Data mining: Clustering
How to let data tell you what sub-groups you have

Warning

Clustering methods, and much of expression methods, like to use matrices, which is different from the tidyverse way of thinking about plots.

So, there we will walk quite a lot between base R and tidyverse R...

Subgrouping and classification: Not in any way unique for genomics – classification and reduction methods basically defines natural science

In "The Analytical Language of John Wilkins," Borges describes 'a certain Chinese Encyclopedia,' the *Celestial Emporium of Benevolent Knowledge*, in which it is written that animals are divided into:

1. those that belong to the Emperor,
2. embalmed ones,
3. those that are trained,
4. suckling pigs,
5. mermaids,
6. fabulous ones,
7. stray dogs,
8. those included in the present classification,
9. those that tremble as if they were mad,
10. innumerable ones,
11. those drawn with a very fine camelhair brush,
12. others,
13. those that have just broken a flower vase,
14. those that from a long way off look like flies.

Are the entries here mutually exclusive?

- Up to now we have been looking at pairs of vectors, and plotted those against each other (2D plots)
- But say that we have a table with gene expression for 20000 genes in patients 1-10. We want to know which ones that correlate well
  - How can we plot all of that in a smart way?
  - 11-dimensional plots - not a great idea.

Using numbers to say how similar vectors are

- Since we cannot plot all these vectors, we use numbers instead ,saying how similar the vectors are
- For instance, we could have done

```
> cor(genes_patient1, genes_patient2)
[1] 0.7101906
> cor(genes_patient1, genes_3)
[1] -0.003013128
> cor(genes_patient2, genes_patient)
[1] -0.005726535
# etc...
```

These are _almost_ like distances between the vectors. What is wrong with that analogy?

Real distances

- Distance is the opposite of a similarity score in the sense that similar objects are close to each other
- In math, a real distance measurement requires
  - Distance between a and b is always positive, or 0 of a==b
  - It is symmetric, so a → b is the same as b → a
  - Distances satisfies triangle inequality (not going into that, too mathy)

- cor() gives a similarity score which is can be negative, so it is not an intuitive distance. Also, high correlation means that objects are similar, so the direction is wrong.

What do you mean, close?

- We need to define how we measure distance, or similarity
- Many ways of doing it, with different advantages
- We looked at `cor` before
- Let us start out with the classic, Euclidian, measurement (known from high school), which is dealing with real-world distances

Euclidian, in two dimensions

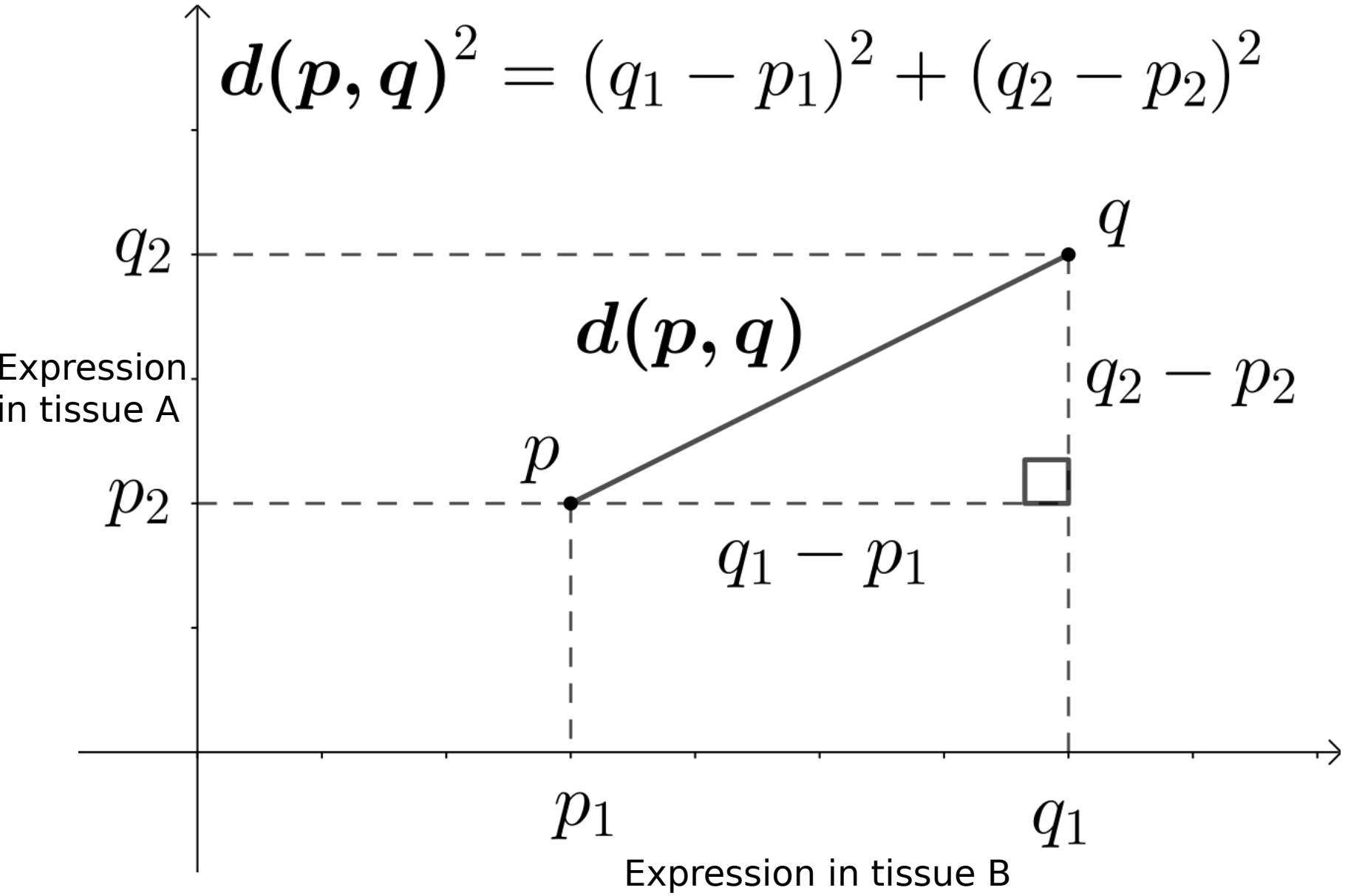
If I have two points on the board, the distance *D* between them is the distance measured by a ruler

We can calculate this instead by knowing the coordinates – we always get a triangle, which takes us back to high-school geometry –

$c^2=a^2+b^2$

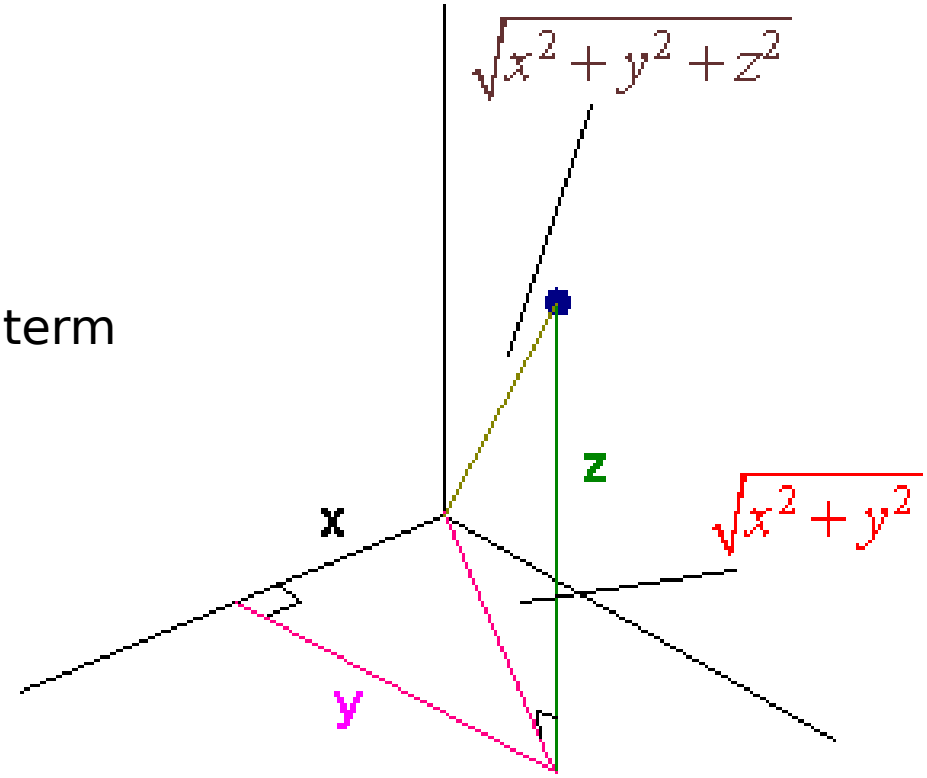
Or...

