Introduction to Clustering

Lieven Clement

Contents

1 Introduction

1.1 Objective

Objective: grouping of observations into **clusters**, so that

- similar observations appear in the same cluster
- dissimilar observations appear in distinct clusters

$\longrightarrow$ need for a measure for **similarity** and **dissimilarity**?

1.2 Example 1

Single cell transcriptomics: $n \times p$ Matrix for which

- every column contains the expression levels of one of p genes for n cells
- every row contains the expression levels of p genes for one cell (**sample**)
- Research question: look for groups of cells that have similar gene expression patterns
- Or, look for groups of genes that have similar expression levels across the different cells. This can help us in understanding the regulation and functionality of the genes.

$\longrightarrow$ both **observations** (rows) and **variables** (columns) can be clustered

1.3 Example 2

Abundance studies: the abundances of n plant species are counted on p plots (habitats)

- look for groups that contain species that live in the same habitats, or, look for groups of habitats that have similar species communities

$\longrightarrow$ both **observations** (rows) and **variables** (columns) can be clustered

2 Partition Based Cluster Analysis

- Partition based cluster methods require the number of clusters (k) to be specified prior to the start of the algorithm.

2.1 K-means Methods

- To use the k-means clustering algorithm we have to pre-define k, the number of clusters we want to define.
- The k-means algorithm is iterative.
- The algorithm starts by defining k cluster centers (centroids).
- Then the algorithm proceeds as follows

    1. First each observation is assigned to the cluster with the closest center to that observation.
    2. Then the k centers are redefined using the observations in each cluster, i.e. the multivariate means (column means) of all observations in a cluster are used to define each new cluster center.
    3. We repeat these two steps until the centers converge.

2.2 Example

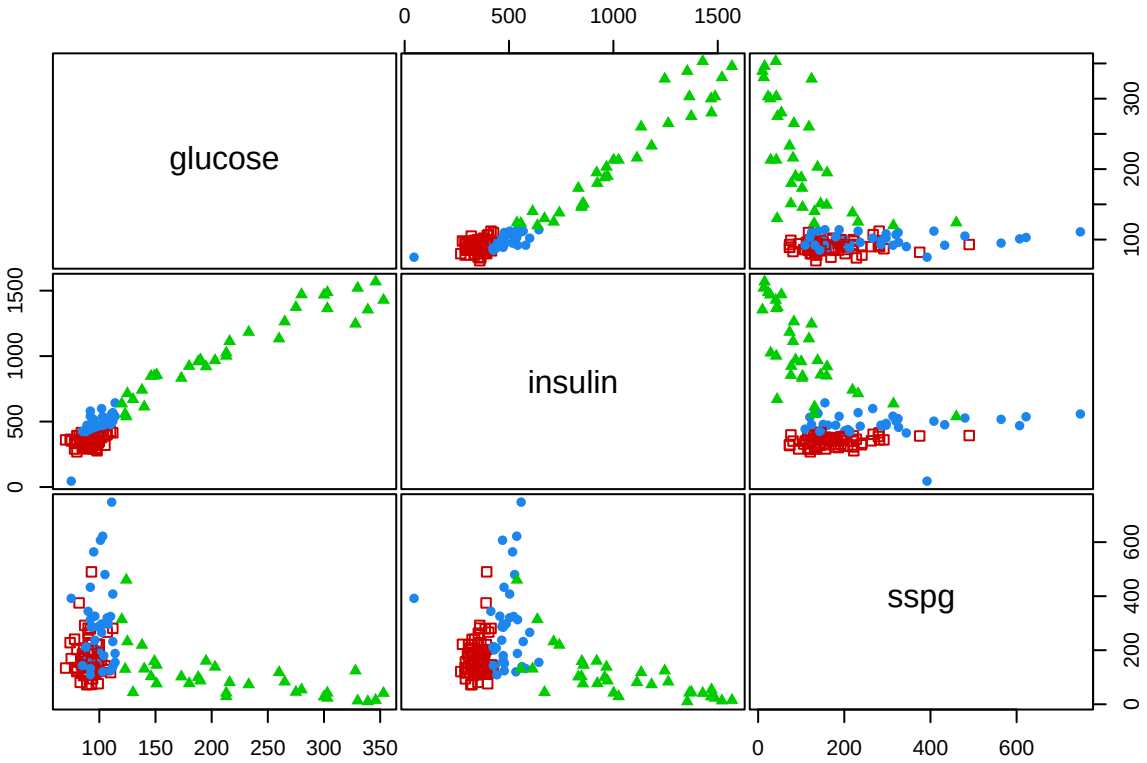
```
library(tidyverse)
data(diabetes,package = "mclust")
class <- diabetes$class
table(class)
```

```
#> class
#> Chemical   Normal    Overt
#>       36       76       33
```

```
head(diabetes)
```

```
#>    class glucose insulin sspg
#> 1 Normal      80     356  124
#> 2 Normal      97     289  117
#> 3 Normal     105     319  143
#> 4 Normal      90     356  199
#> 5 Normal      90     323  240
#> 6 Normal      86     381  157
```

