Statistics 452: Statistical Learning and Prediction

Chapter 10, part 3: Hierarchical Clustering

Brad McNeney

Hierarchical Clustering

- Instead of setting the number of clusters in advance, as in K-means/medoids, we create a tree drawing (dendrogram) that represents a hierarchy of nested partitions of the objects into clusters.
  - See the example on the next slide.
- We can create the hierarchy in a top-down or bottom-up fashion.
- Bottom-up, or agglomerative clustering is the most common and will be described.
  - Given a measure of dissimilarity between clusters, we successively fuse, or merge clusters, starting with n clusters of size one and ending with a single cluster of size n.

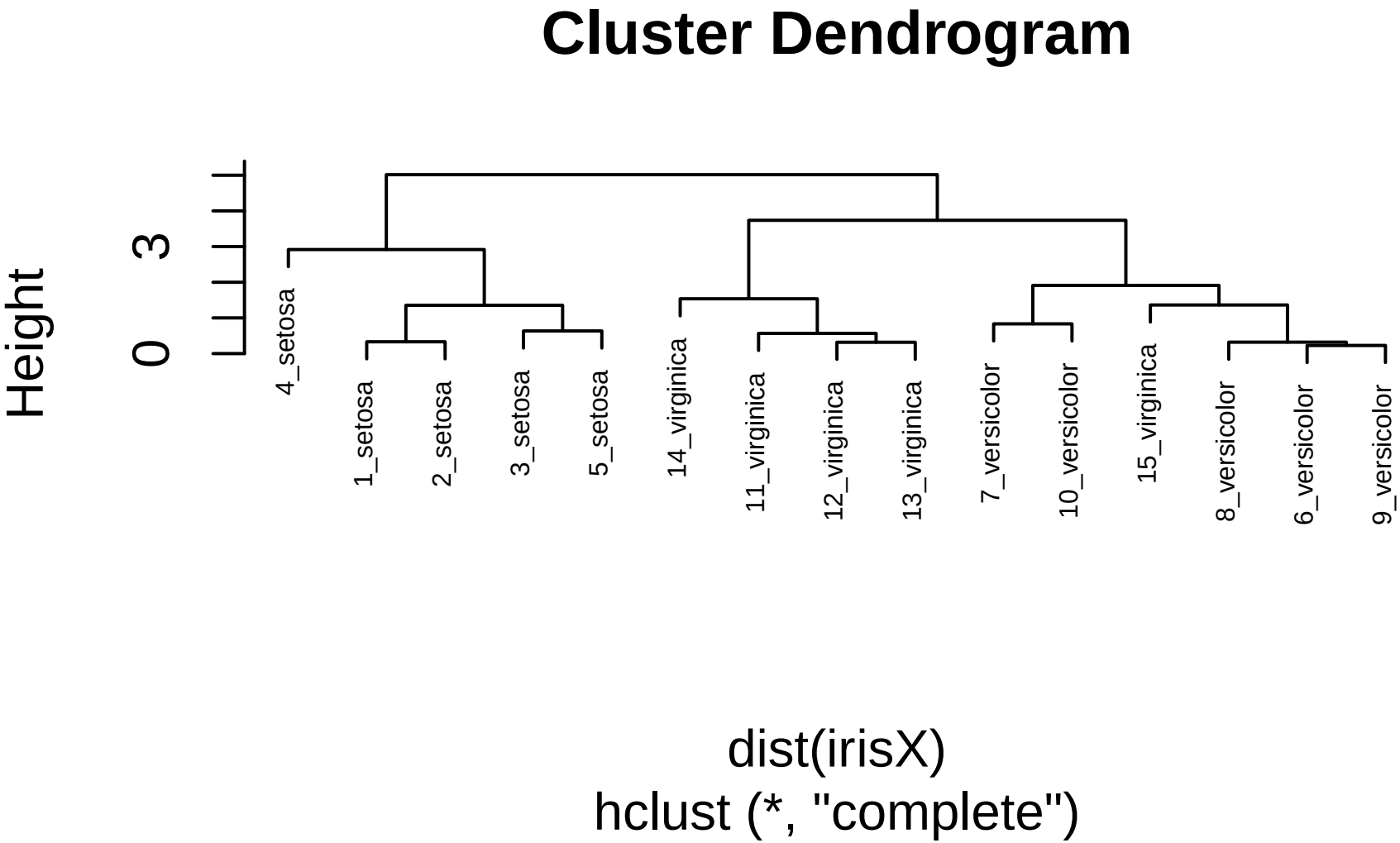
Example: Hierarchical Clustering of Irises

- First select a subsample of 5 irises from each species, then remove the species information.
- We use the function `hclust()` to generate the hierarchical clustering.

```
data(iris)
set.seed(1)
iris <- iris %>%
  group_by(Species) %>%
  sample_n(size=5) %>%
  ungroup()
irisX <- iris %>%
  select(-Species) %>%
  scale()
rownames(irisX) <- paste(rownames(iris),iris$Species,sep="_")
ic <- hclust(dist(irisX))
```