Each dot is a gene: what is the distance between genes?



hree dimensional euclician distance
nagine that we have 3 tissue (x, y, z)
easurements for two genes

his just expands the math with an extra term



Two dimensions

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2}.$$

Three dimensions

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+(p_3-q_3)^2}.$$

And so forth. It is possible to add as many dimensions as you want, but it is hard to visualize after 3

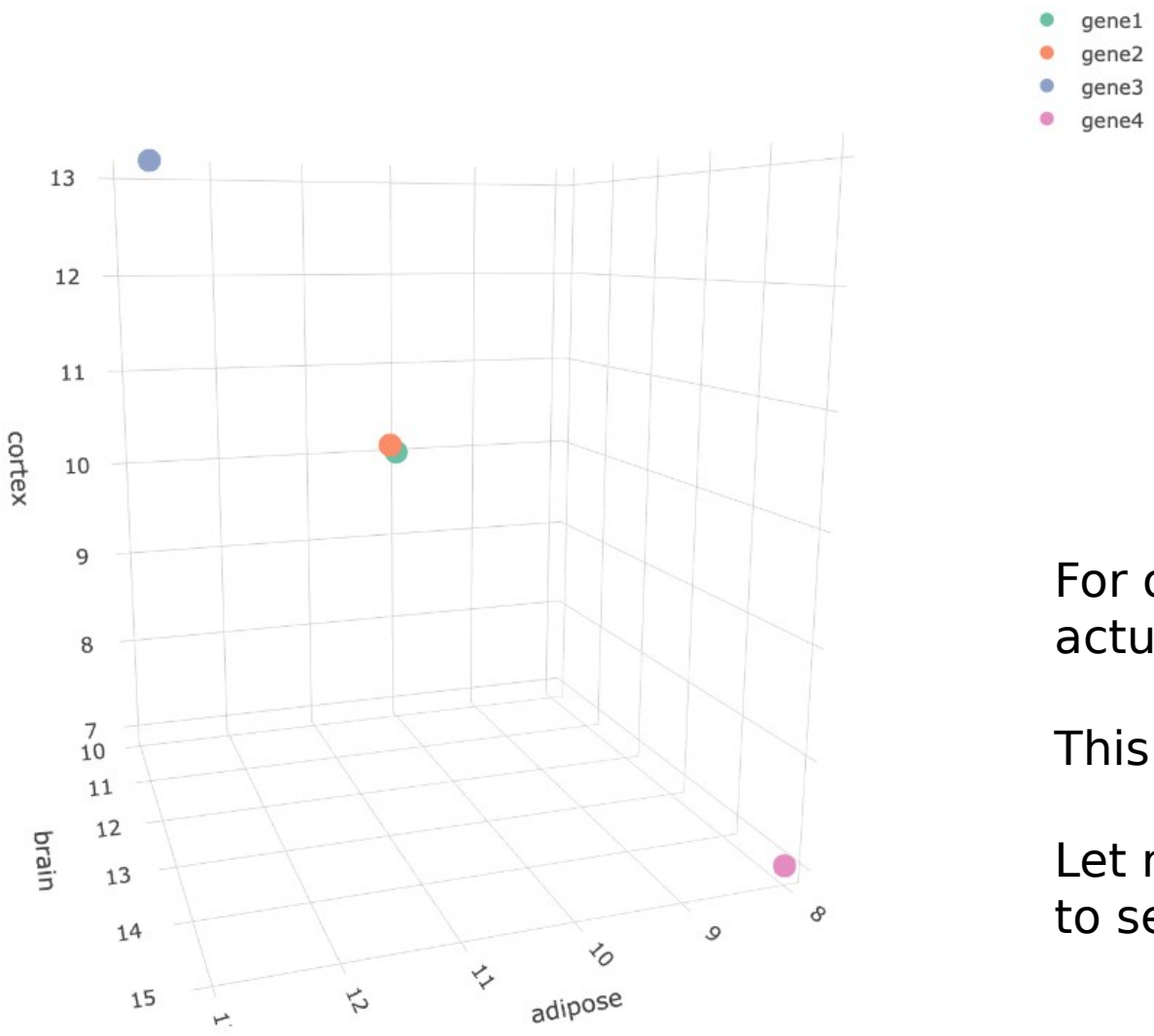
If we have four tissues, it would be a four-dimensional space

Euclian distance in R - example

- Say that we have three tissue measurements for four genes
- This can be described by four vectors of size 3 in a matrix

```
tissue_matrix<- matrix(c( 10, 10, 10, 10.1, 10.1, 10.1,
13, 13, 13, 8, 15, 7 ),
nrow=4, byrow=T, dimnames=list(
c("gene1", "gene2", "gene3", "gene4"),
c("adipose", "brain", "cortex")))

tissue_matrix

   adipose brain cortex
gene1 10.0 10.0 10.0
gene2 10.1 10.1 10.1
gene3 13.0 13.0 13.0
gene4  8.0 15.0  7.0
```

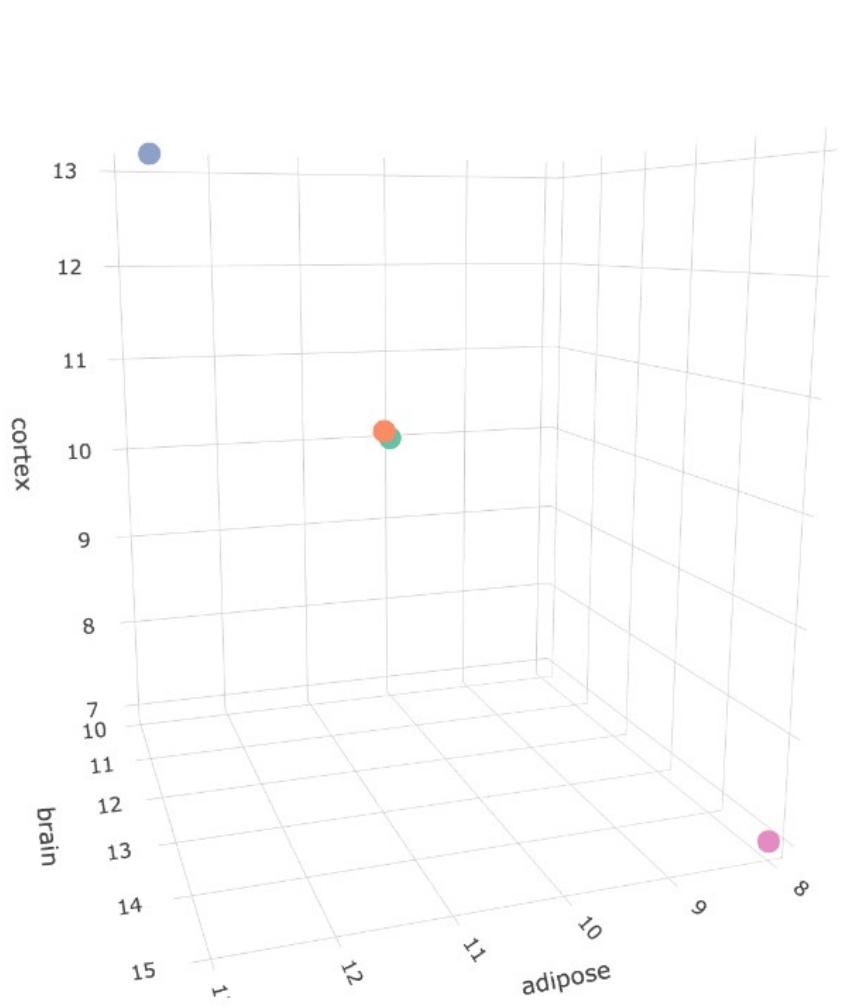
We can view these 4 vectors as coordinates in a three-dimensional space (because, we have three tissues)



