# Statistical Learning for Data Science: Basic techniques using R

Stefan Zohren

22 February 2016

# Content of this lecture: Unsupervised learning methods

- Introduction to unsupervised learning
- Principal component analysis (PCA)
- Clustering algorithms
  - k-means clustering
  - hierarchical clustering
- Summary

# Supervised learning

So far we looked at regression models, classification models, etc. In all cases we had **labelled data**, consisting of a response variable and p predictors:

$$\text{labelledData} = (\mathbf{y}, \mathbf{x}_1, ..., \mathbf{x}_p)$$

All the above learning algorithms fall under the category of **supervised learning**.

# Unsupervised learning

In this lecture we focus on **unsupervised learning**, where we have **unlabelled data**,

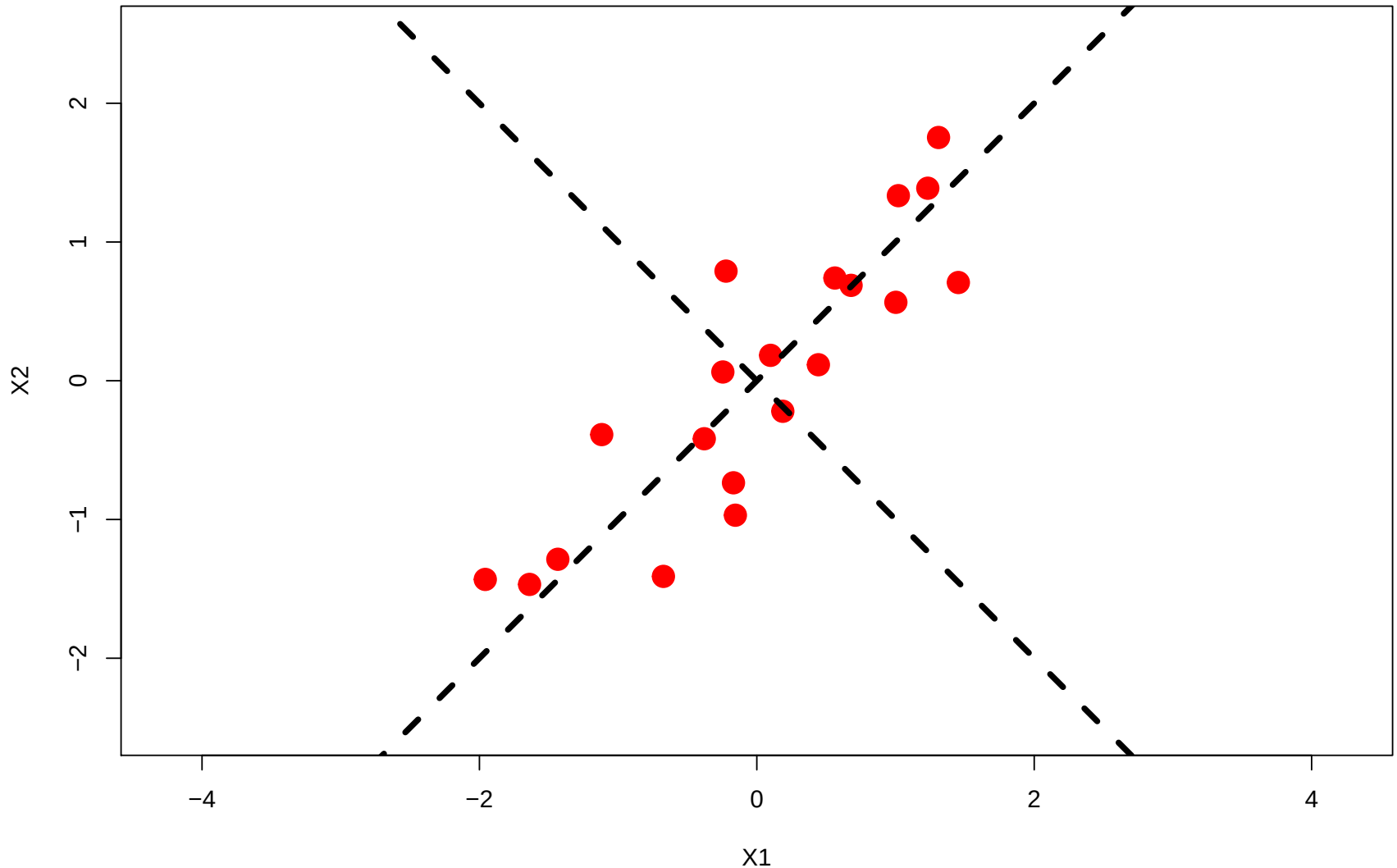
$$\text{unlabelledData} = (\mathbf{x}_1, ..., \mathbf{x}_p)$$

Instead of training a model to predict a certain response, we analyse models which find certain structures within the unlabelled data.