Plotting the Dendrogram

```
plot(ic,cex=.5)
```

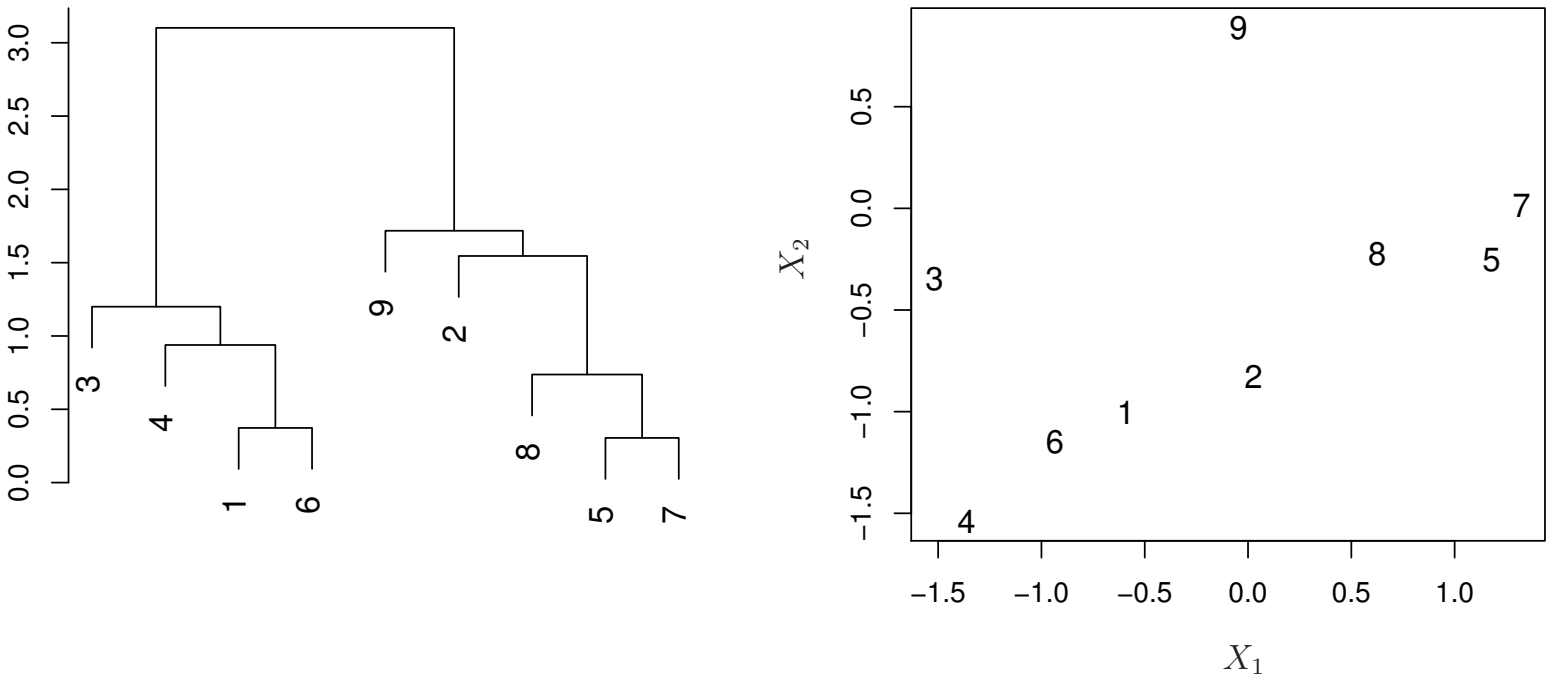


Interpretation of the Dendrogram

- The height of a node reflects the dissimilarity of its two descendant clusters.
  - Branch lengths are not generally informative.
- The first node on the dendrogram partitions the data in to two clear clusters.
  - Knowing the species we can see this reflects the separation between the *setosa* and other species.
- The second node roughly separates the *versicolor* and *virginica* species, though there is one *virginica* in the *versicolor* cluster.
  - Note: The subtrees can be rotated without changing the structure of the dendrogram, so we should not interpret the horizontal placement of the leaves and/or branches.
  - Note: This clustering is for a *subset* of the iris data. Different subsets yield slightly different clusters.

Simulated Data Example

- Figure 10.10 from the text:



- The height of the node that merges $\{9\}$ with $\{2, 8, 5, 7\}$ reflects the dissimilarity. Branch lengths separating, say, 9 and 2 are not meaningful.

Cutting Dendrograms to Obtain Clusters

- Cutting the dendrogram at a given dissimilarity value leads to clusters.
  - For example, on the iris dendrogram, cutting at about 4 gives two clusters (*setosa vs* others) and cutting at about 3 gives three clusters (*setosa* and roughly *vesicolor* and *virginica*.)
- The `cutree()` function allows us to cut at either a height `h` or where there are `k` clusters.

```
cutree(ic,k=3)
```

```
##      1_setosa      2_setosa      3_setosa      4_setosa      5_setosa
##             1             1             1             1             1
##  6_versicolor  7_versicolor  8_versicolor  9_versicolor 10_versicolor
##             2             2             2             2             2
##  11_virginica  12_virginica  13_virginica  14_virginica  15_virginica
##             3             3             3             3             2
```

```
table(cutree(ic,k=3))
```

```
##
## 1 2 3
## 5 6 4
```

Hierarchical Clustering Algorithm

1. Begin with n observations and a measure of the $n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.
2. For $i = n, n-1, \ldots, 2$:
   (a) Identify the pair of clusters that are least dissimilar and merge them. The dissimilarity between merged clusters is the height of the new node on the dendrogram.
   (b) Compute the pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

- To be determined: How do we measure dissimilarity between objects and between **clusters**?

Dissimilarity Between Objects

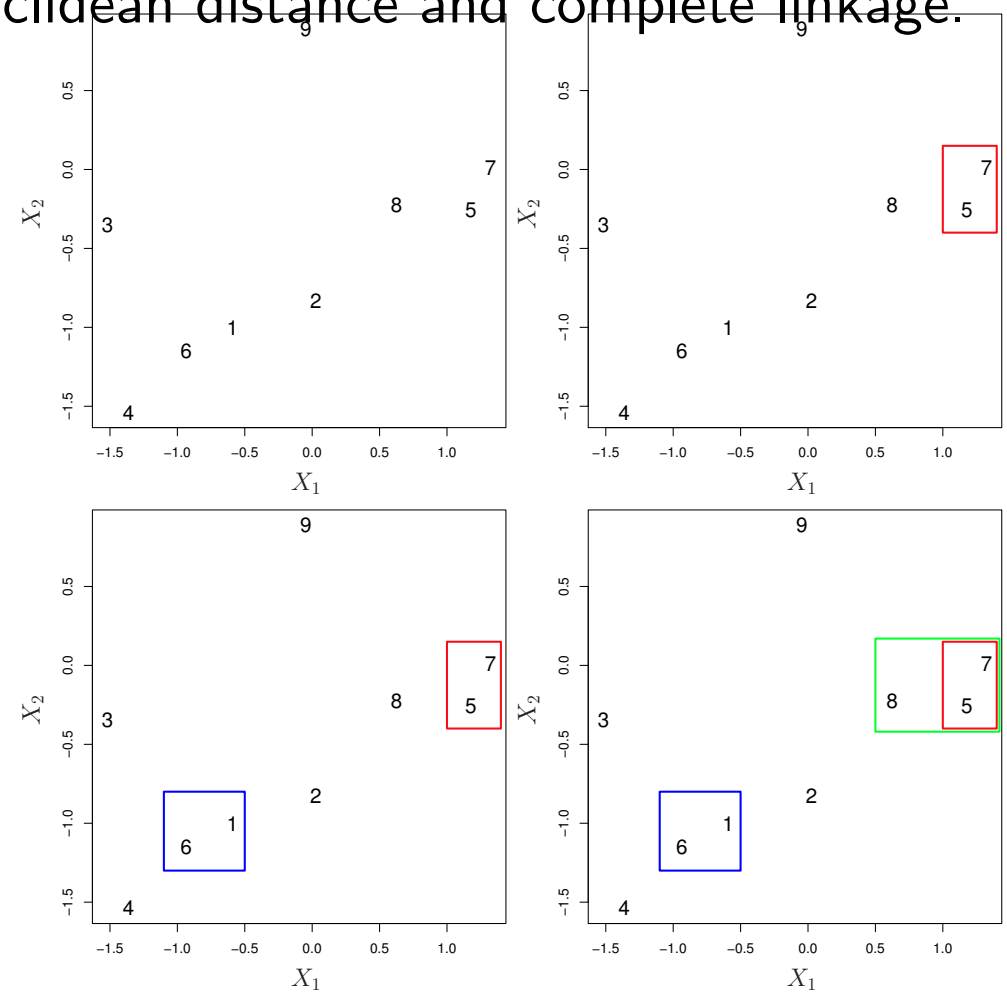
- Several possibilities:
  - Euclidean (ℓ_2) distance, $d(a,b) = \sqrt{\sum_{i=1}^{p}(a_i - b_i)^2}$.
  - Squared Euclidean (ℓ_2^2) distance, $d(a,b) = \sum_{i=1}^{p}(a_i - b_i)^2$.
  - Manhattan (ℓ_1) distance, $d(a,b) = \sum_{i=1}^{p}|a_i - b_i|$
  - Maximum (ℓ_∞) distance, $d(a,b) = \max_i |a_i - b_i|$.
- Euclidean, Manhattan and maximum distances are implemented in the `dist()` function in R, and squared Euclidean can be computed with `dist(x,method="euclidean")^2`.

Linkage: Dissimilarity Between Clusters

- The four most common linkages are:
  - Complete: $\max\{\, d(a,b) : a \in A,\ b \in B \,\}$.
  - Average: $\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$.
  - Centroid: $||c_s - c_t||$ where c_s and c_t are the centroids of clusters s and t, respectively.
  - Single: $\min\{\, d(a,b) : a \in A,\ b \in B \,\}$.
- According to the text, average, complete and single linkage are the most popular among statisticians, and average and complete are preferred because they produce more balanced dendrograms than single.
- In `hclust()`, complete is the default, and the three others are options.

Example: Clustering of the Simulated Data Example

- Figure 10.11 from the text: The first three steps of clustering using Euclidean distance and complete linkage.