For once, a 3d chart actually makes sense

This is actually rotatable

Let me know if you want to see the R code



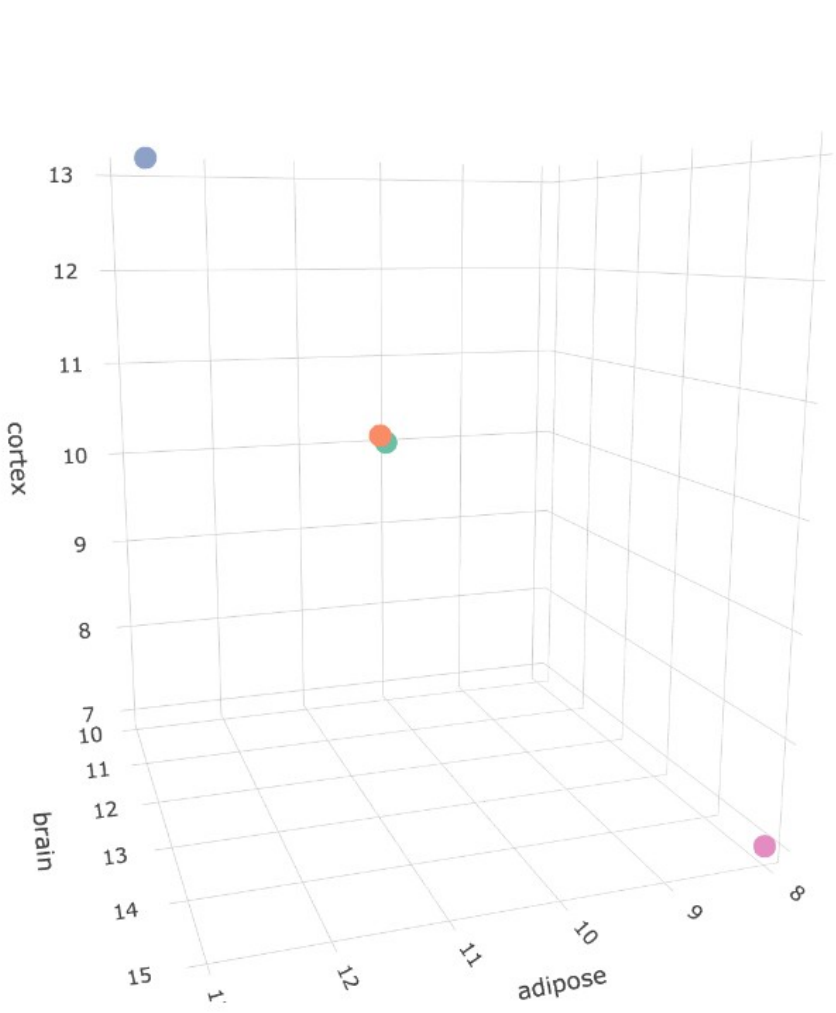
It is intuitive that genes 1 and 2 are very similar and genes 3 and 4 are different from any other genes

another way of viewing this is that genes 1 and 2 are very close in 3d space

Can we quantify it? Yes, by Eucluidian distance

`dist()` calculates the pairwise euclidian distance between all rows in matrices. Which is exactly what we want:

```
tissue_matrix
    adipose brain cortex
gene1 10.0 10.0 10.0
gene2 10.1 10.1 10.1
gene3 13.0 13.0 13.0
gene4  8.0 15.0  7.0

dist(tissue_matrix)
           gene1     gene2     gene3
gene2 0.1732051
gene3 5.1961524 5.0229473
gene4 6.1644140 6.1668468 8.0622577
```

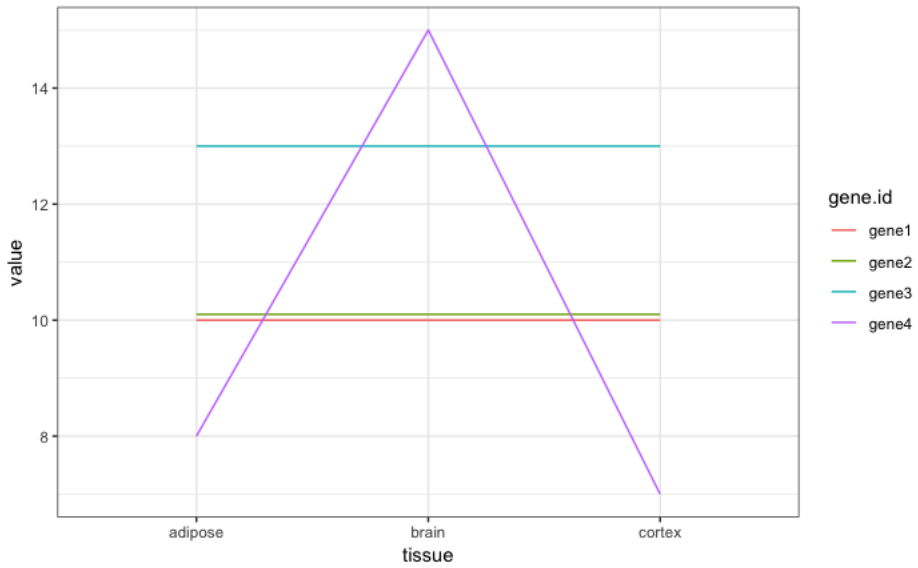



```
tissue_matrix
   adipose brain cortex
gene1 10.0 10.0 10.0
gene2 10.1 10.1 10.1
gene3 13.0 13.0 13.0
gene4  8.0 15.0  7.0

dist(tissue_matrix)
          gene1     gene2     gene3
gene2 0.1732051
gene3 5.1961524 5.0229473
gene4 6.1644140 6.1668468 8.0622577
```

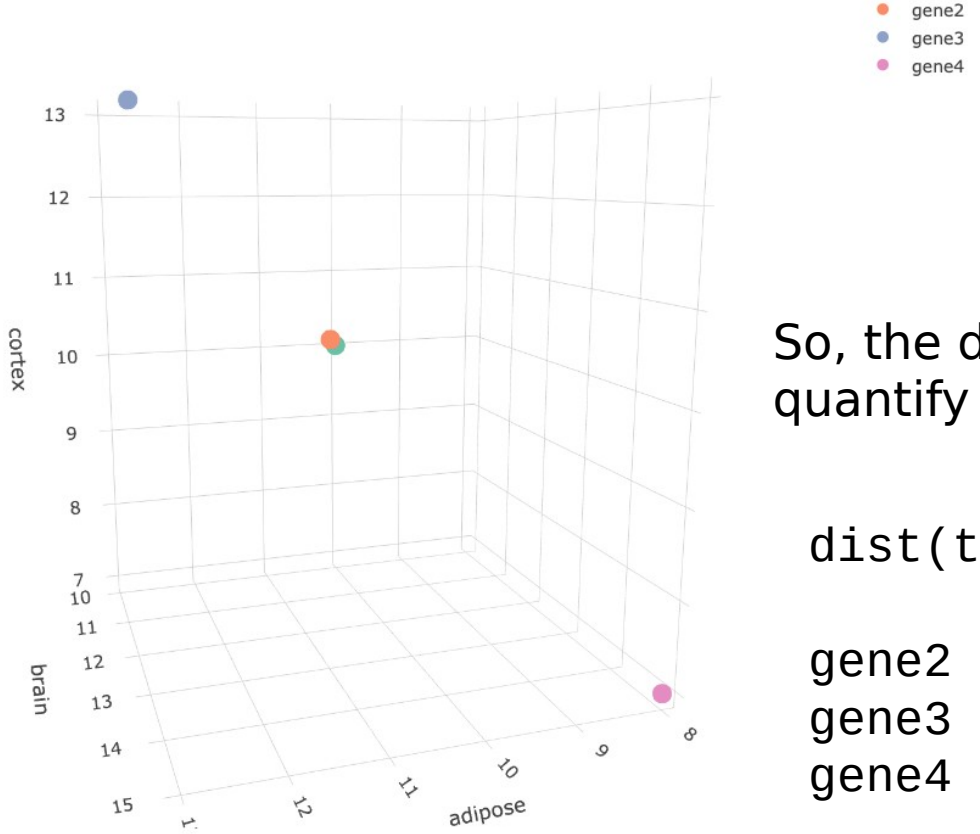
We can also plot the gene expression as a line plot – so each gene becomes a line rather than a dot:

```
tissue_matrix
      adipose brain cortex
gene1    10.0  10.0   10.0
gene2    10.1  10.1   10.1
gene3    13.0  13.0   13.0
gene4     8.0  15.0    7.0
```

```
as_tibble(tissue_matrix) %>%
mutate(gene.id=row.names(tissue_matrix)) %>%
gather(key="tissue", value="value", -gene.id)%>%
ggplot(aes(y=value, x=tissue, col=gene.id,
group=gene.id))+ geom_line()+theme_bw()
```