# Unsupervised learning: Models

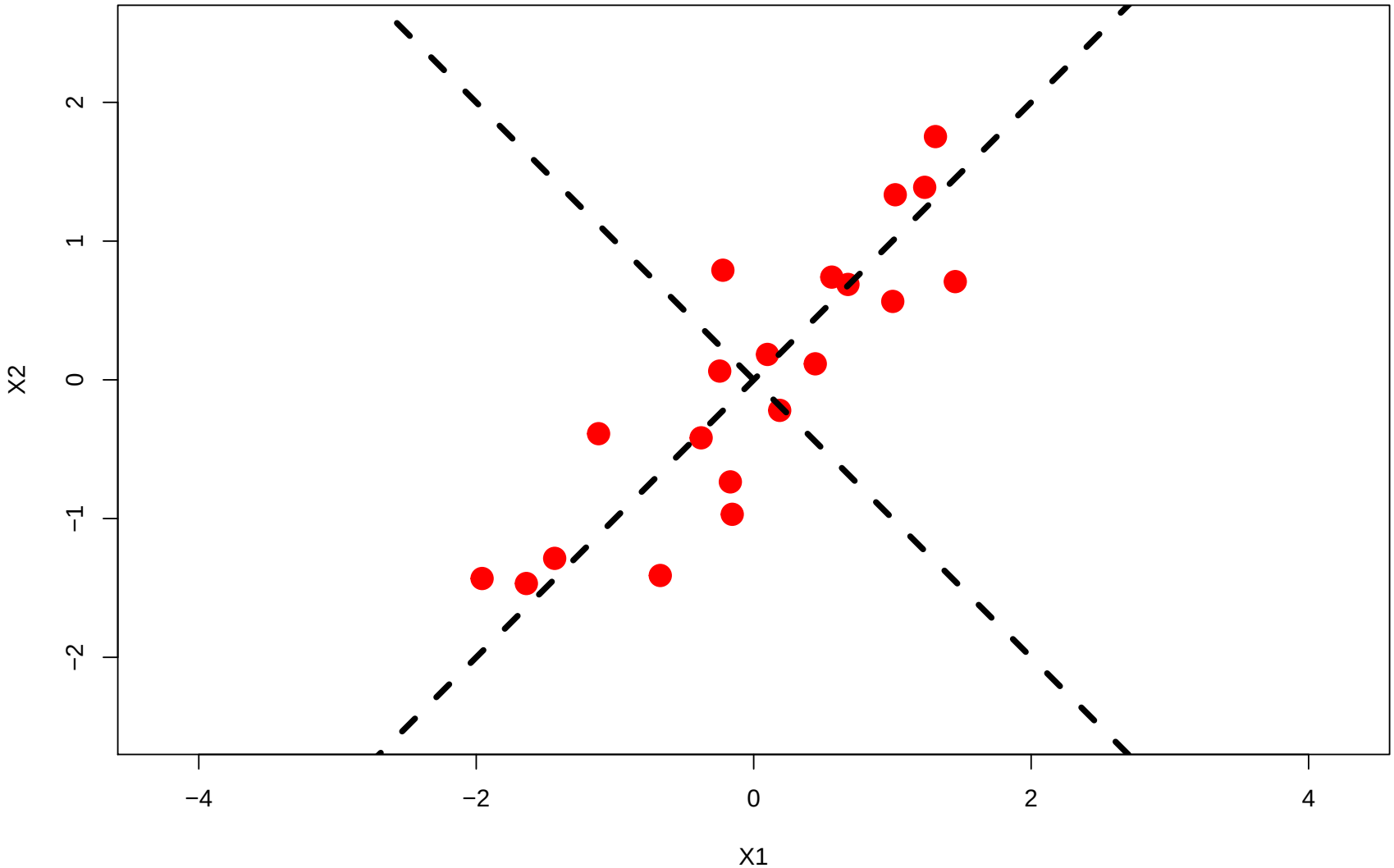
- The first model is **principal component analysis (PCA)** which aims to find a lower-dimensional representation of the data.
- The second class of models are **clustering algorithms**, which aim to cluster points in predictor space, thus being able to label them according to their cluster labels.

# Principal component analysis




Let us look at some toy data of two predictors.

# Principal component analysis




The first principal component corresponds to the direction (given by loading vector) where the data varies most, shown by the dashed line.

# Principal component analysis: Mathematical formulation

Let us denote

$$z_{i1} = u_{11}x_{i1} + ... + u_{p1}x_{ip} = \sum_{j=1}^{p} u_{j1}x_{ij}, \quad i = 1, ..., n$$

where $\mathbf{u}_1 = (u_{11}, ..., u_{p1})$ is the so-called **loading vector**. Here $\mathbf{z}_1$ is the first **principal component**.

Saying that the first principal component corresponds to the loading vector in which direction the data varies most is equivalent to finding the normalised vector $\mathbf{u}_1$ which maximises the sample variance of $z_i$, i.e. $\mathrm{Var}(\mathbf{z}_1) = \frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$.

# Principal component analysis: Mathematical formulation

Mathematically, the loading vector $\mathbf{u}_1$ is thus the solution to the optimisation problem

$$\max_{\mathbf{u}_1} \left( \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{j=1}^{p} u_{j1} x_{ij} \right)^2 \right), \quad \text{subject to } ||\mathbf{u}_1|| = 1.$$

Generally speaking the loading vector of the m-th principal component must be orthogonal to the loading vector of the first m-1 principal components:

$$\max_{\mathbf{u}_m} \left( \frac{1}{n} \sum_{i=1}^{n} \left( \sum_{j=1}^{p} u_{jm} x_{ij} \right)^2 \right),$$

$$\text{subject to } ||\mathbf{u}_m|| = 1 \text{ and } \mathbf{u}_m \perp (\mathbf{u}_1, ..., \mathbf{u}_{m-1}).$$

# Principal component analysis: Example

Let us determine the principal components for the above example using R. The function for PCA is called prcomp:

```
pca <- prcomp(data1)
```

We can look at the loading vectors through the attribute rotations

```
pca$rotation
```

```
##            PC1        PC2
## X1 -0.7071068 -0.7071068
## X2 -0.7071068  0.7071068
```

# Principal component analysis: PVE

We can also obtain the variances of the principal components:

```
pca.var <- pca$sdev^2
```

from which we can calculate the **proportion of variance explained (PVE)** by each principal component:

```
pca.pve <- pca.var/sum(pca.var)
pca.pve
```

```
## [1] 0.9356939 0.0643061
```

Thus we see that the first principal component already explains around 93 % of the variance of the data.

# PCA Case study: NCI60 data

The NCI60 cancer cell line microarray data, is a data set included in the package `ISLR` of your text book An Introduction to Statistical Learning:

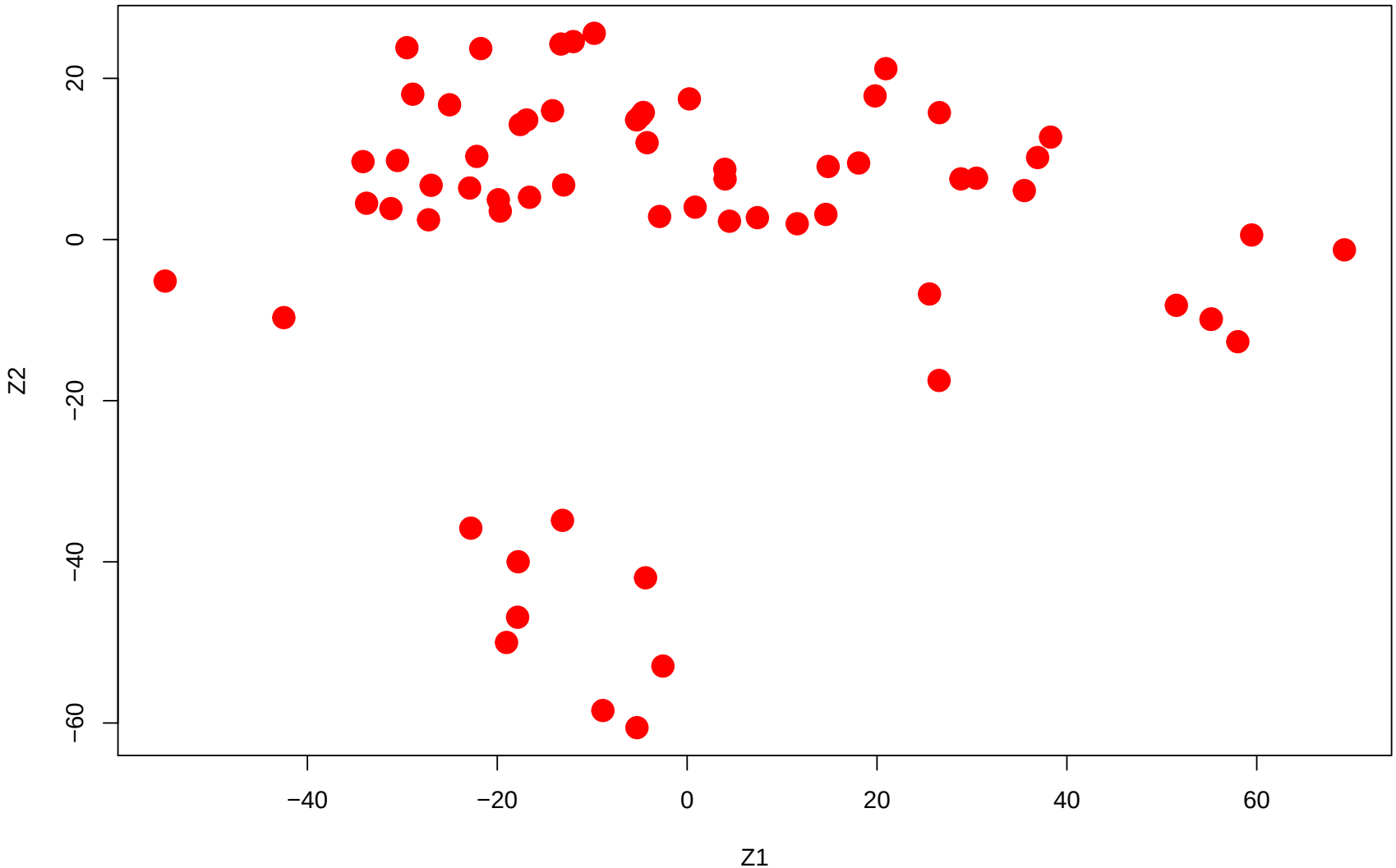
```
library(ISLR)
nci.data <- NCI60$data
dim(nci.data)
```

```
## [1]   64 6830
```

It has n = 64 observations and p = 6830 predictors. We also standardise the data.

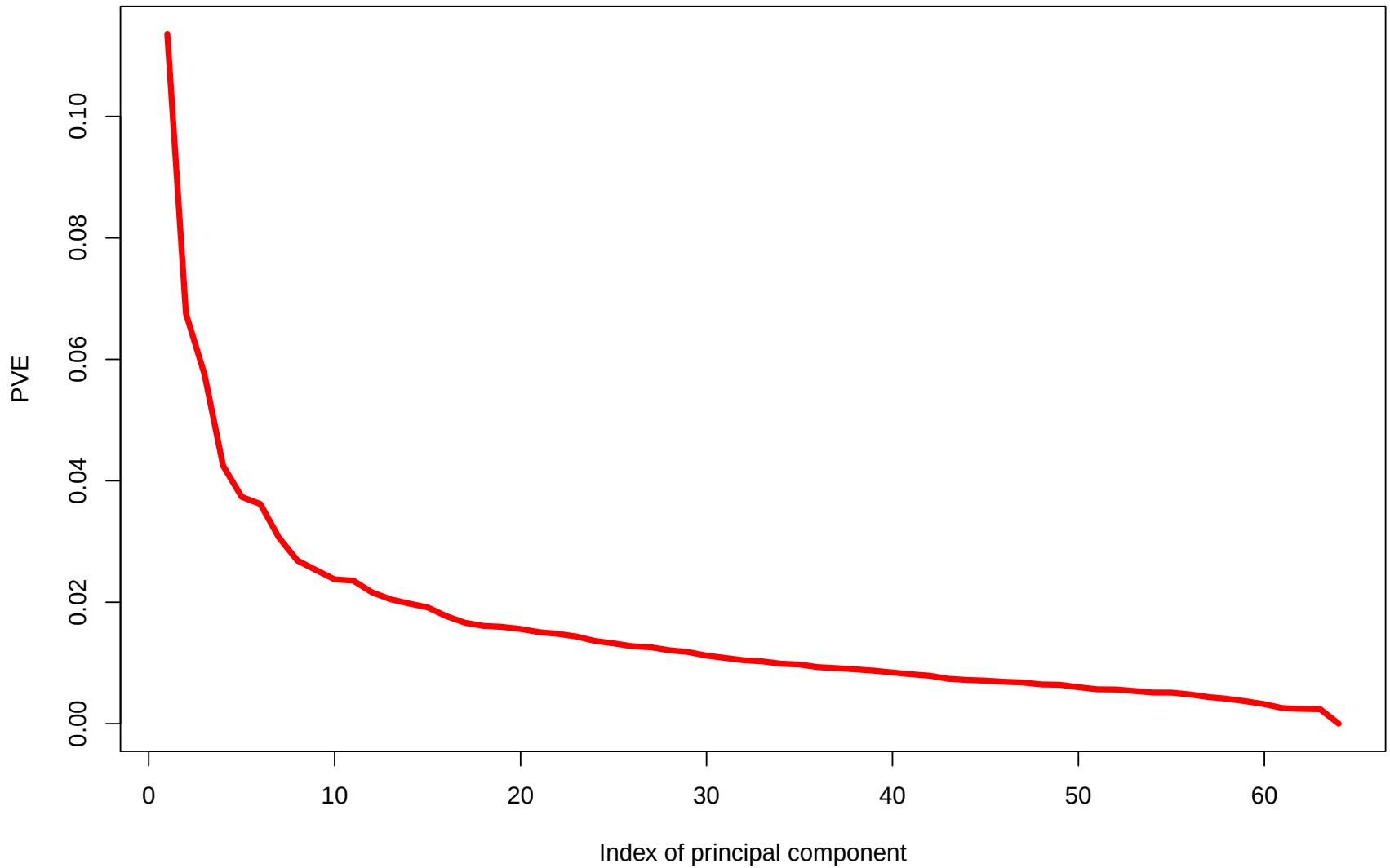
# PCA Case study: NCI60 data

We can do a PCA on the data to obtain a two-dimensional representation of the data for plotting:



# PCA Case study: NCI60 data

Here we also show the plot of the PVE for each of the principal components:



# Clustering algorithms

We now move to **clustering algorithms**, where one aims to **assign labels to data** according to some kind of similarity measure of different points. If some group of points is 'similar', then we assign a given label to them.