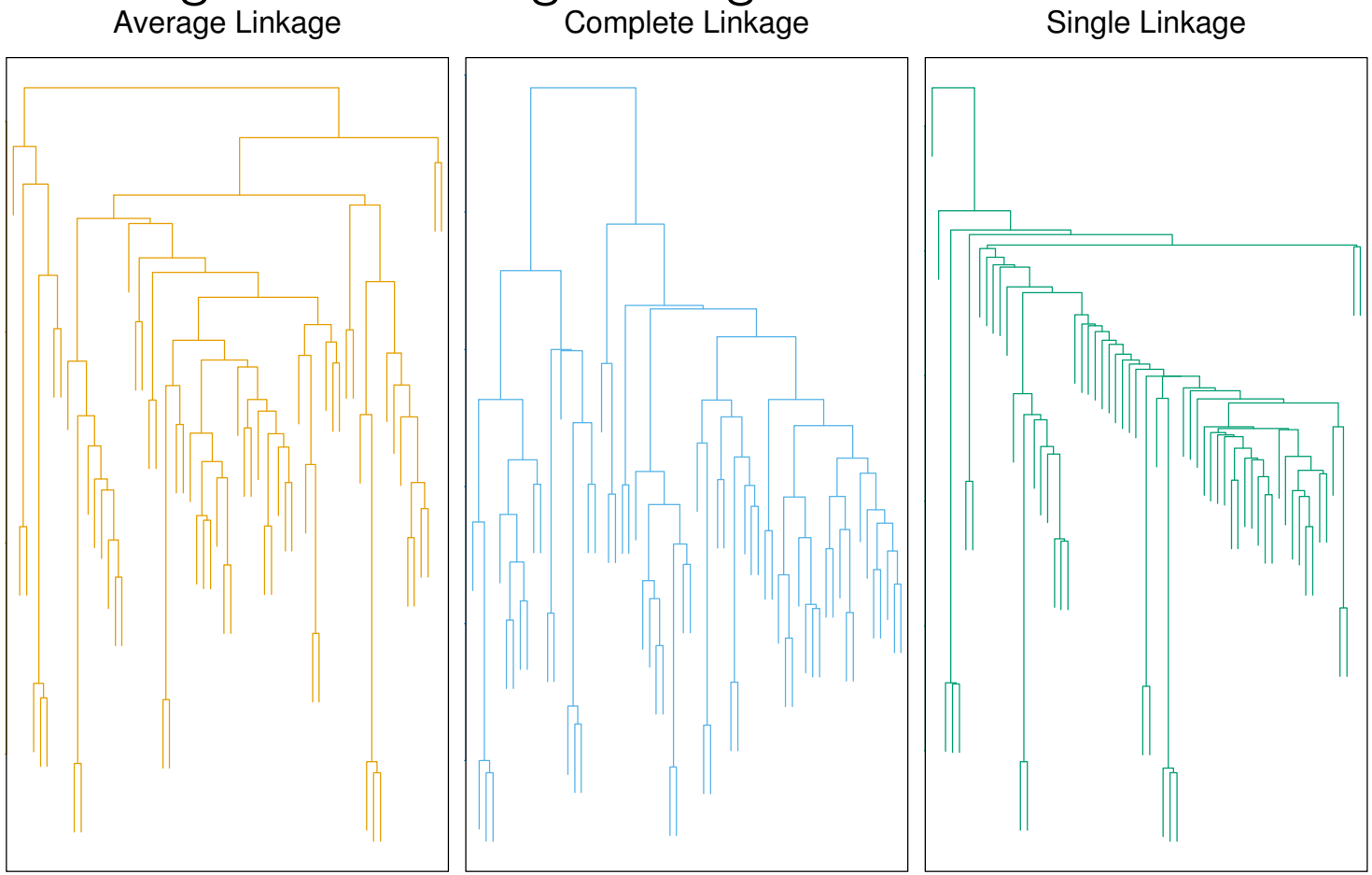
- Read top-left, top-right, bottom-left, bottom-right.

The Choice of Dissimilarity, Linkage and Scaling Affect the Dendrogram

- Each choice will influence the dendrogram.
- Illustrate sensitivity to linkage and scaling.

Sensitivity to Linkage

- Figure 10.12 from the text. Note the imbalance in the dendrogram under single linkage:



Sensitivity to Scaling

- Whether to scale or not is problem dependent.
- The amount of variation in a variable will determine how much it influences the dissimilarities, and therefore the linkages between clusters.
- Example: Decathlon data from the `FactoMineR` package.

```
library(FactoMineR) #install.packages("FactoMineR")
data(decathlon)
# Data processing strips off row names. Save for later.
rnames <- rownames(decathlon)
# Extract Olympics competition
rnames <- rnames[decathlon$Competition=="OlympicG"]
decathlon <- filter(decathlon,Competition=="OlympicG")
# Extract data on the 10 events.
decathlon <- decathlon[,1:10] # Extract data on the 10 events.
# In most events, a high score is good, but the opposite is true for running.
# Change the running to scores where high is good by subtracting the times from
# the maximum time
diffmax <- function(x) { max(x)-x }
decathlon <- mutate(decathlon,
                    `100m` = diffmax(`100m`),
                    `400m` = diffmax(`400m`),
                    `110m.hurdle` = diffmax(`110m.hurdle`),
                    `1500m` = diffmax(`1500m`))
rownames(decathlon) <- rnames
```

```r
head(decathlon)
```

```
##           100m Long.jump Shot.put High.jump 400m 110m.hurdle Discus
## Sebrle    0.51      7.84    16.36      2.12 4.84         1.34  48.72
## Clay      0.92      7.96    15.23      2.06 4.01         1.26  50.11
## Karpov    0.86      7.81    15.93      2.09 6.39         1.42  51.65
## Macey     0.47      7.47    15.73      2.15 4.23         0.83  48.34
## Warners   0.74      7.74    14.48      1.97 5.23         1.38  43.73
## Zsivoczky 0.45      7.14    15.31      2.12 3.80         0.44  45.62
##           Pole.vault Javeline 1500m
## Sebrle           5.0    70.52 36.99
## Clay             4.9    69.71 35.00
## Karpov           4.6    55.54 38.89
## Macey            4.4    58.46 51.58
## Warners          4.9    55.39 38.95
## Zsivoczky        4.7    63.45 47.46
```