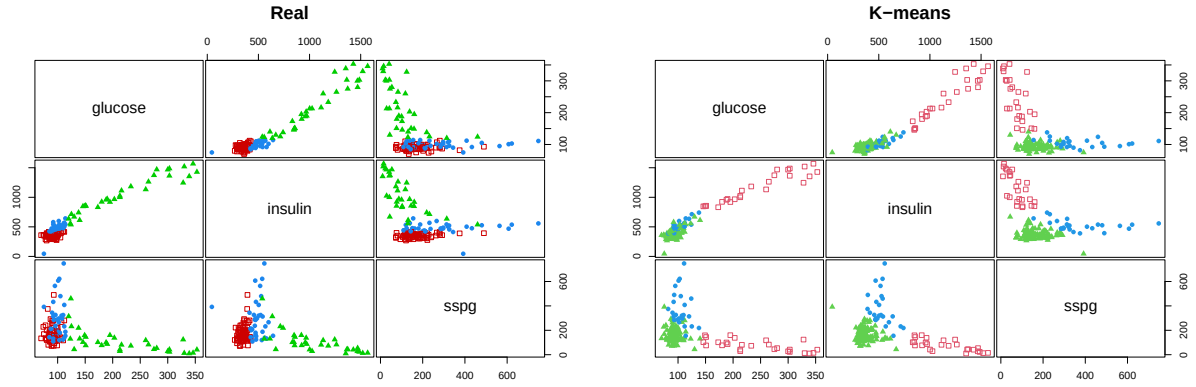
```
mclust::clPairs(diabetes[,-1], diabetes$class)
```



```r
diabetesKmeans <- kmeans(diabetes[,-1], centers = 3)
diabetesKmeans
```

```
#> K-means clustering with 3 clusters of sizes 26, 94, 25
#>
#> Cluster means:
#>     glucose   insulin      sspg
#> 1 241.65385 1152.8846  75.69231
#> 2  93.39362  375.5213 166.17021
#> 3 105.04000  525.6000 375.96000
#>
#> Clustering vector:
#>   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
#>   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
#>  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
#>   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
#>  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
#>   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
#>  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
#>   2   2   2   2   2   2   2   2   3   2   3   2   2   2   2   2   2   2   2   2
#>  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
#>   2   3   2   2   2   3   3   2   3   3   3   3   3   3   3   2   3   3   3   3
#> 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
#>   3   3   2   2   2   3   2   3   2   2   3   2   1   1   3   1   1   1   1   1
#> 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
#>   1   1   1   2   1   1   1   1   1   1   3   1   1   2   2   3   3   1   1   1
#> 141 142 143 144 145
#>   1   1   1   1   1
#>
#> Within cluster sum of squares by cluster:
#> [1] 1738796.1  947827.2  687281.9
#>  (between_SS / total_SS =  80.6 %)
#>
#> Available components:
#>
#> [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
#> [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
mclust::clPairs(diabetes[,-1], diabetes$class, main = "Real"); mclust::clPairs(diabetes[,-1], diabetesKm
```



3 Hierarchical Cluster Analysis

- Distinction between agglomerative and divisive methods
- Agglomerative start from the situation where each individual observations forms its own cluster (so it starts with n clusters). In the next steps clusters are sequentially merged, until finally there is only one cluster with n observations.
- Divisive methods work just the other way around.
- The solution of an hierarchical clustering is thus a sequence of n nested cluster solutions.