In particular we discuss:

- K-means clustering
- Hierarchical clustering

# K-means clustering

- We introduce the within cluster variance, defined by

$$W(C_k) = \frac{1}{|C_k|} \sum_{m,n \in C_k} \sum_{j=1}^{p} (x_{mj} - x_{nj})^2$$

  Here $C_k$ is the set of points in the k-th cluster and $|C_k|$ is the number of points in this set.

- Given a predefined number of clusters K, we aim to find the choice of disjoint clusters whose union is the entire set of points and which minimise the sum over within cluster variances:

$$\min_{C_1,\ldots,C_k} (W(C_1) + \ldots + W(C_K))$$

# K-means clustering: Algorithm

Solving the exact optimisation problem is very difficult, but there exists an approximate algorithm, which works as follows:

- Label the points randomly with K different labels
- Iterate the following until convergence:
  - Calculate the centroid of each cluster (centre of mass)
  - Assign each point to the closed cluster

# K-means clustering: Example and implementation in R

Let us illustrate this with a toy example of simulated data.

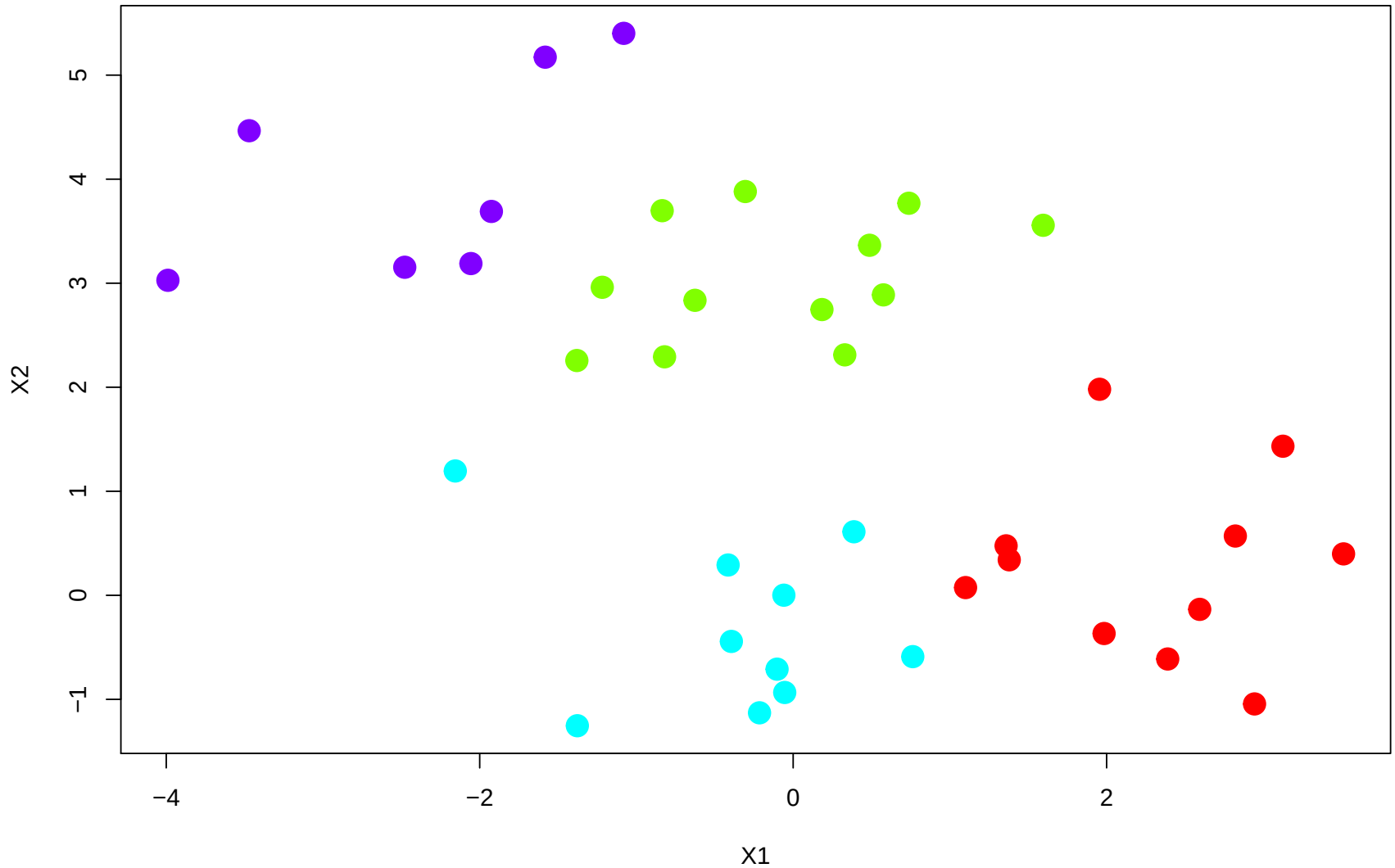
The K-means clustering algorithm is implemented with the function `kmeans`. Let us perform a clustering with K=4 clusters:

```
K=4
clust1 <- kmeans(data2, K ,nstart = 10)
```

We extract the cluster labels of each point using the attribute `cluster`.