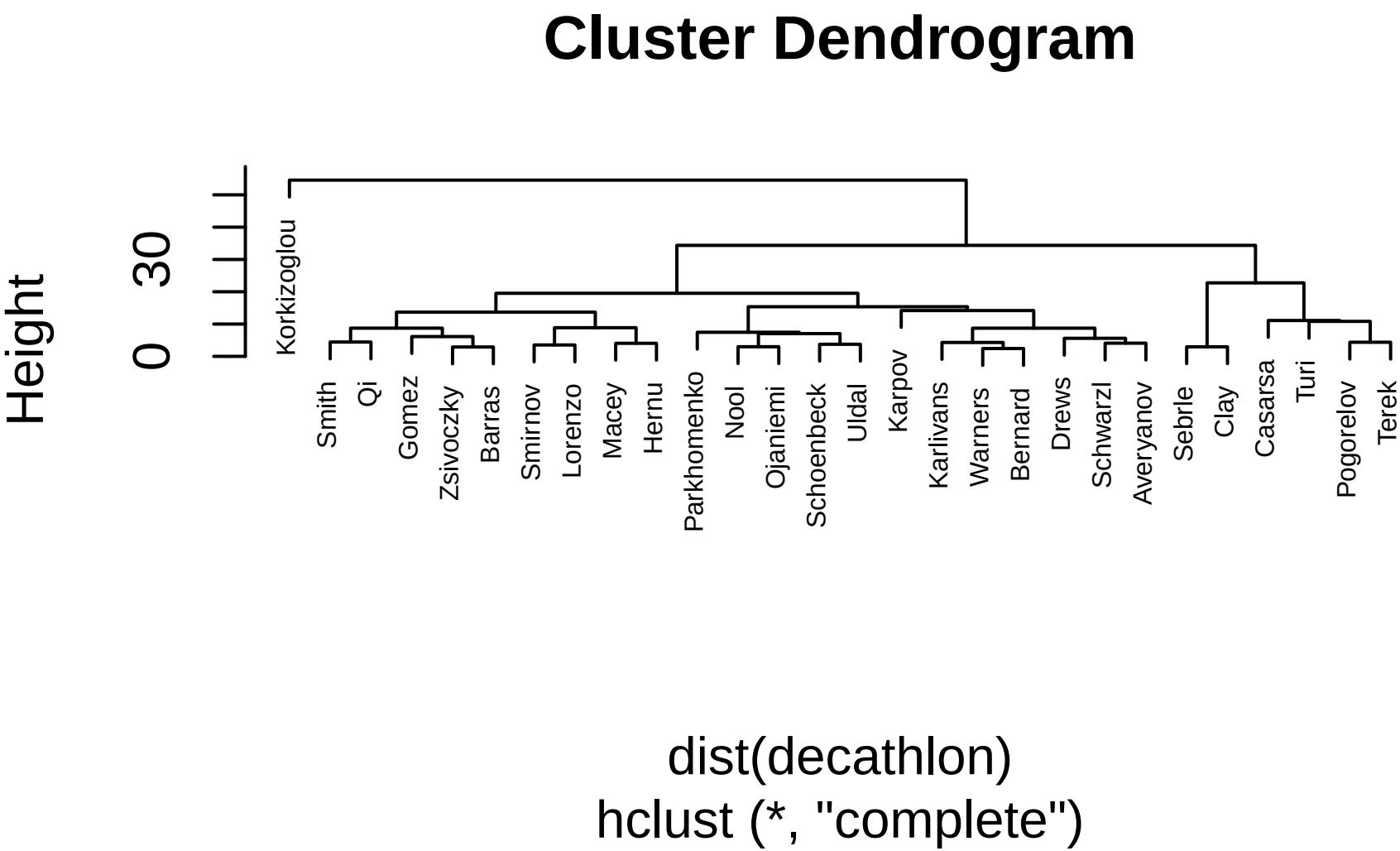
```r
round(diag(var(decathlon)),3)
```

```
##        100m   Long.jump    Shot.put   High.jump        400m 110m.hurdle
##       0.053       0.116       0.733       0.008       1.609       0.196
##      Discus  Pole.vault    Javeline       1500m
##      10.887       0.084      24.759     128.184
```

Clustering of the Decathlon Data Without Scaling

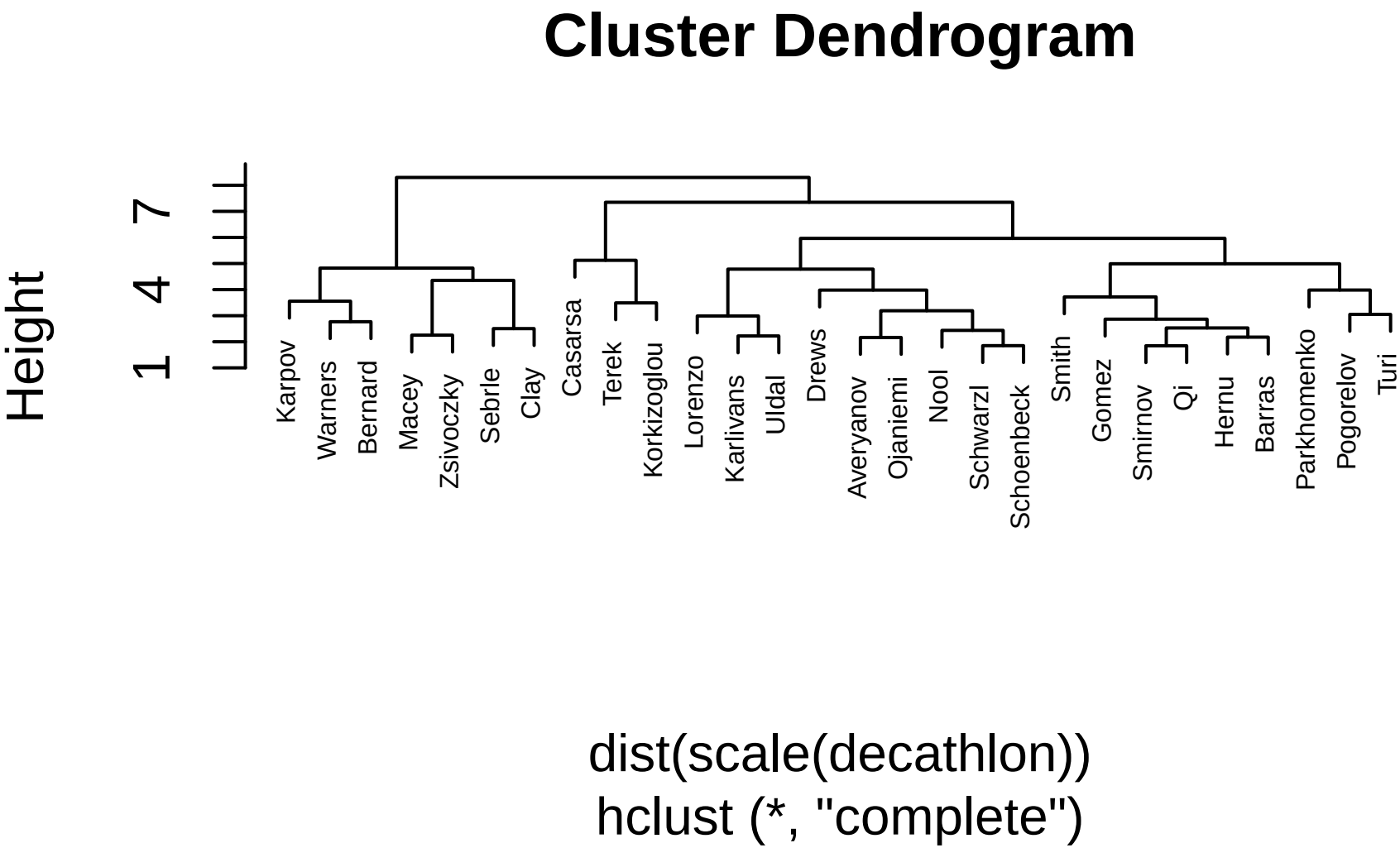
- Korkizoglou stands apart, because he beat the rest of the field by more than 20 seconds in the 1500m.

```
plot(hclust(dist(decathlon)),cex=.5)
```



Clustering of the Decathlon Data With Scaling

```
plot(hclust(dist(scale(decathlon))),cex=.5)
```