3.1 General Algorithm of Agglomerative Hierarchical Clustering

- In step 0 each observations is considered as a cluster (i.e. n clusters).
- Every next step consists of:

    1. merge the two clusters with the smallest intercluster dissimilarity
    2. recalculate the intercluster dissimilarities

In step 0 the intercluster dissimilarity coincides with the dissimilarity between the corresponding observations

$\rightarrow$ intercluster dissimilarity?

3.2 Intercluster Dissimilarities

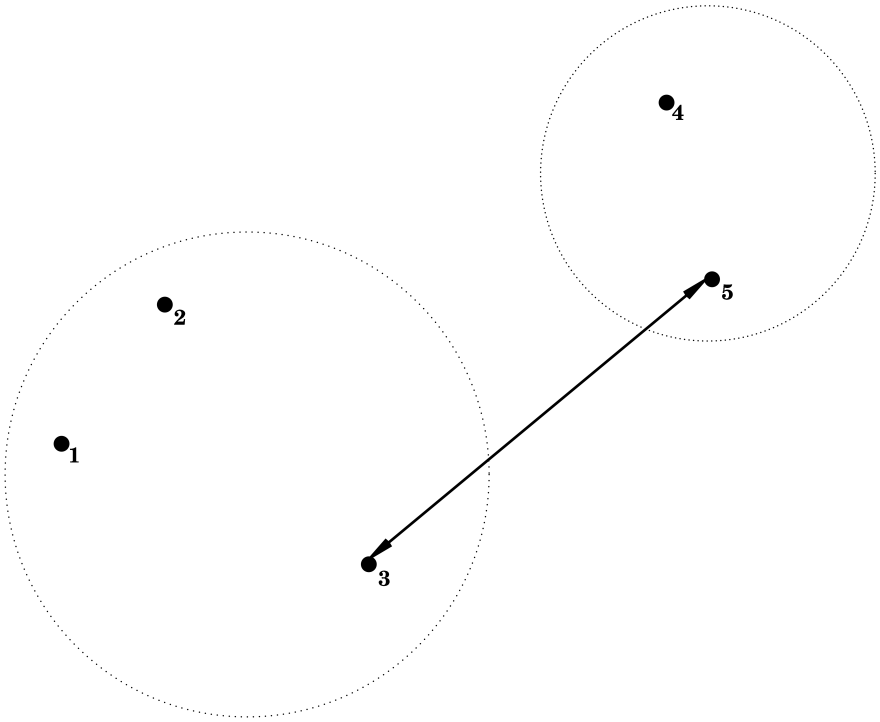
- Represent clusters (e.g. C_1 and C_2) as sets of points $\mathbf{x}_i$ which belong to that cluster
- $d(C_1, C_2)$: intercluster dissimilarity between

We consider three intercluster dissimilarities.

3.2.1 Single Linkage = Nearest Neighbour

$$d(C_1, C_2) = \min_{\mathbf{x}_1 \in C_1; \mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2),$$