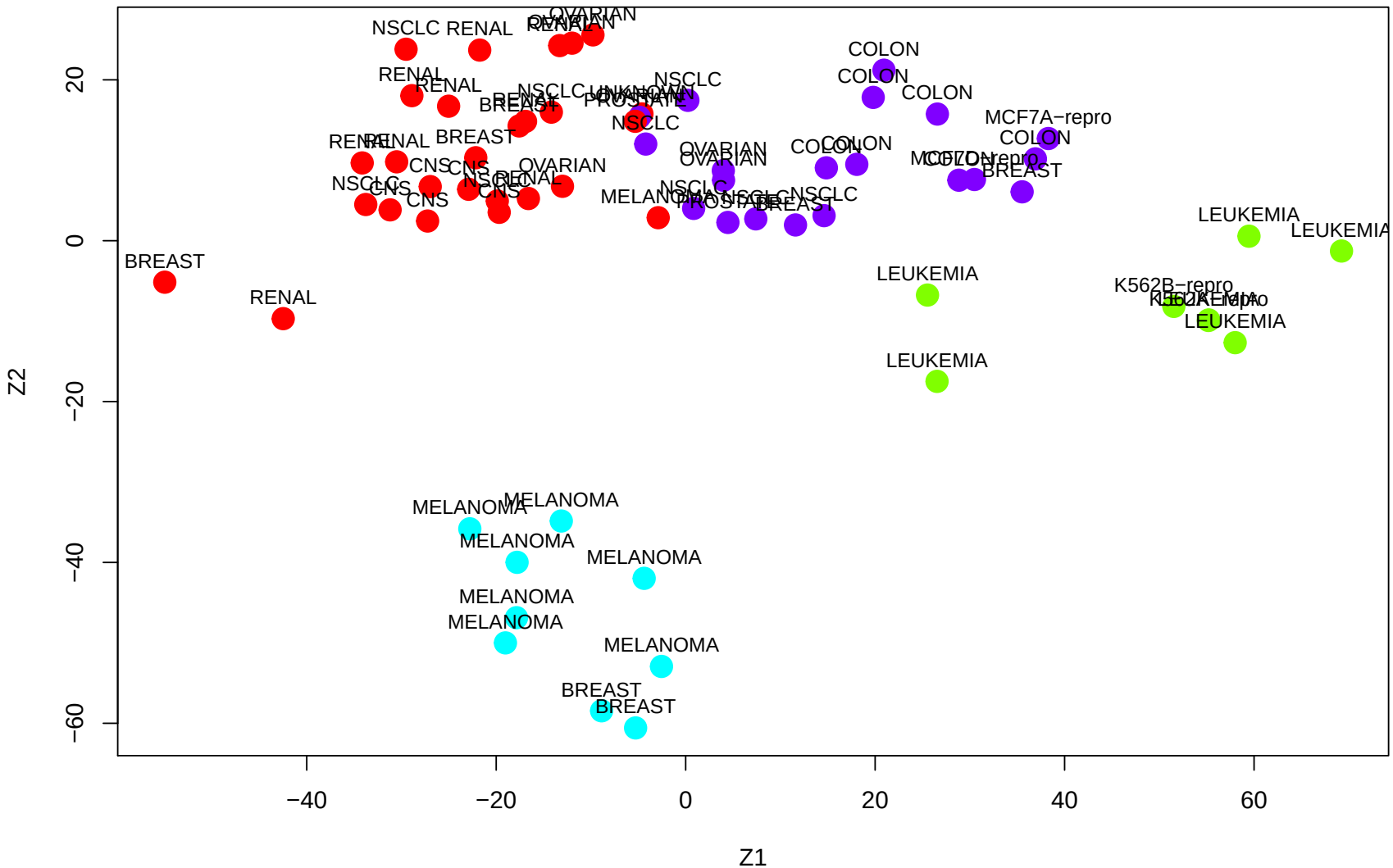
# K-means clustering: Example and implementation in R

Below you see an example of the corresponding plot for K=4.



# Clustering Case study: NCI60 data

We can now also return to the NCI60 data. Shown is a K-means clustering with K=4, plotted on the reduced space of the first 2 PC:



# Hierarchical clustering

Hierarchical clustering, as the name suggests, produces a whole hierarchy of clusters. In particular, it produces clusters of size n, n-1, . . . , 1. In a bottom-up approach starting from each observation being its own cluster and then merging pairs of clusters.

In a nutshell, the **algorithm for hierarchical clustering** works as follows:

- Start with each observation being its own cluster (we have n clusters);
- Given a similarity measure between pairs of clusters (based on distance), merge the pair of clusters which are most similar (we now have n-1 clusters);
- Continue to merge until we have a single cluster.