NCI60 data

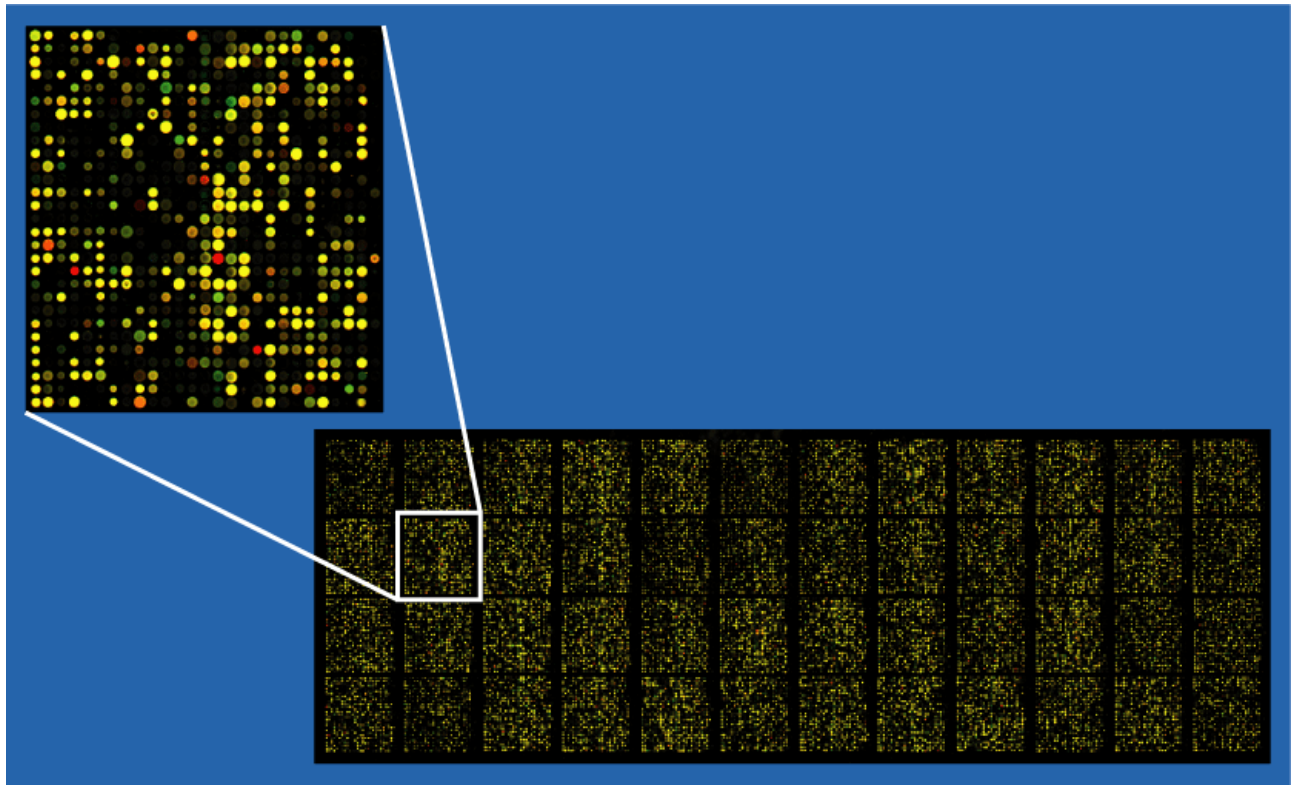
- We follow the lab on clustering of the NCI60 data set, which contains the results of a DNA microarray study of 64 cancer cells.
- Cancer cells are labelled by location of the cancer.
  - However, recent research suggests that classification based on location of the cancer may not be as useful as classification based on the cancer-causing mutation (e.g., a mutation in a gene responsible for DNA repair).

DNA Microarray Experiments

- See the Wikipedia page on microarrays (https://en.wikipedia.org/wiki/DNA_microarray) for a description.
- Briefly:
  - Genes in a cell are transcribed to produce messenger RNA, which is extracted and copied into DNA.
  - The DNA is fragmented, flourescently labelled, and then exposed to an ordered array of complementary DNA molecules called probes that identify specific genes.
  - The array "lights up" where the labelled DNA has bound to the probes. The flourescence intensity at each probe is a measure of how much of the corresponding gene was being expressed in the cell.

Microarray Picture

- Example micorarray with about 40,000 probes



Source: Wikimedia Commons

The NCI Data

- Rows are cancer cells, labelled by the type of cancer, and columns are the probes (genes).
- Entries of the data matrix are the flourescence intensities after quality control has been applied.

```
library(ISLR)
data(NCI60)
nciX <- NCI60$data
dim(nciX)
```

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

```
cancer_types <- NCI60$labs
unique(cancer_types) # MCF7's are Breast and K562's are Leukemia
```

```
##  [1] "CNS"          "RENAL"        "BREAST"       "NSCLC"        "UNKNOWN"
##  [6] "OVARIAN"      "MELANOMA"     "PROSTATE"     "LEUKEMIA"     "K562B-repro"
## [11] "K562A-repro" "COLON"        "MCF7A-repro" "MCF7D-repro"
```

Collapse cancer types

```
ll <- (cancer_types=="MCF7A-repro") | (cancer_types=="MCF7D-repro")
cancer_types[ll] <- "LEUKEMIA"
bb <- (cancer_types=="K562A-repro") | (cancer_types=="K562B-repro")
cancer_types[bb] <- "BREAST"
rownames(nciX) <- cancer_types
nciX[1:5,1:5]
```

```
##                1         2         3         4         5
## CNS     0.300000  1.180000  0.550000  1.140000 -0.265000
## CNS     0.679961  1.289961  0.169961  0.379961  0.464961
## CNS     0.940000 -0.040000 -0.170000 -0.040000 -0.605000
## RENAL   0.280000 -0.310000  0.680000 -0.810000  0.625000
## BREAST  0.485000 -0.465000  0.395000  0.905000  0.200000
```

PCA on the NCI60 Data

- One could argue that highly expressed genes **should** drive the PCs, but we scale.

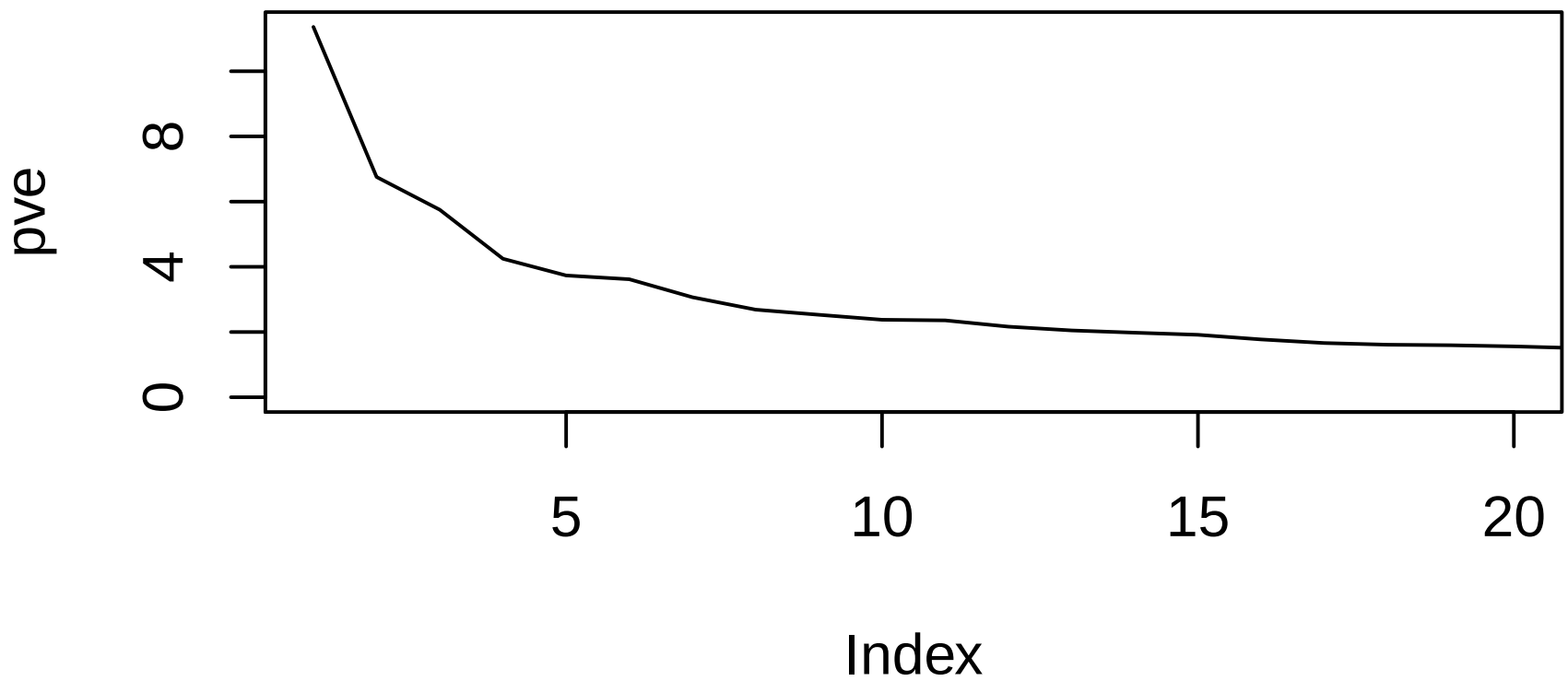
```
nciX <- scale(nciX)
pcout <- prcomp(nciX)
summary(pcout)
```

```
## Importance of components:
##                             PC1      PC2      PC3      PC4      PC5
## Standard deviation     27.8535 21.48136 19.82046 17.03256 15.97181
## Proportion of Variance  0.1136  0.06756  0.05752  0.04248  0.03735
## Cumulative Proportion   0.1136  0.18115  0.23867  0.28115  0.31850
##                             PC6      PC7      PC8      PC9     PC10
## Standard deviation     15.72108 14.47145 13.54427 13.14400 12.73860
## Proportion of Variance  0.03619  0.03066  0.02686  0.02529  0.02376
## Cumulative Proportion   0.35468  0.38534  0.41220  0.43750  0.46126
##                            PC11     PC12     PC13     PC14     PC15
## Standard deviation     12.68672 12.15769 11.83019 11.62554 11.43779
## Proportion of Variance  0.02357  0.02164  0.02049  0.01979  0.01915
## Cumulative Proportion   0.48482  0.50646  0.52695  0.54674  0.56590
##                            PC16     PC17     PC18     PC19     PC20
## Standard deviation     11.00051 10.65666 10.48880 10.43518 10.3219
## Proportion of Variance  0.01772  0.01663  0.01611  0.01594  0.0156
## Cumulative Proportion   0.58361  0.60024  0.61635  0.63229  0.6479
##                            PC21    PC22    PC23    PC24    PC25    PC26
## Standard deviation     10.14608 10.0544 9.90265 9.64766 9.50764 9.33253
```