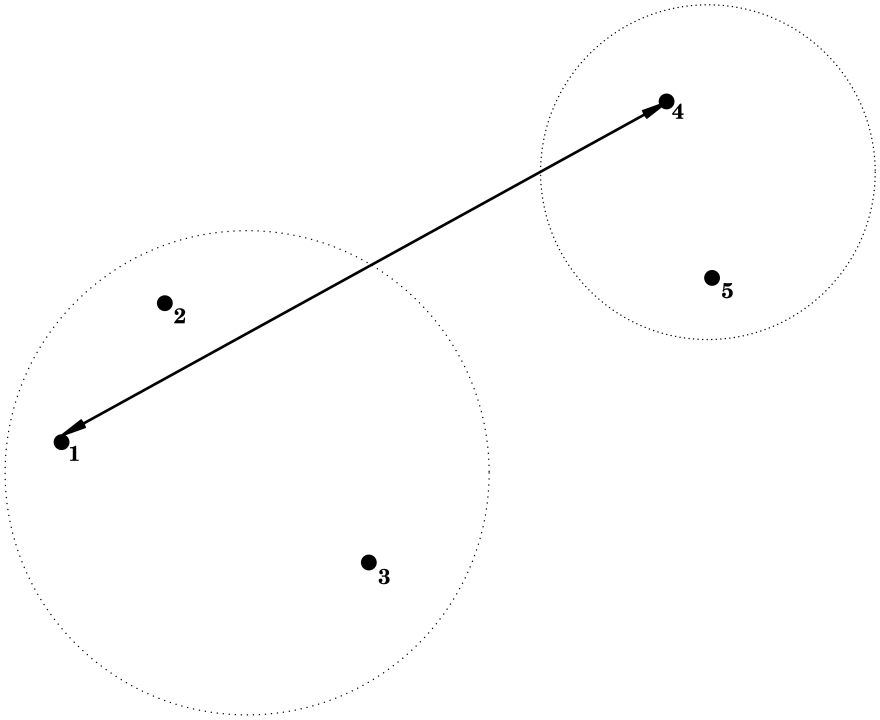
i.e. the dissimilarity between C_1 and C_2 is determined by the smallest dissimilarity between a point of C_1 and a point of C_2.



3.2.2 Complete Linkage = Furthest Neighbour

$$d(C_1, C_2) = \max_{\mathbf{x}_1 \in C_1; \mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2),$$

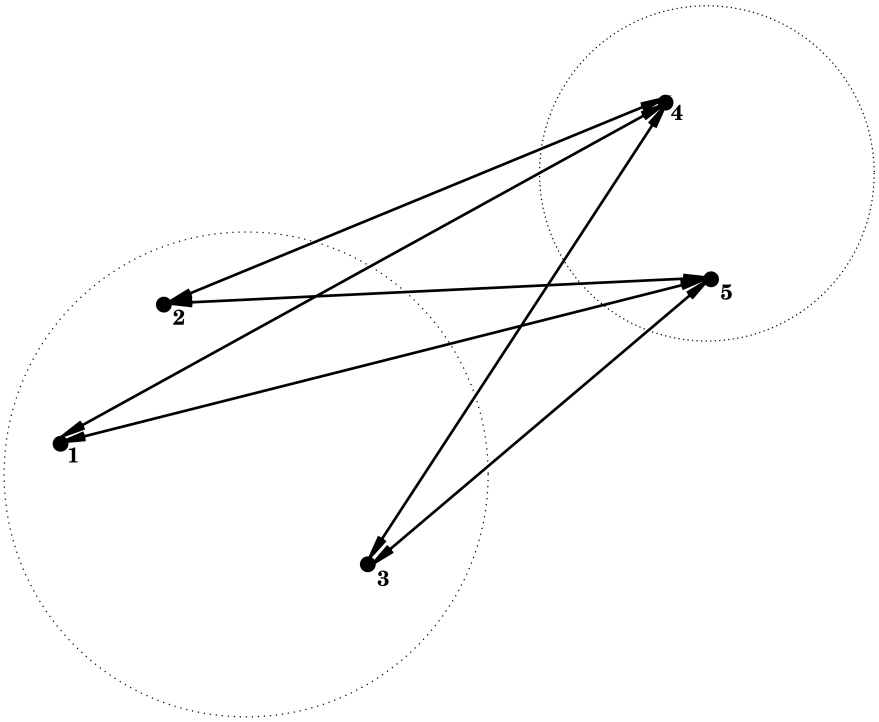
i.e. the dissimilarity between C_1 and C_2 is determined by the largest dissimilarity between a point of C_1 and a point of C_2.



3.2.3 Average Linkage = Group Average

$$d(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{\mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2} d(\mathbf{x}_1, \mathbf{x}_2),$$

i.e. the dissimilarity between C_1 and C_2 is determined by the average dissimilarity between all points of C_1 and all points of C_2.