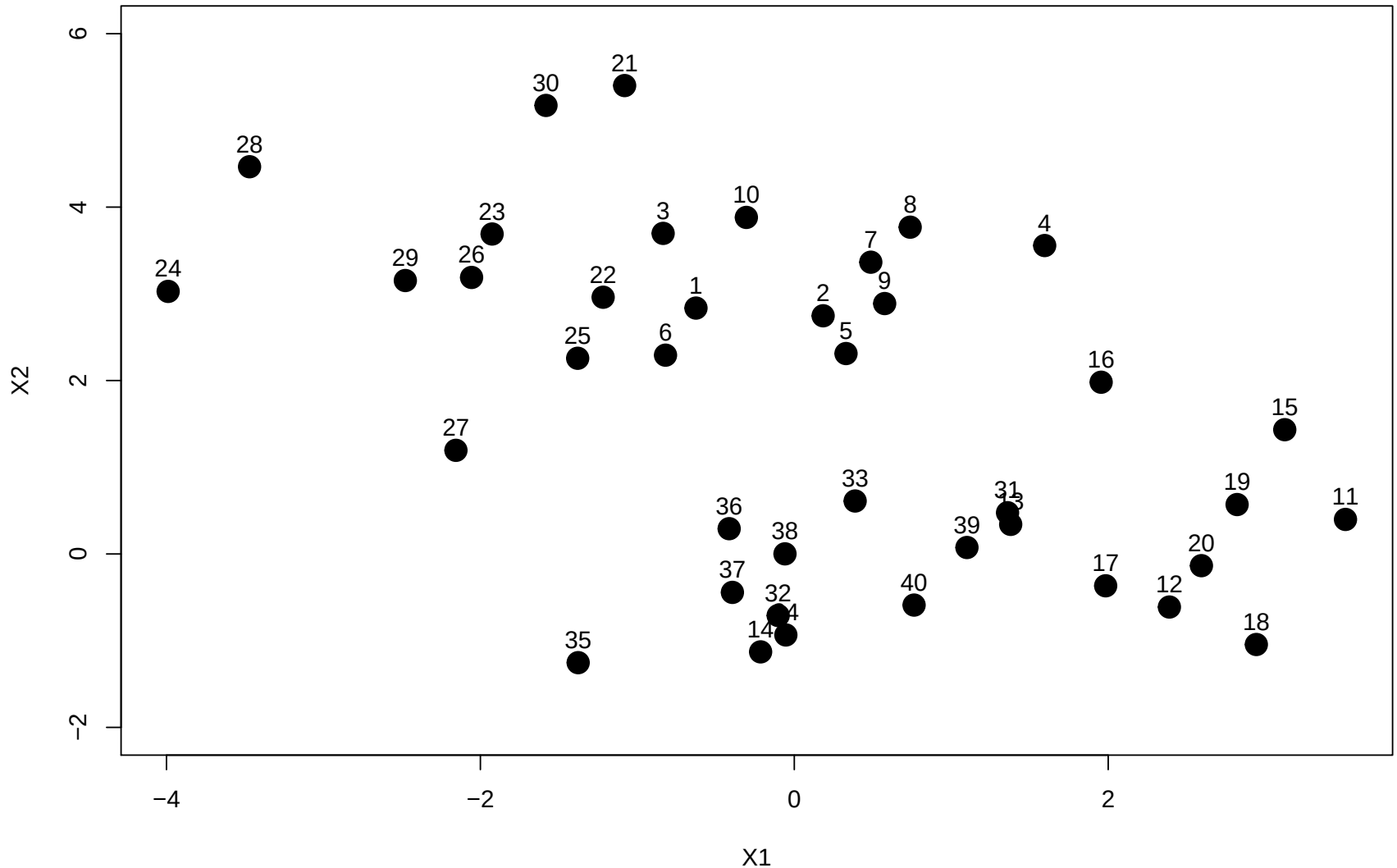
# Hierarchical clustering: Example and implementation in R

Let us return to the toy example of simulated data from the previous section, saved under `data2` and illustrate the above. The hierarchical clustering is performed as follows:

```
clust2 <- hclust(dist(data2))
```

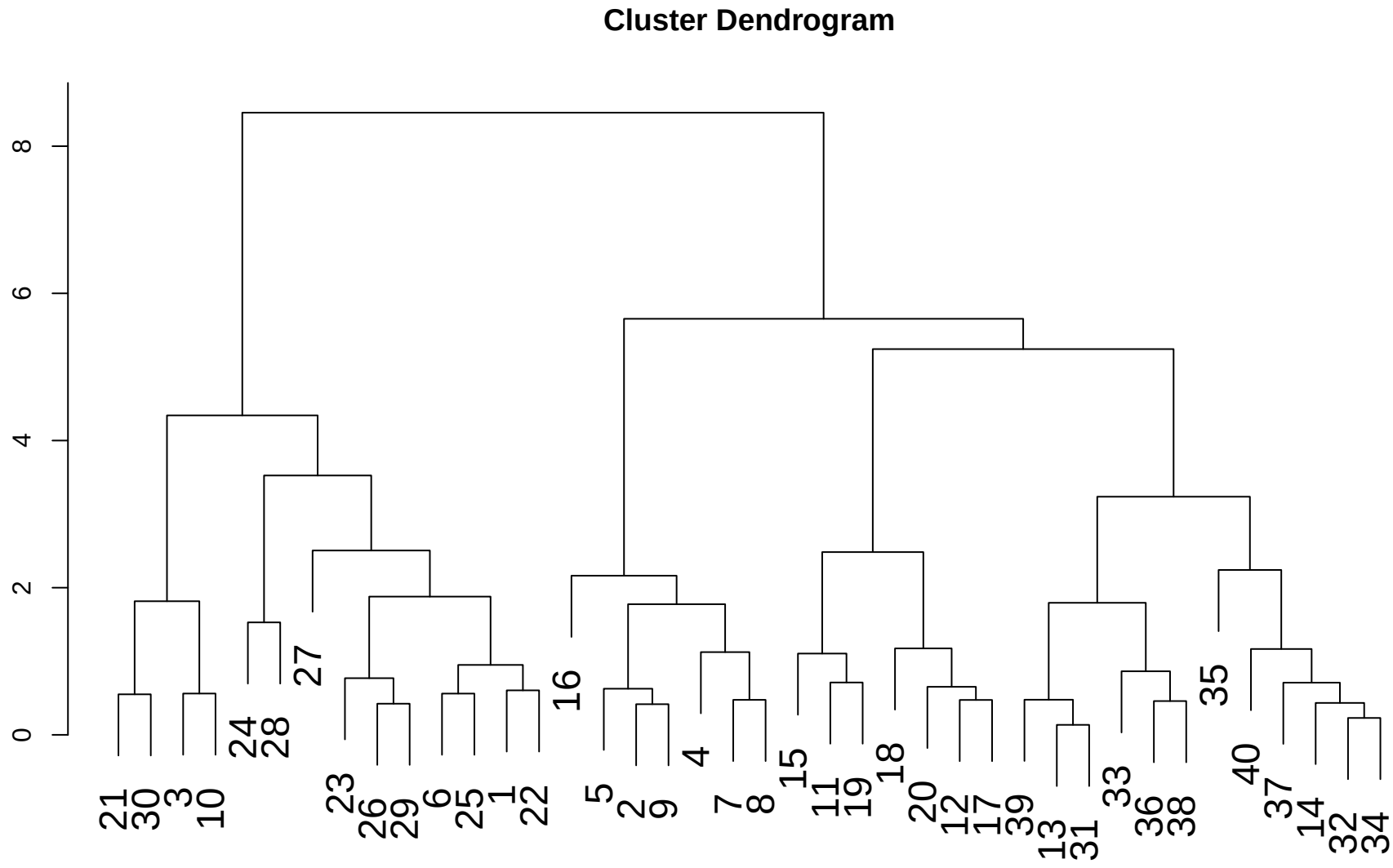
# Hierarchical clustering: Example and implementation in R

Let us plot again the data, now labelling every data point for i=1,. . . ,20.



