.

We don't have any idea how many clusters might be appropriate so we just start to sort based on how far away two points are from each other.

For example, two points that are close to each other should be two observations that are a lot alike (at least for the things you measured).

So we proceed as follows:

We calculate all possible distances between points.

A “distance” matrix, if you're curious.

The two closest points become one cluster.

The second two most closest points become another cluster.

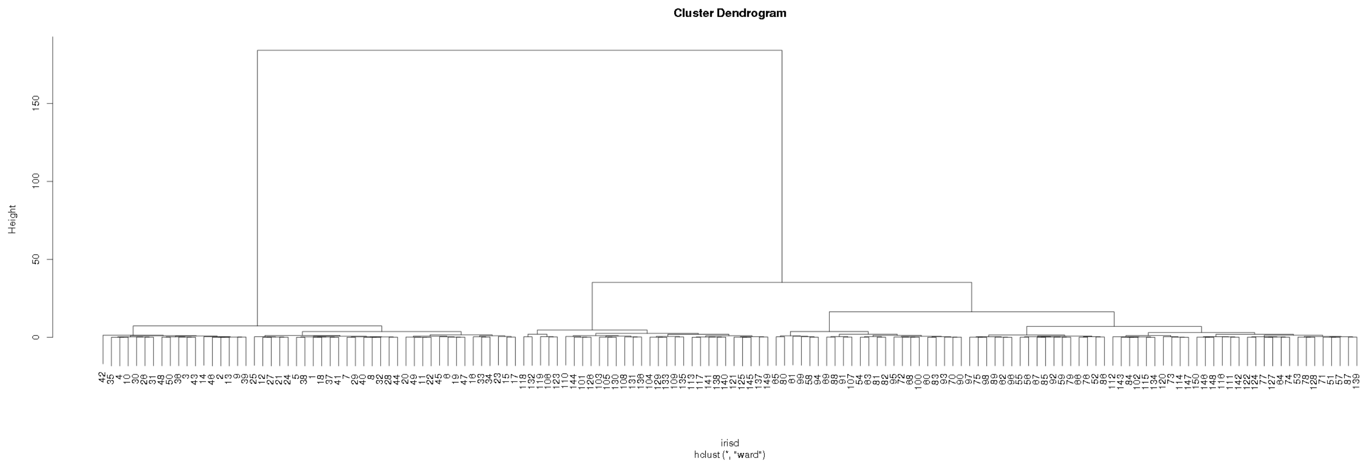
We keep going, combining clusters until everything is one big happy cluster.

The question is, how many clusters do we really have?

Frequently we follow this up and arrange everything into a “distance” tree.

It might be easiest just to do this in R to see how it's different:

```
# We start exactly the same way
# see the explanations above if you're not quite sure of the first
# few steps

iris
iris2 <- iris[c(2,3)]

# We need to tell R to omit missing values:
# Comment: there are no missing values in the iris dataset, but
# frequently this can cause problems.  It's probably safest just
# to include this statement (it's not always clear why it causes
# problems)

iris2 <- na.omit(iris2)

# Now we do our hierarchical clustering:

irisd <- dist(iris2, method = "euclidean")
# This tells R to calculate all possible distances and to use
# normal (Euclidean) distances.  There are lots of other ways
# to measure distances!

# if you want to see what it looks like, just type:

irisd

# you get a list of every possible pair of distances.
# Now let's cluster:

irisfit2 <- hclust(irisd,method = "ward")
# This does the actual hierarchical clustering.  "ward" is one way
# to do hierarchical clustering (there are lots).

plot(irisfit2)
# This plots the results.  Unfortunately the plot doesn't look
# very nice because we have so many observations (150).
```



Note the branches. The branches "branch" at the distances between the points.

For example, the largest division is at a distance of about 125 and has two branches.

One branch goes to the points that are "red" in the diagram above, the other branch goes to the blue and green points.

(Remember that the colors represent different species).

The second most distant branch separates the green and blue points (= species), but it's not perfect (our clustering can't do a perfect job separating out the blue and green species, but that was also true for K-means clustering.

Concluding comments:

You'll get one of these to practice on the homework - don't worry, it's not difficult if you follow the R-code above step by step.

This is only a very brief introduction to the idea of clustering. Whole text books have been written on this subject!!

But it should be enough so you have some idea of what it's all about.