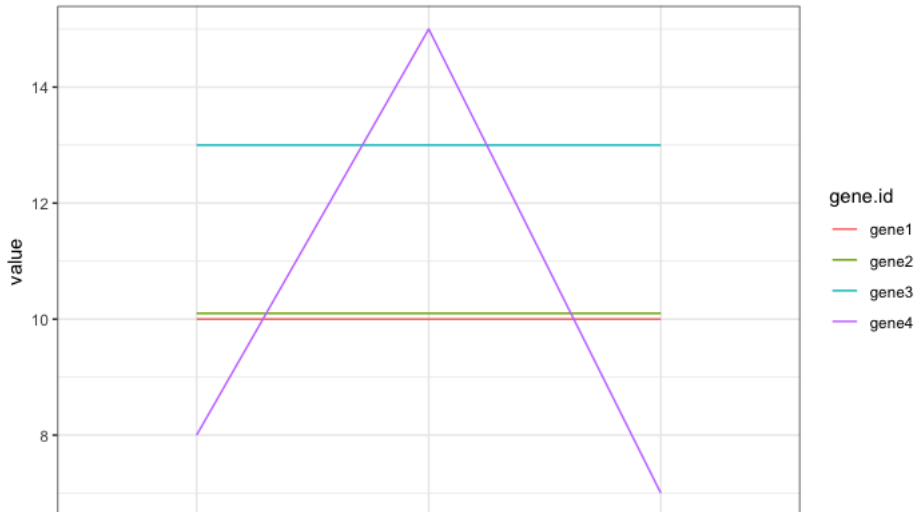


The same pattern comes up again, but in another way



So, the distance measurement is useful to quantify these relationships

```
dist(tissue_matrix)
          gene1     gene2     gene3
gene2 0.1732051
gene3 5.1961524 5.0229473
gene4 6.1644140 6.1668468 8.0622577
```



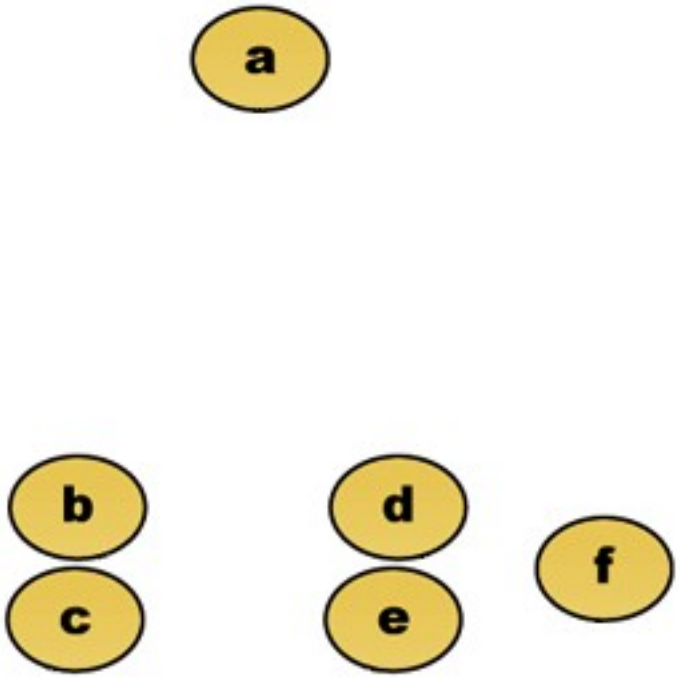
Wehn we get more than 4 tissues, we cannot visualize the points anymore, but the line plot wil work

Other distance measures – just an orientation

- Manhattan distance – walk in blocks instead of diagonals
- Maximal – the largest difference at any pair of points
- Identity, or edit distance – only makes sense if you compare words or sequences
- 1- R^2 (Pearson) – based on correlation

An example with 6 patients, with distances to each other

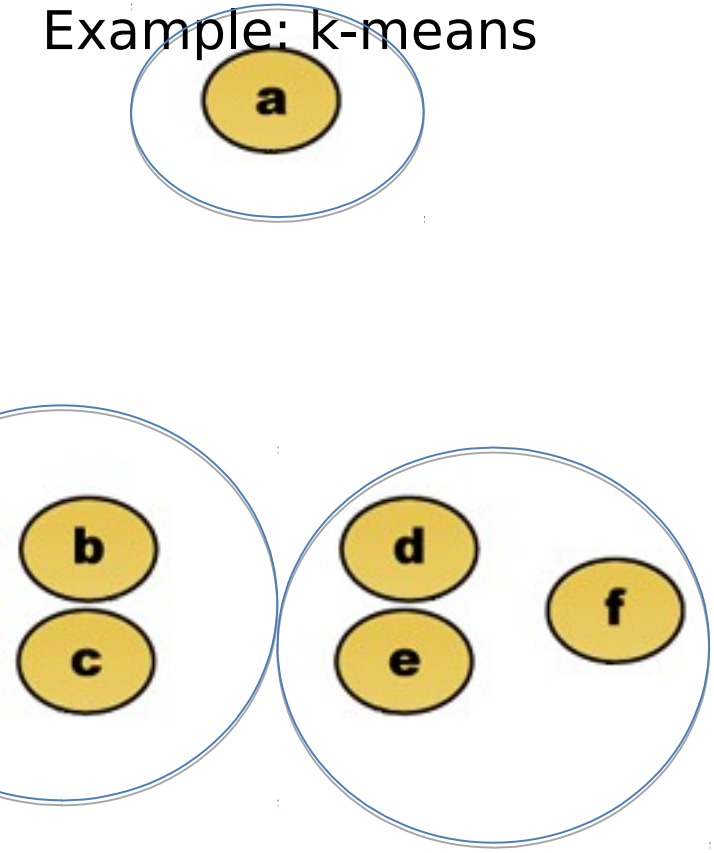
How do we cluster these?



Centroid-based clustering

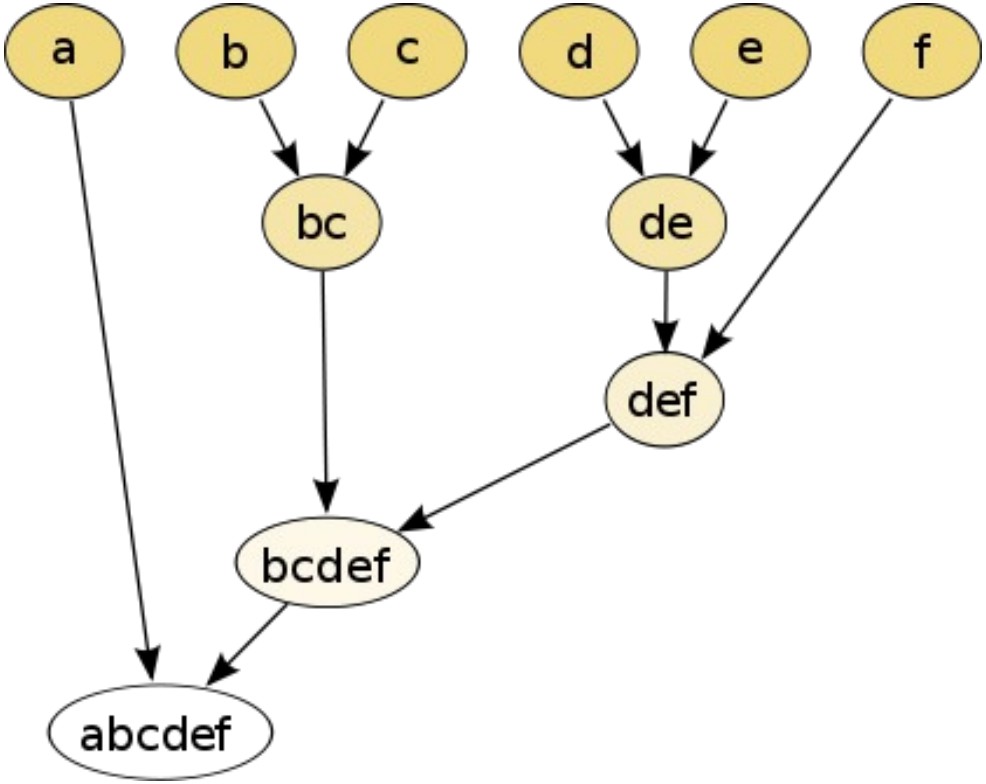
Fundamental idea is to create a “center” that defines sub-groups