Scree Plot

- Express variances as percent total

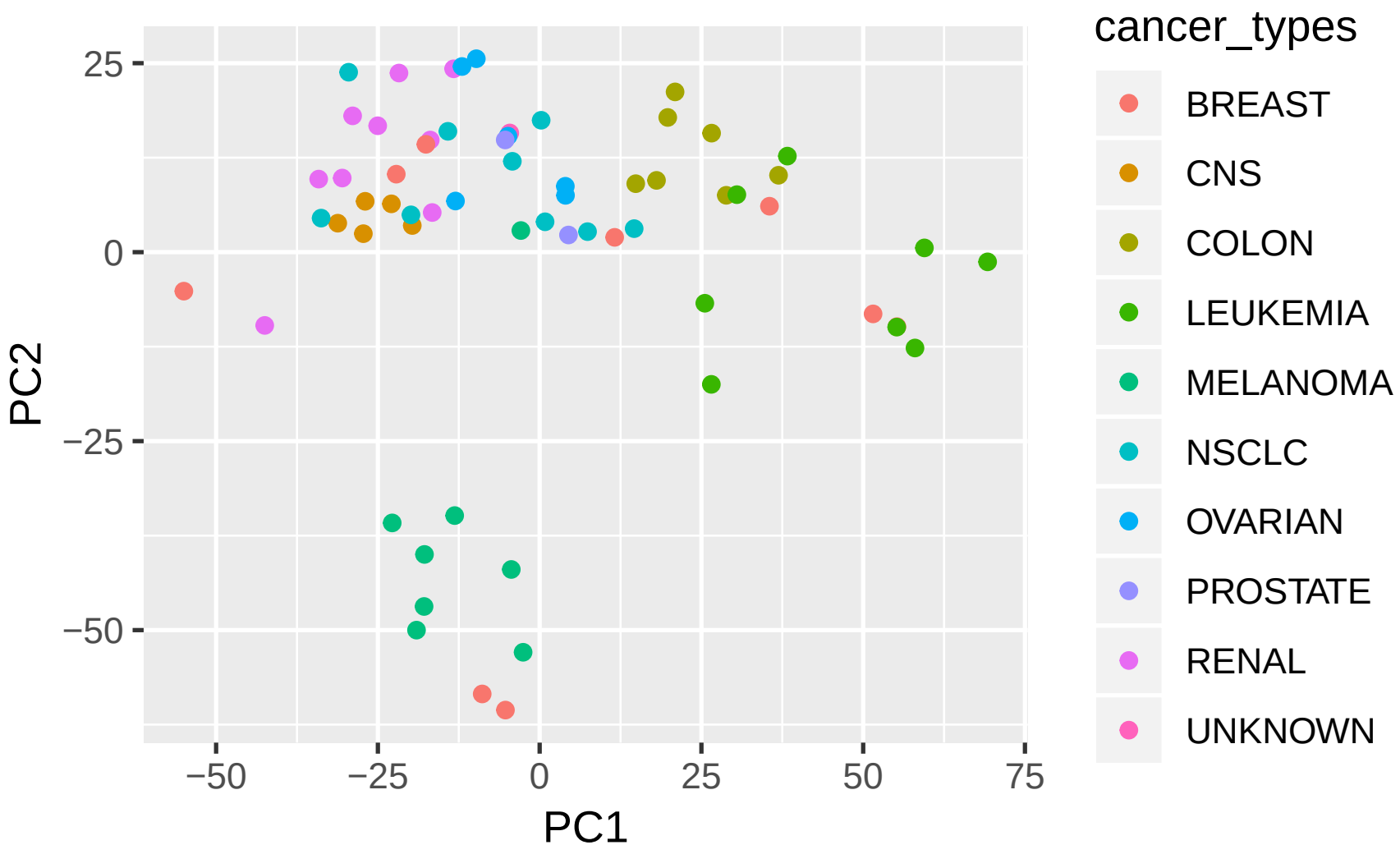
```
pve <- 100*pcout$sdev^2/sum(pcout$sdev^2)
plot(pve,xlim=c(1,20),type="l")
```