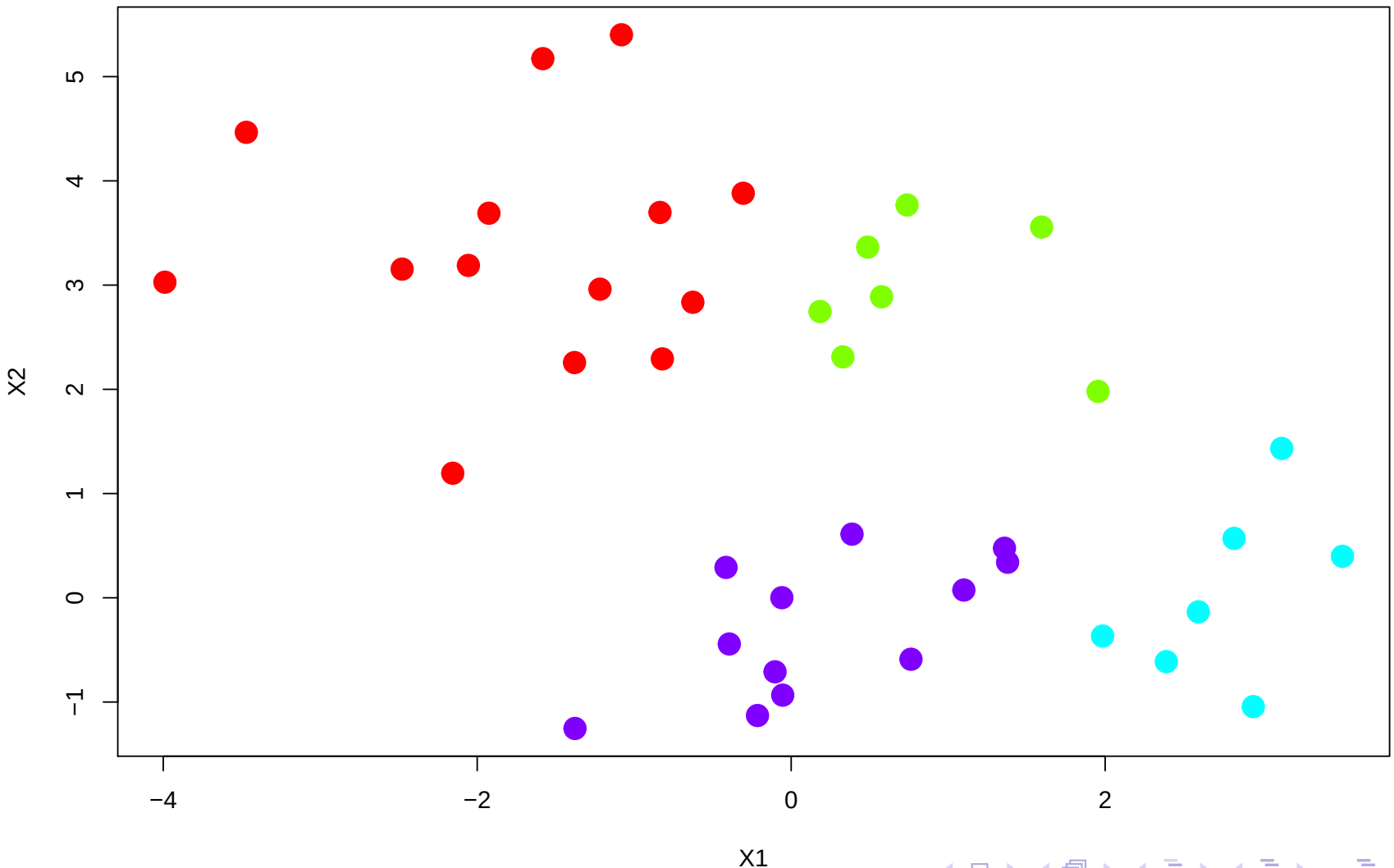
# Hierarchical clustering: Example and implementation in R

We can also plot the **dendrogram** by calling the standard `plot` function on the object created by the `hclust` function:



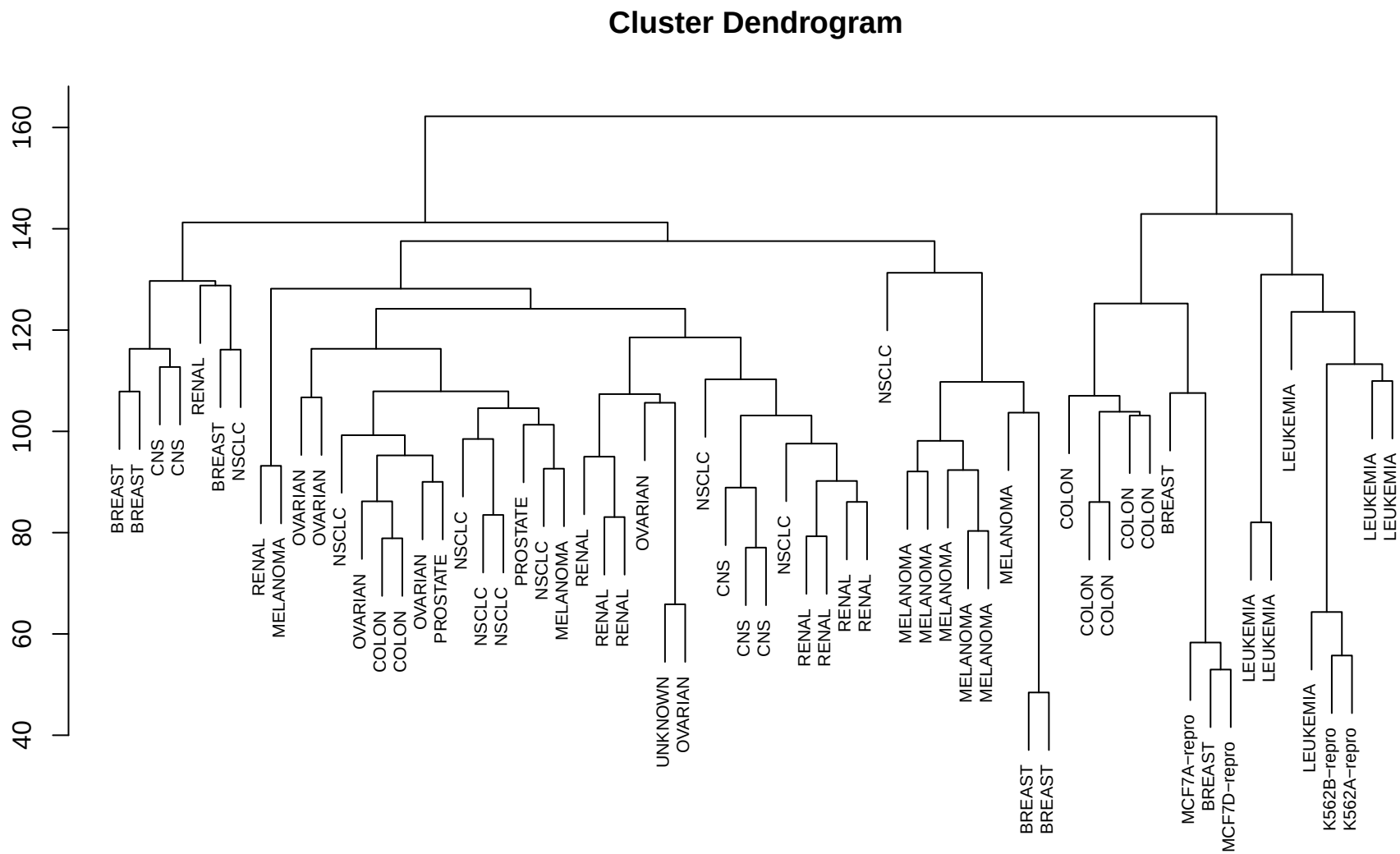
# Hierarchical clustering: Example and implementation in R