Example: k-means



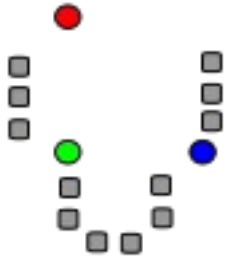
Hierarchical clustering

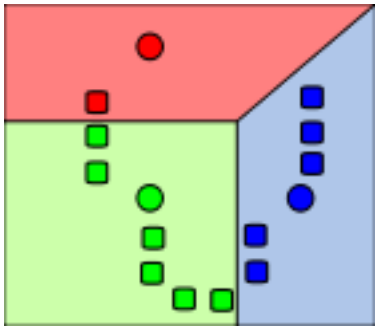
We build a tree that connects all the nodes, where similar nodes will be in sub-trees

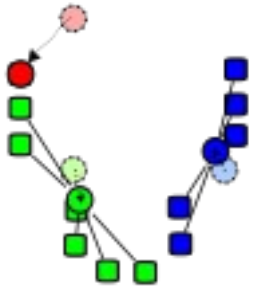


K means

- K-means is a method that is not hierarchical
- It instead makes the “k” most “stable” clusters from the data, where we select “k”
- It uses a random component – this means your results are not always the same. This is due to that finding the optimal solution is very expensive





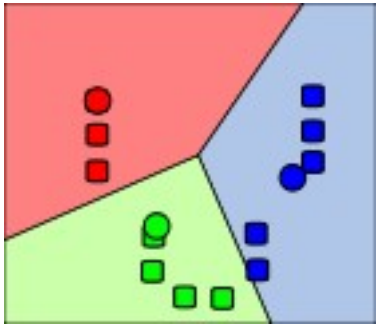


1) Randomly assign k start nodes. We will call these the “means”
Here k is selected to be 3

2) Assign clusters, based on closest mean

3) Update means:
For each cluster, figure out the “centroid” – the cente of the current cluster. This wi be the new “mean”

4) Redo 2-3 until clusters do not change anymore



In action example:
https://www.youtube.com/wat VFG7fd1H30

Let’s try k-means on some patient data

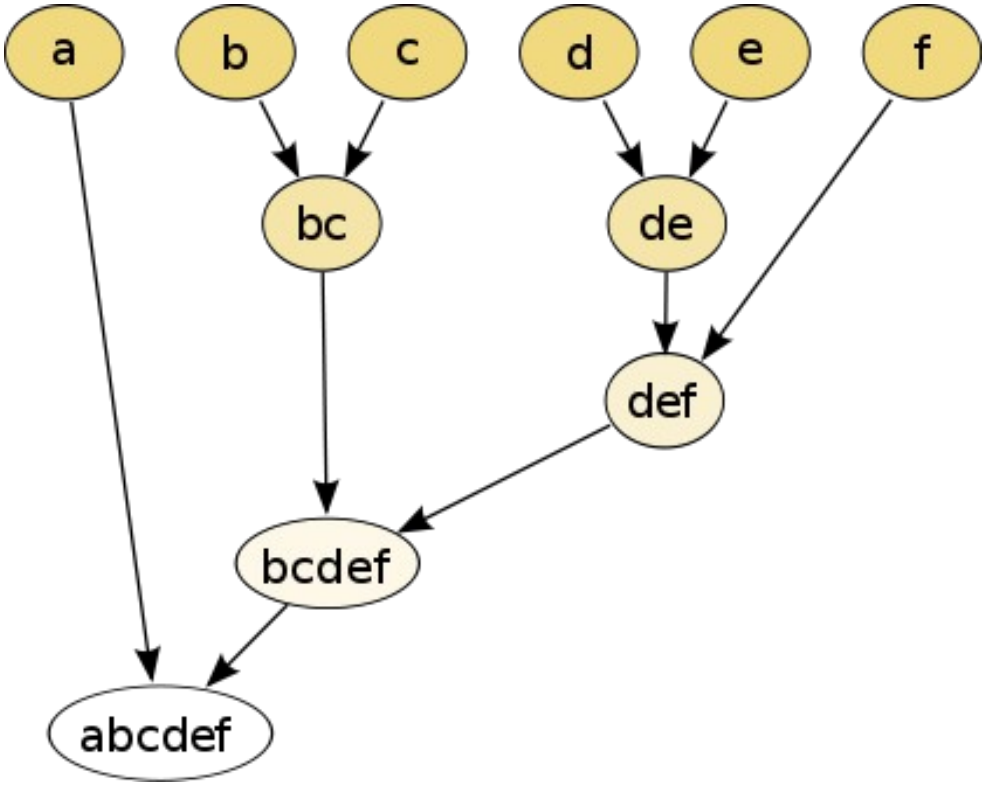
- Do part I: k-means part of the clustering tutorial

Issues with k-means

- How do we select a good k?
  - Some mathematical ways of doing it (% explained variance, and more – not part of this course)
  - Biological checks – do these clusters make any sense at all
  - Practicality – you often want ~4-10 clusters, not 100
- Random component
- Slow for large datasets (there are faster variants)

Hierarchical clustering

- Take the two closest nodes
- Merge them to a new node
- Redo until you have one node left



Back to business: building trees/dendrograms by clustering stuff

- So, we have selected a way to say how close A and B and C are to each other.
- How to make a “tree”?

Agglomerative hierarchical clustering

- We start with every gene in a separate cluster
- We keep merging the most similar pairs of data points/clusters until we have one big cluster left
- Merged points will make a new point – if we merge two points, these points will be “removed”
- Continue until we only have a single point left

To make a tree we need

- 1) Distances, all vs all – the output of is a ew matrix with all-vs-all distances of the columns of my_matrix:

```
my_distances<- dist(my_matrix)
```

- 2) A method to construct and plot the hierarchical tree

```
my_tree<-hclust(my_distances)
plot(my_tree)
```

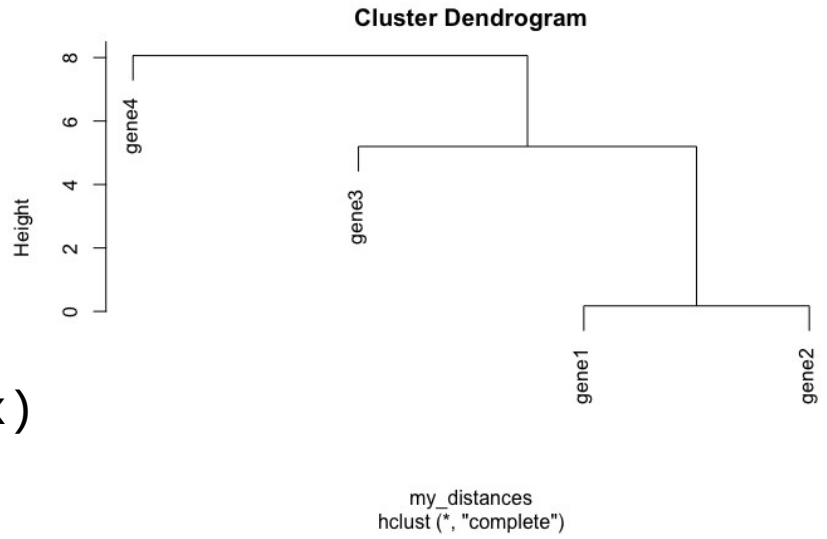
hclust() or agnes()