- Possible "elbow" at about 5 PCs

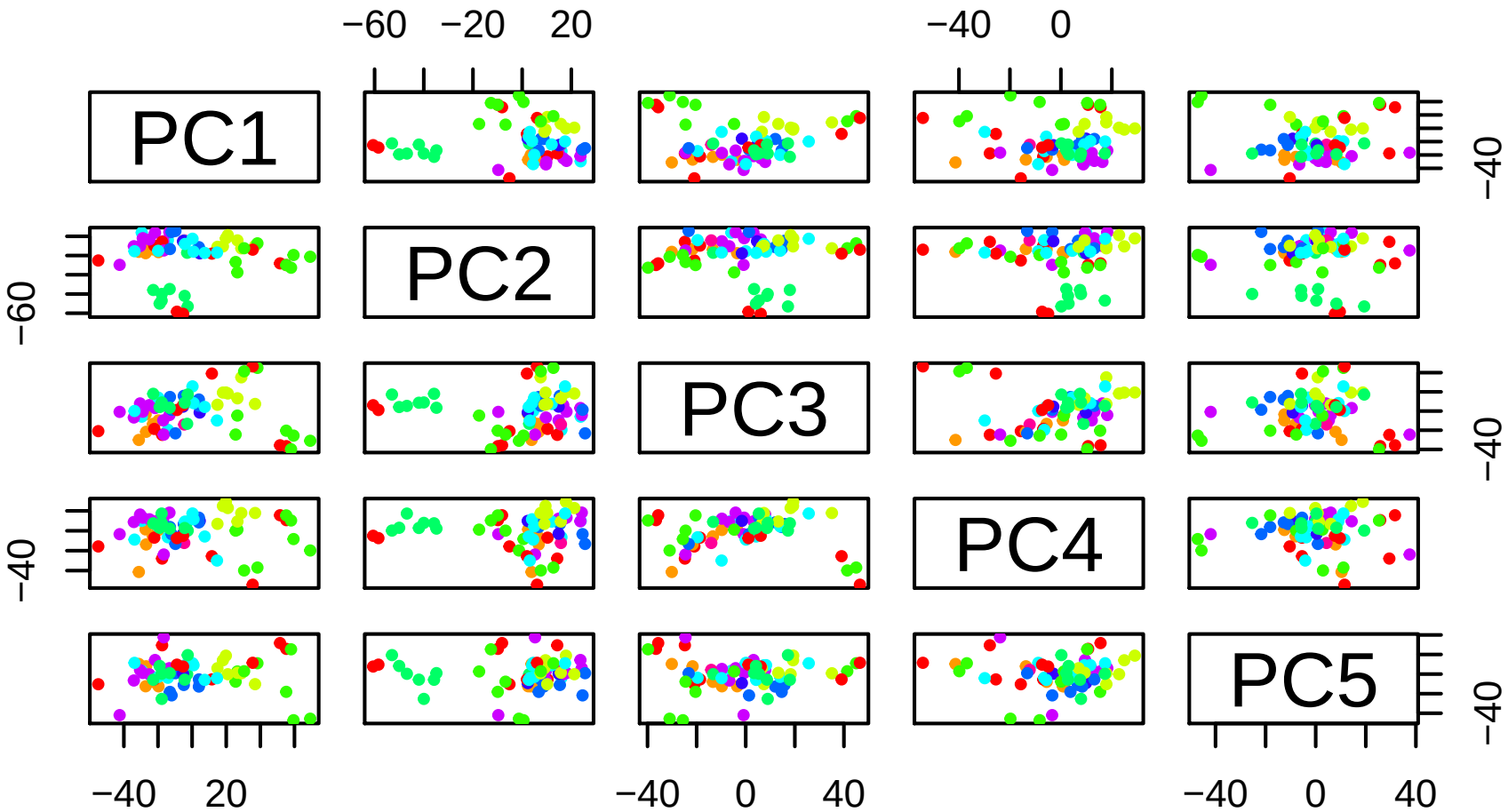
PC Plots

```
pcs <- as_tibble(pcout$x) %>% mutate(cancer_types = factor(cancer_types))
ggplot(pcs,aes(x=PC1,y=PC2,color=cancer_types)) + geom_point()
```



Pairwise PC plots

```
rcols <- rainbow(length(unique(cancer_types)))
ccols <- rcols[as.numeric(factor(cancer_types))]
pairs(pcout$x[,1:5],col=ccols,pch=19,cex=.5)
```



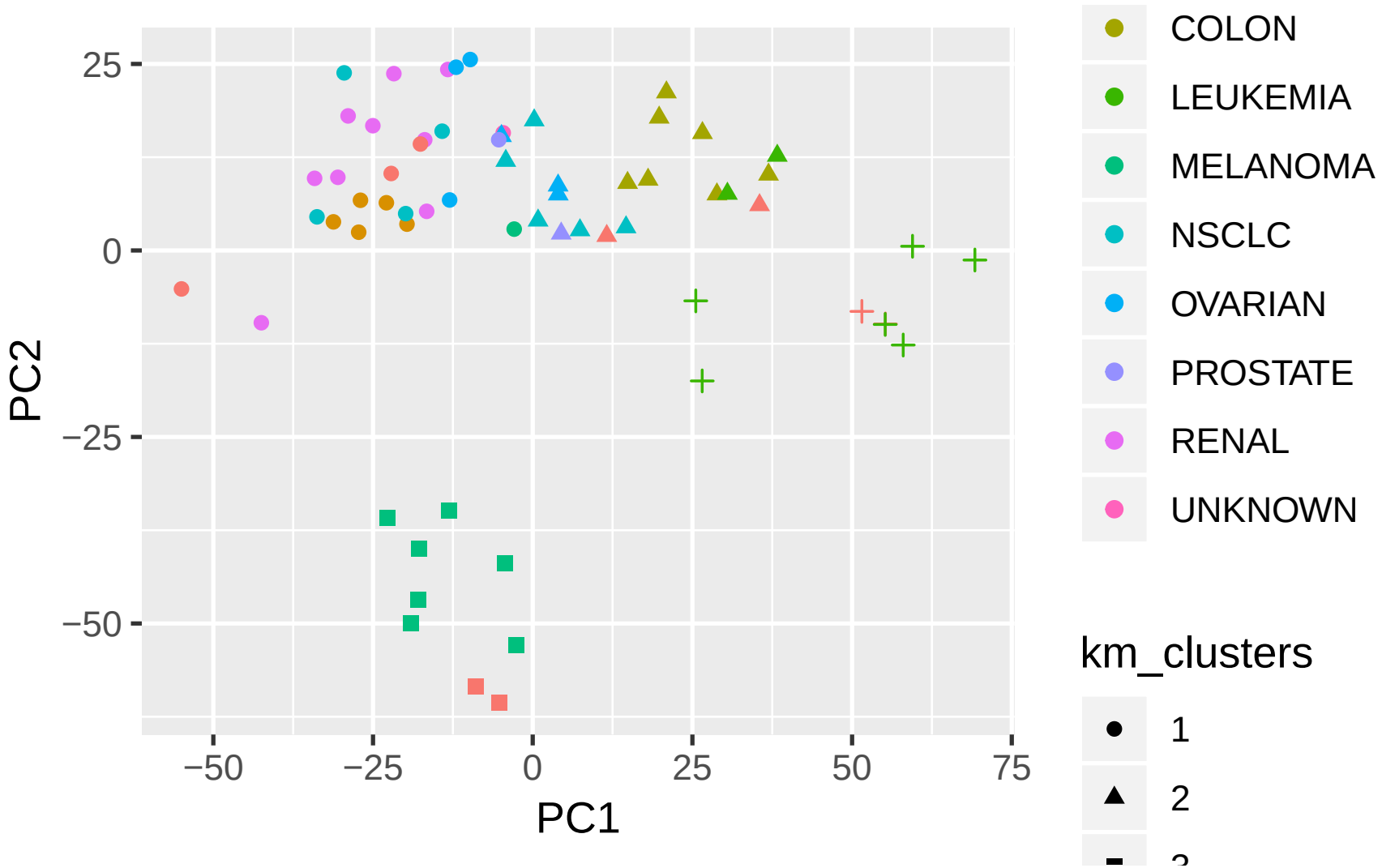
K-Means Clustering of NCI60 Data

- Remember that `nciX` has already been scaled.
- We know there are 16 different cancer types, but would not specify this many clusters in practice.
  - Try $K = 4$.

```
kout <- kmeans(nciX,centers=4)
table(kout$cluster,cancer_types)
```

```
##    cancer_types
##     BREAST CNS COLON LEUKEMIA MELANOMA NSCLC OVARIAN PROSTATE RENAL
##   1      3   5     0        0        1     4       3         1     9
##   2      2   0     7        2        0     5       3         1     0
##   3      2   0     0        0        7     0       0         0     0
##   4      2   0     0        6        0     0       0         0     0
##    cancer_types
##     UNKNOWN
##   1       1
##   2       0
##   3       0
##   4       0
```