To obtain a clustering of a given number of clusters K, we can cut the tree at a level where it has K branches. This is done using `cutree`:



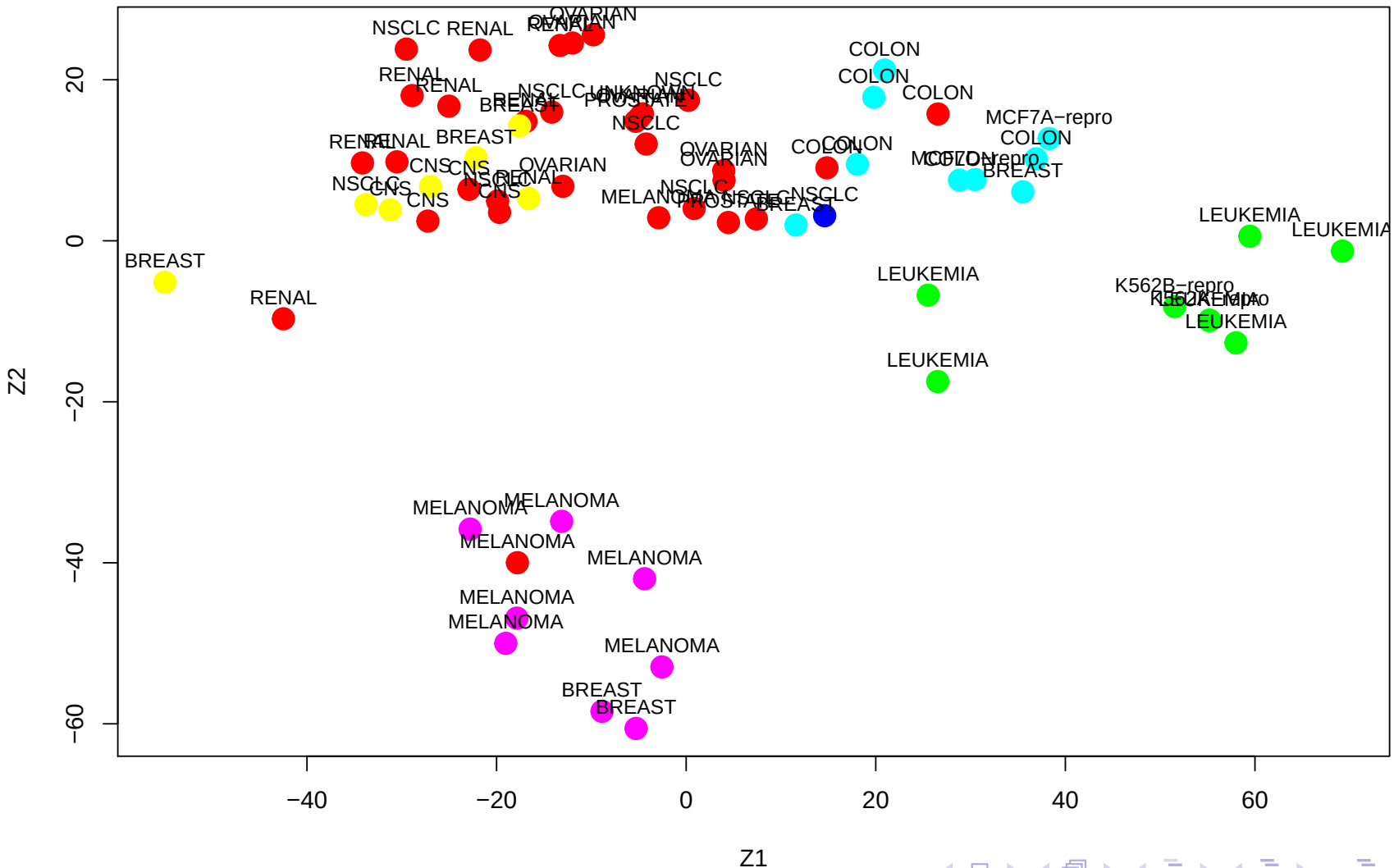
# Clustering case study: NCI60 data (continuation)

We can also perform hierarchical clustering on the NCI60 data. The dendogram looks as follows



# Clustering case study: NCI60 data (continuation)

Cut the dendrogram such that you have K = 6 clusters. Colour each cluster and plot the data in the two-dimensional representation obtained from PCA:



# Summary

- We discussed different types of **unsupervised learning**.
- In **principal component analysis (PCA)** our aim was to **obtain a lower-dimensional representation** of the unlabelled data. This reduced data can then be used for visualisation or as inputs to other supervised learning algorithms.
- In **clustering algorithms** we aim to **assign labels to data** according to some kind of similarity measure of different points. If some group of points is 'similar', then we assign a given label to them. Concretely, we looked at:
  - K-means clustering
  - Hierarchical clustering