- R can cluster hierarchically with the hclust method (comes with R), or the more advanced agnes() method, in the cluster package
- hclust requires a distance matrix as those we made before
- agnes can make its own from the data – either way goes
- Can be as simple as saying

```
my_tree <- hclust(my_distance_matrix)
plot(my_tree)
```

Example from before, but as tree

```
tissue_matrix

    adipose brain cortex
gene1 10.0 10.0 10.0
gene2 10.1 10.1 10.1
gene3 13.0 13.0 13.0
gene4  8.0 15.0  7.0

my_distances<- dist(tissue_matrix)
my_tree<-hclust(my_distances)
plot(my_tree)
```

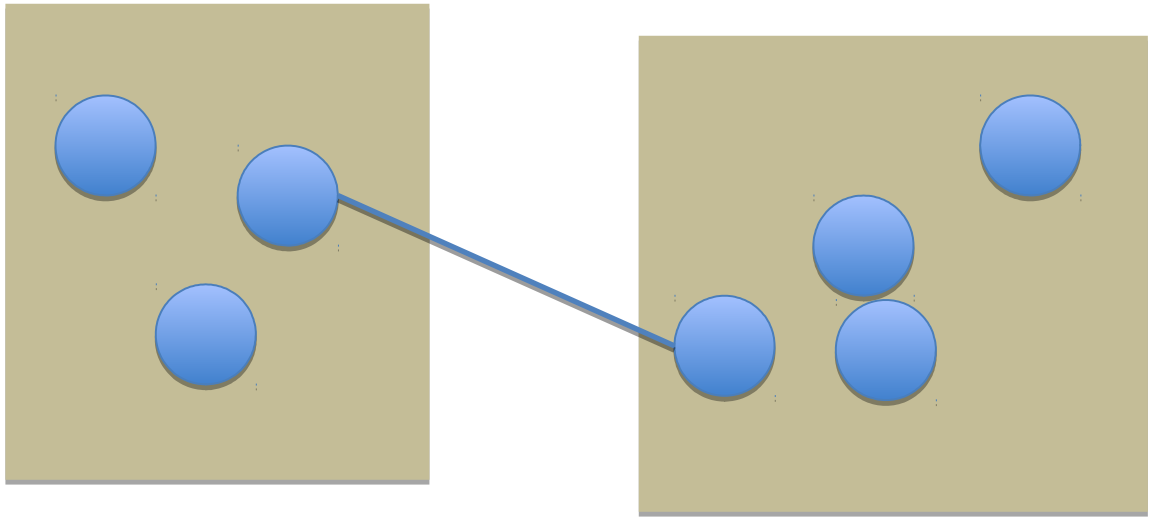


Agglomeration

- How do we merge points? What is the distance to the “child” nodes?
- How can we compare two merged points?
- Three examples:
  - Single, complete and average linkage

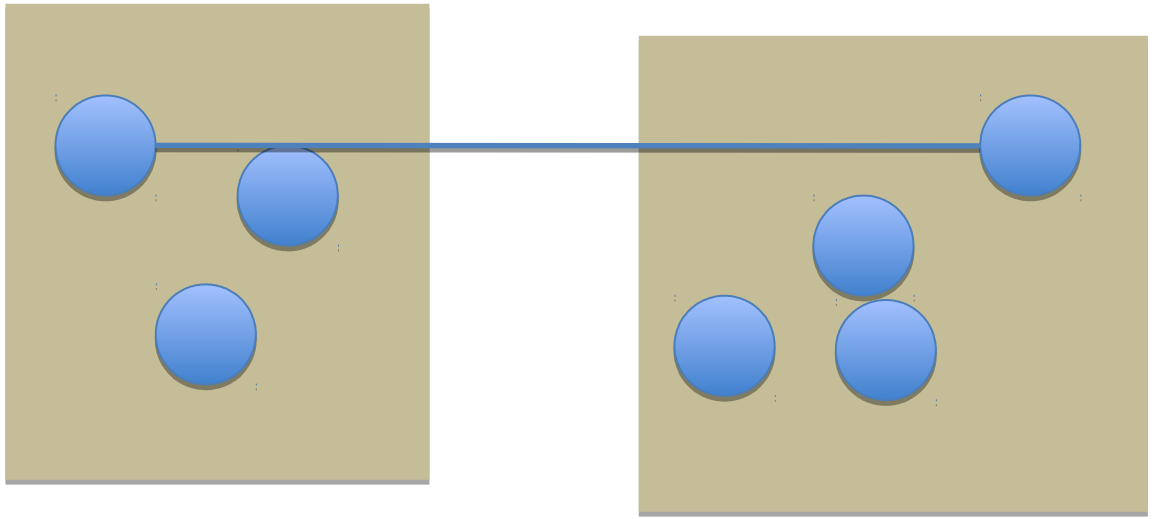
Single linkage

The distance between two merged points is the smallest of all the pair-wise distances, counting all the original data points inside each merged point



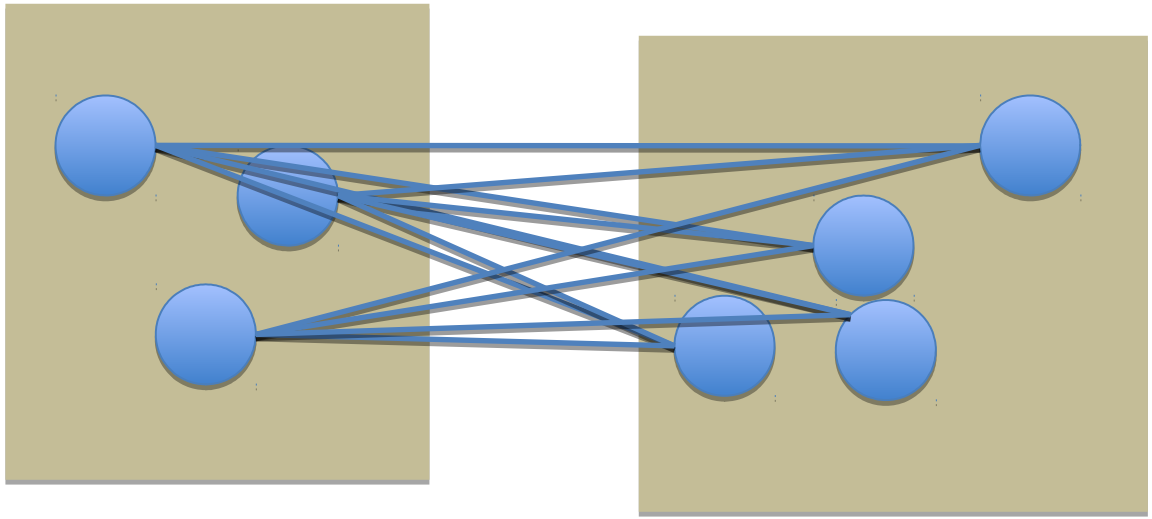
Complete linkage

The distance between two merged points is the LARGEST of all the pair-wise distances, counting all the original data points inside each merged point



Average linkage

The distance between two merged points is the MEAN of all the pair-wise distances, counting all the original data points inside each merged point



- Lets try making trees:
- Do part II, Hierarchical clustering part of the tutorial