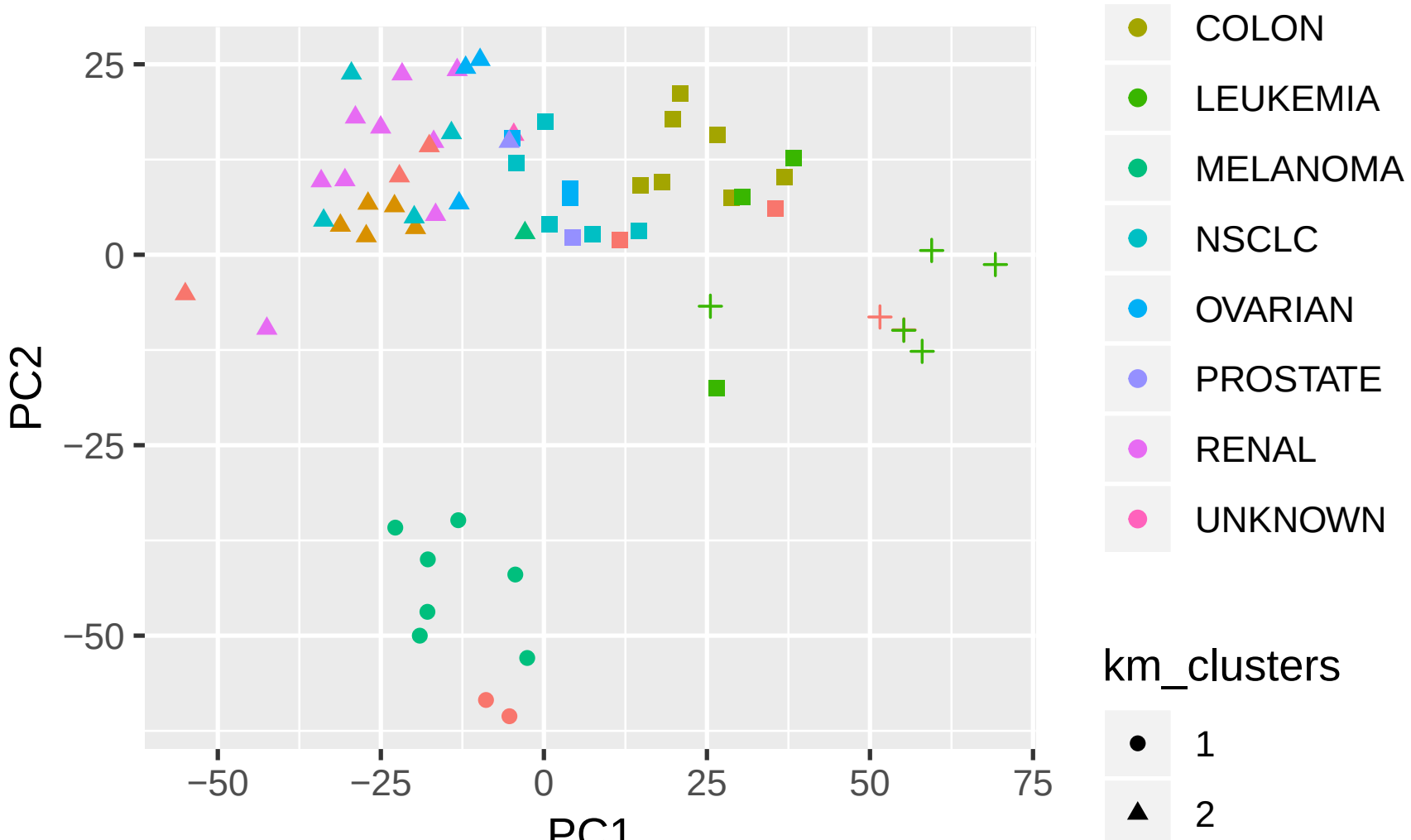
```
pcs <- mutate(pcs,km_clusters = factor(kout$cluster))
ggplot(pcs,aes(x=PC1,y=PC2,color=cancer_types,shape=km_clusters)) +
  geom_point()
```



Clustering on PCs

- Can also use the PCs as the data.

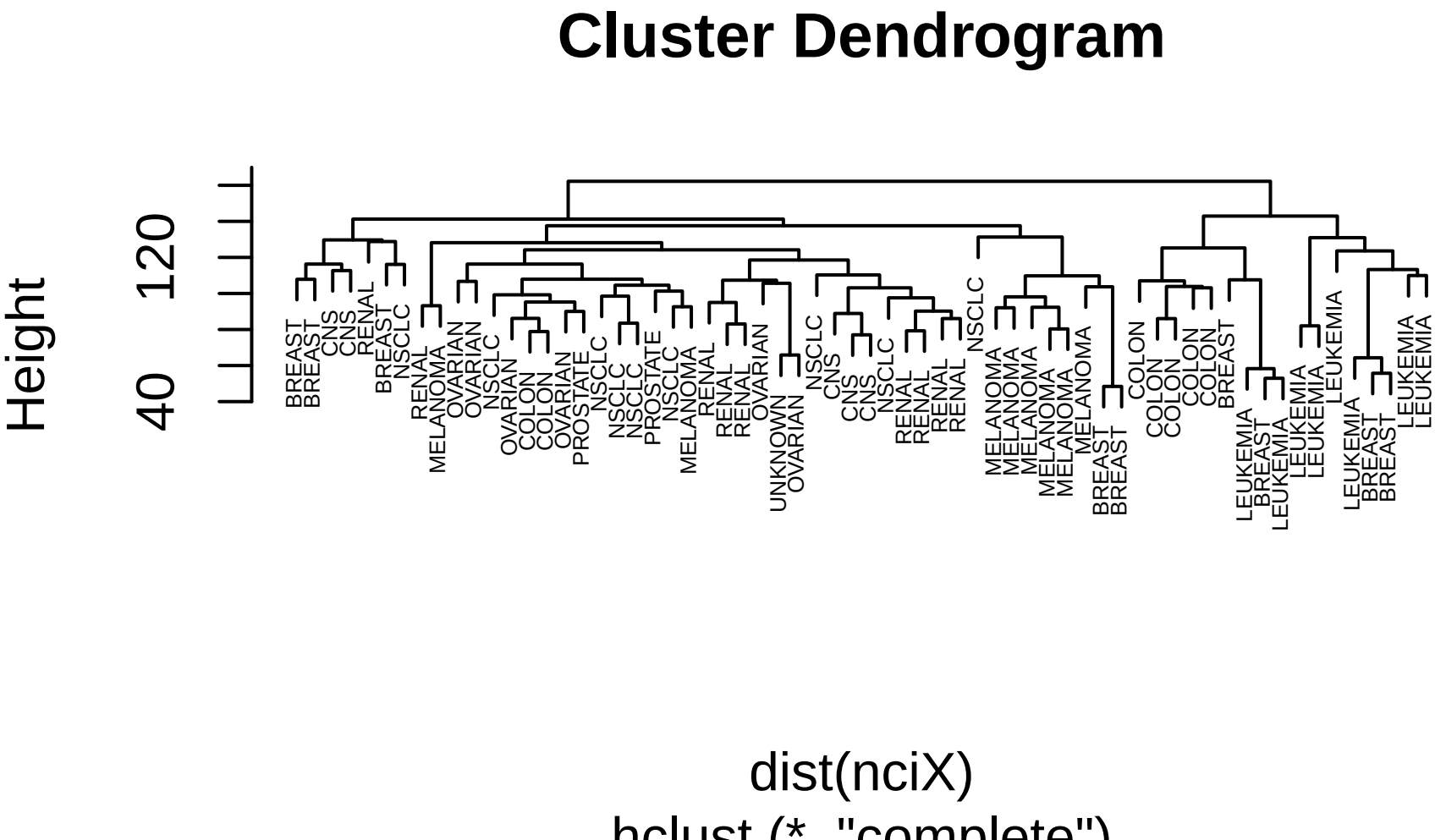
```
kout2 <- kmeans(pcout$x[,1:5],centers=4)
pcs <- mutate(pcs,km_clusters = factor(kout2$cluster))
ggplot(pcs,aes(x=PC1,y=PC2,color=cancer_types,shape=km_clusters)) +
  geom_point()
```



Hierarchical Clustering of NCI60 Data

- Use Euclidean distance and complete linkage
  - See the text for a comparison of complete, average and single linkage

```
hcout <- hclust(dist(nciX))
plot(hcout,cex=.4)
```



```
pcs <- mutate(pcs,h_clusters = factor(cutree(hcout,k=4)))
ggplot(pcs,aes(x=PC1,y=PC2,color=cancer_types,shape=h_clusters)) +
  geom_point()
```