Heatmaps

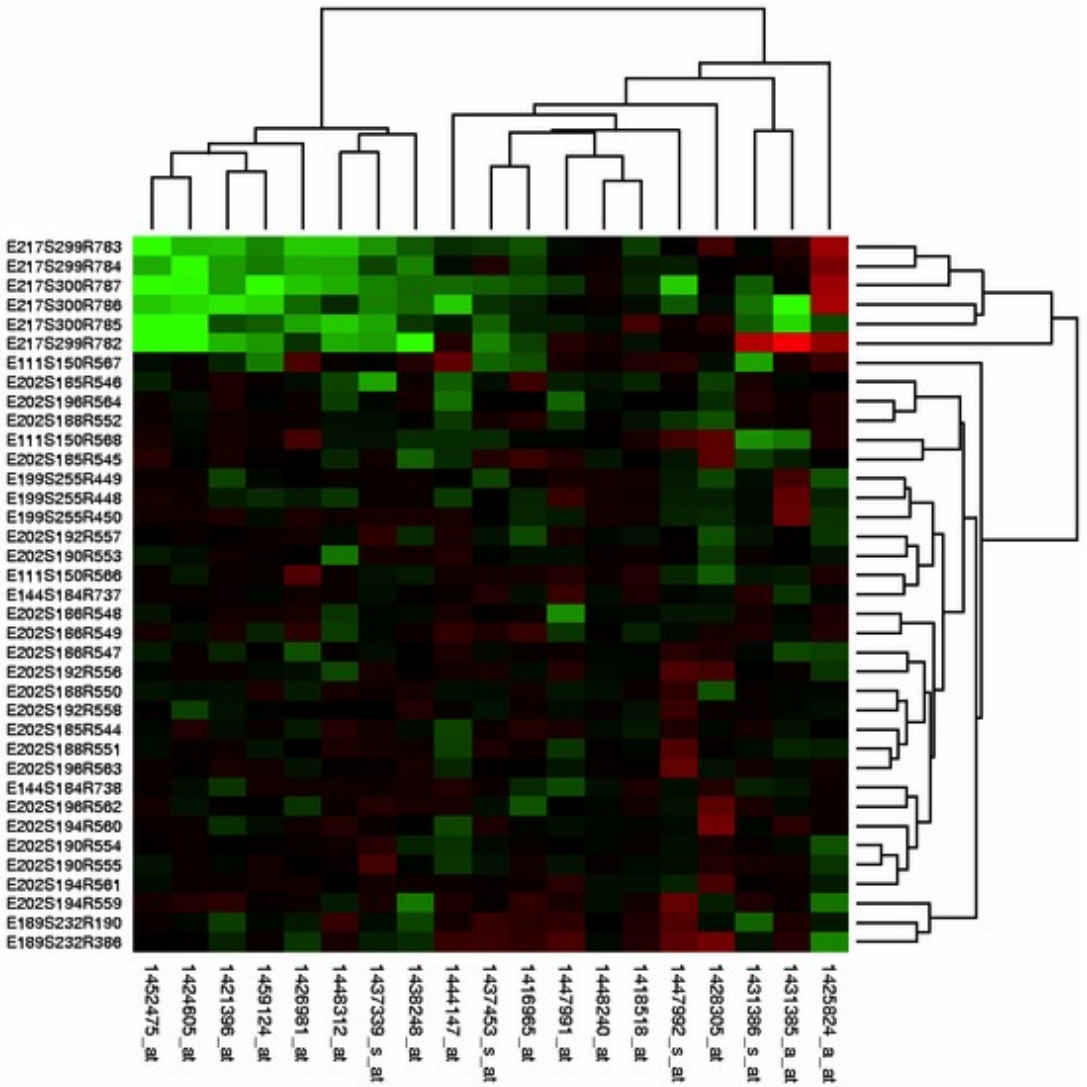
- Heatmaps is a popular way of visualizing gene expression, but can be used for a lot of things
- It uses two concepts:
  - A “false color plot”, showing genes as rows and time as columns, where the color will indicate the expression
  - Hierarchical clustering, which puts
    - genes with similar expression patterns close to one another
    - Time points which are “close” in terms of expression

A typical heatmap

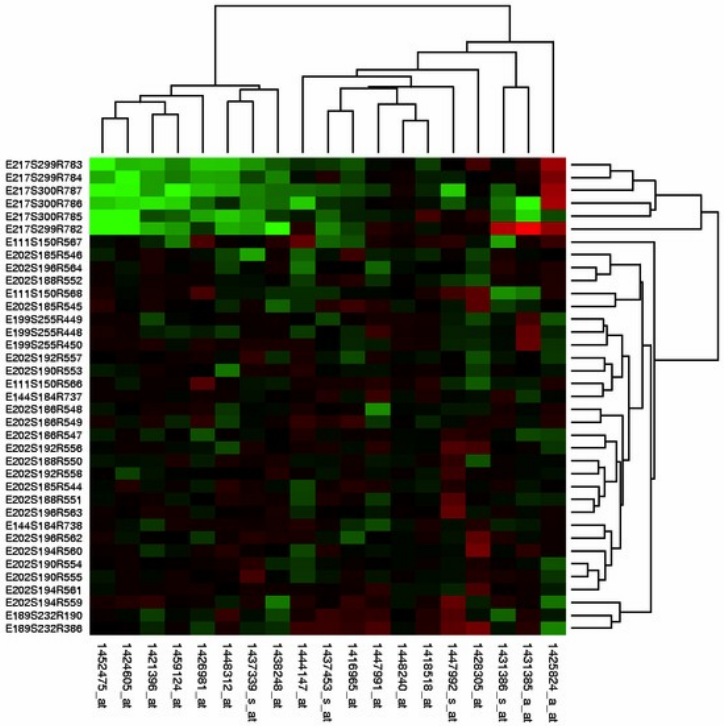
Green=low expression
Red= high expression
Black= medium

genes

Historically, this was done to mimic the colors on microarrays



Time, or treatment



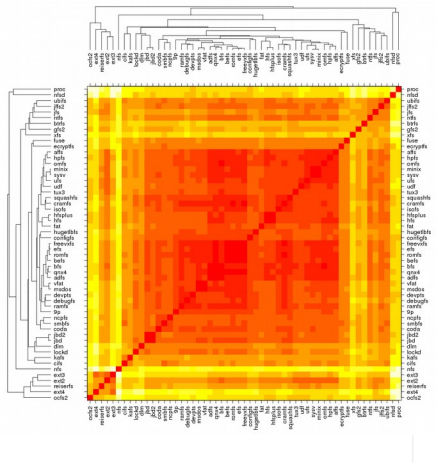
This is a red-green color scheme – very classical, due to that microarrays used to have these two colors

Also VERY non-optimal in terms of human readability!

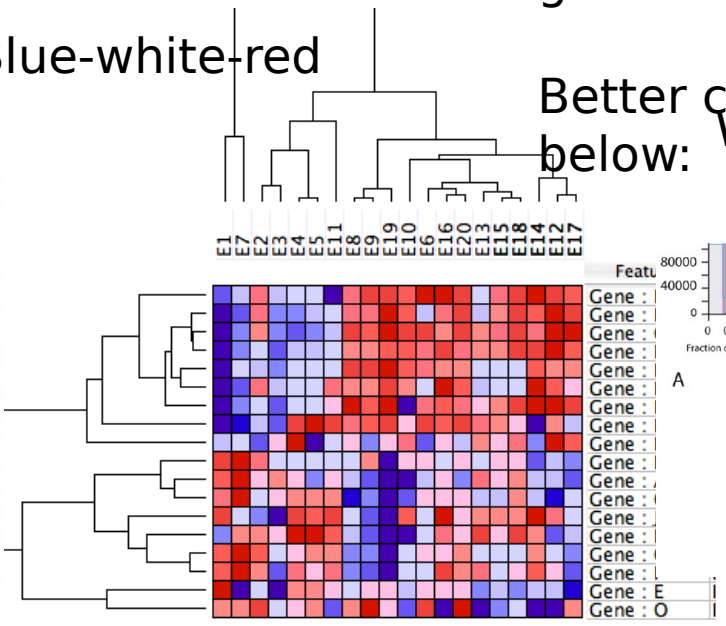
10% of males cannot see the difference between red and green!

Better color schemes shown below:

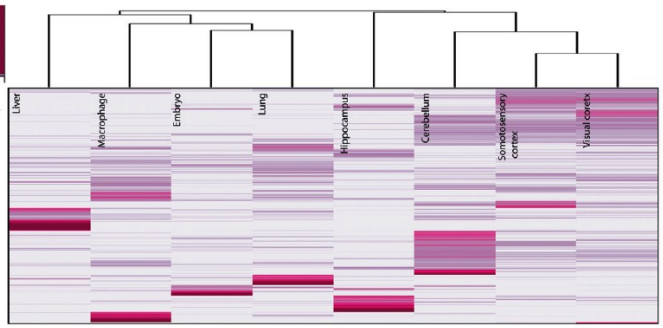
“Heat” colors



Blue-white-red



White-pink



Heatmaps in R

Three different ways – all work fine:

```
heatmap() # basic, in-built
heatmap.2()
# in package gplots: similar, but better
pheatmap()
# much prettier, in package pheatmap. Can also add extra
  ‘cofactors’ as colored rows or columns – more in
  exercise.
```

They work in the same way but has different options

geom_tile in ggplot2 can also be used together with a smooth gradient – although hard to get the trees in (see e.g. https://goo.gl/uhRNSO). pheatmap is often a better choice

Some interesting options (there are many more):

```
heatmap(x, distfun = dist, hclustfun = hclust, col=heat)
```

Heatmap needs to do both a distance calculation and a clustering.
This defaults to

`dist()` for distances – which by default is Euclidian

`hclust()` for clustering – which by default is "complete" merging

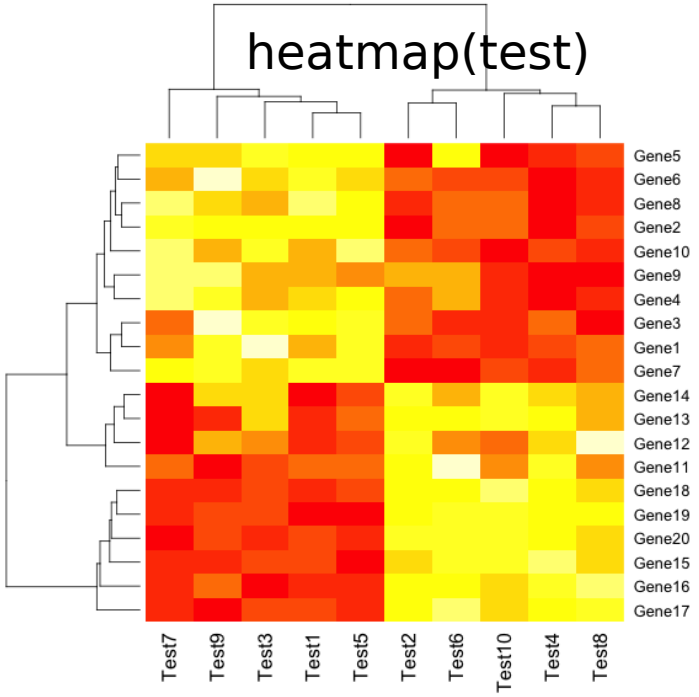
`col` needs to be given a **set of colors**, like `rainbow(10)` or `heat(10)`. `heat()` is the default in vanilla heatmaps
This MAY be what you want, but likely not!

```
# making up some data
> test = matrix(rnorm(200), 20, 10)
> test[1:10, seq(1, 10, 2)] = test[1:10, seq(1, 10, 2)] + 3
> test[11:20, seq(2, 10, 2)] = test[11:20, seq(2, 10, 2)] + 2
> test[15:20, seq(2, 10, 2)] = test[15:20, seq(2, 10, 2)] + 4

> colnames(test) = paste("Test", 1:10, sep = "")
> rownames(test) = paste("Gene", 1:20, sep = "")

> head(test)
         Test1       Test2    Test3       Test4    Test5       Test6    T
Gene1 2.124986  0.07204449 4.598088  0.13706725 3.446507  0.23456636 1.26
Gene2 2.337333 -1.84722206 2.538659 -1.10586680 2.526218  0.14089354 3.30
Gene3 2.718509  1.06618374 3.138857  1.13464246 3.130131 -0.02280634 1.11
Gene4 1.900019  0.25678535 1.411845 -2.92000628 2.883490  1.14154216 4.57
Gene5 2.572145 -0.39352228 2.973321  0.05949788 2.332919  2.41586584 2.07
Gene6 2.654353  0.78715319 1.910087 -0.69274223 2.058266  0.54981091 1.74
          Test10
Gene1 -0.4302813
Gene2  0.5841553
Gene3  0.3219629
Gene4 -1.5217066
Gene5 -0.4349312
Gene6  0.2676397
```

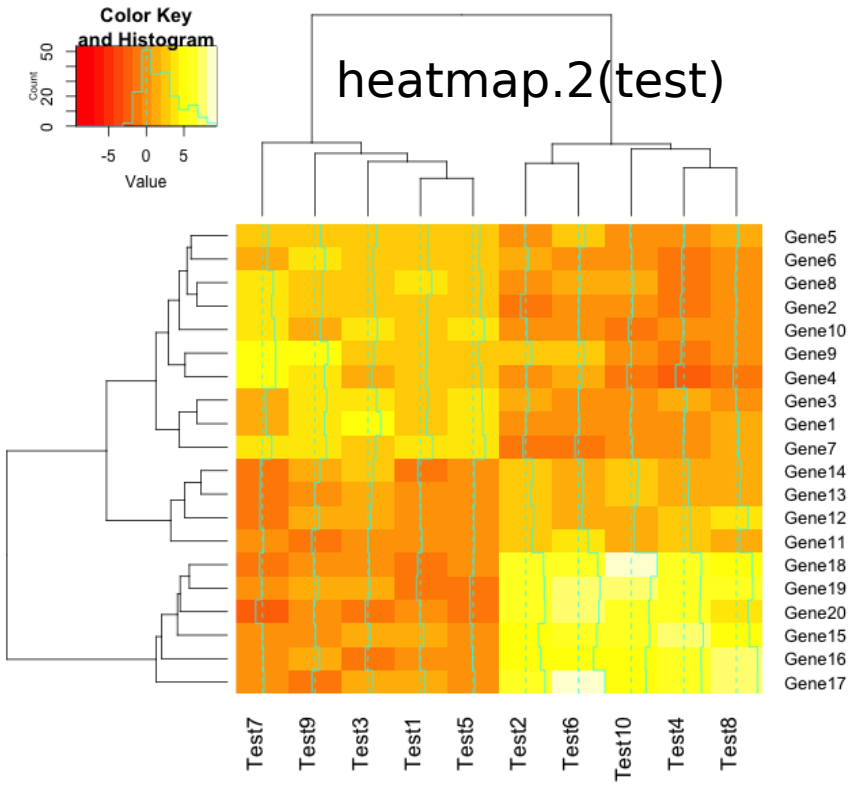
Plotting default heatmaps with the different functions using the data we made using three different heat map methods – all using default settings for respective methods.

```
> heatmap(test)
> heatmap.2(test)
> pheatmap(test)
```

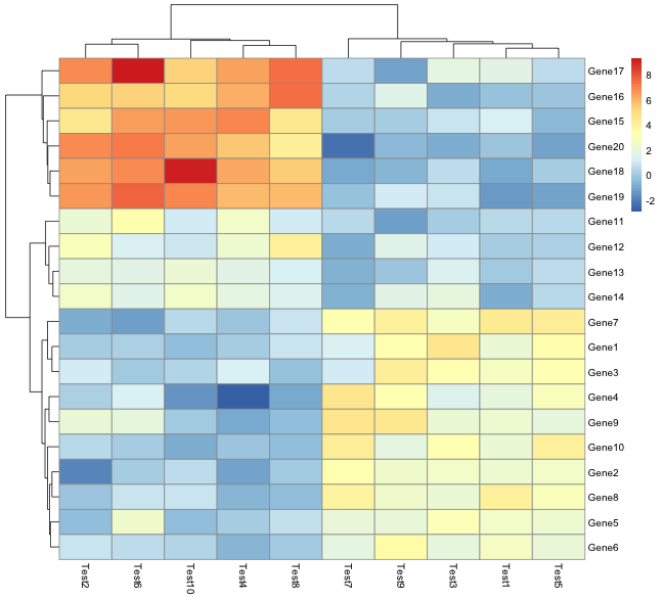
heatmap(test)



heatmap.2(test)



pheatmap (test)



Do the part II: Heat maps in the tutorial .

We will use pheatmap only (which requires a installation of the pheatmap package)

Issues with hierarchical clustering and heatmaps

- Hugely dependent on distance and clustering method
- Easily over-interpreted, especially with heat maps
- Very sensitive to what initial clusters that are made
- Best done together with PCA analysis (next-next lecture), which are complementary – next lecture!
- Is meant as a help to simplify and understand complex data, not a final result...