3.3 Cluster Tree

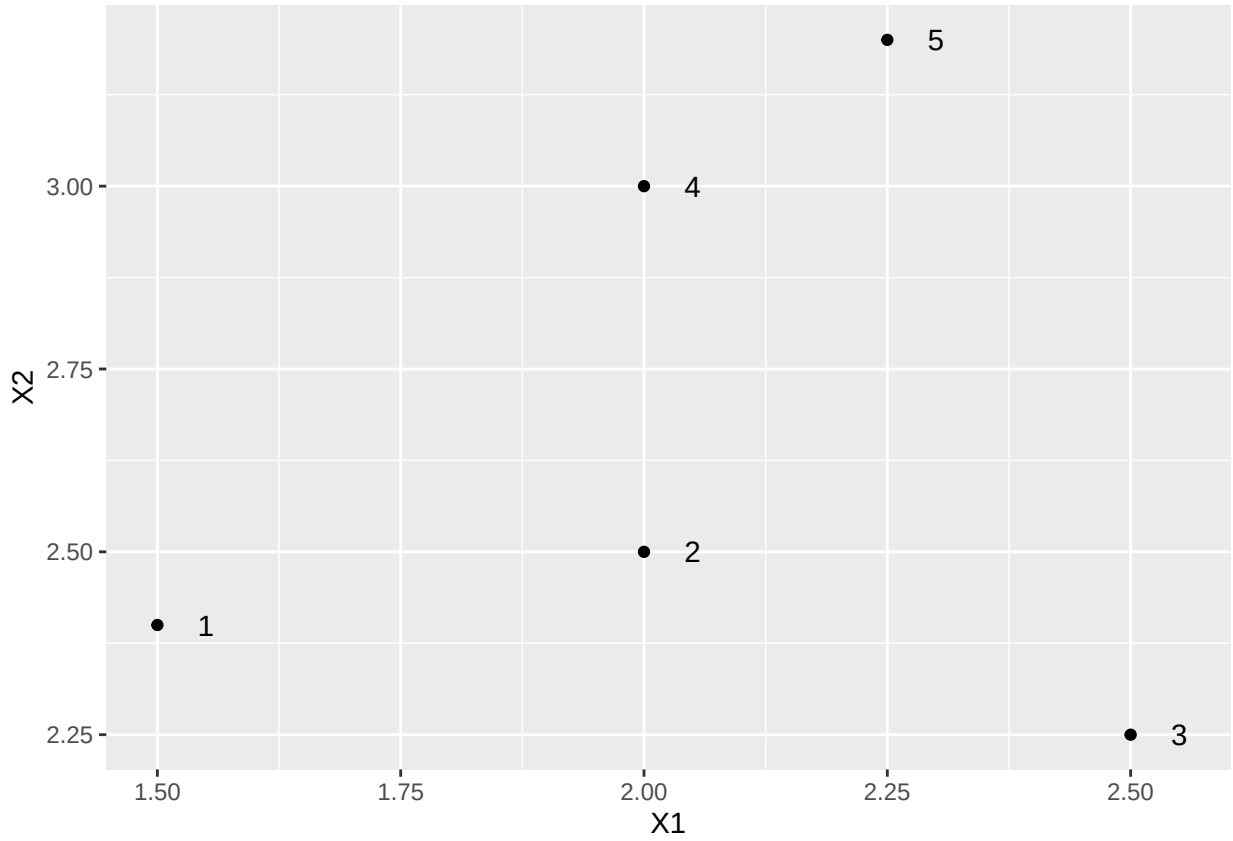
Hierarchical nature of the algorithm:

- Nested sequence of clusters $\longrightarrow$ visualisation via a tree
- Height of branches indicate the intercluster dissimilarity at which clusters are merged.
- Can used as instrument for deciding the number of clusters in the data

4 Toy example

X1	X2	label
1.50	2.40	1
2.00	2.50	2
2.50	2.25	3
2.00	3.00	4
2.25	3.20	5

```r
toy %>%
  ggplot(aes(X1, X2, label = label)) +
  geom_point() +
  geom_text(nudge_x = .05)
```

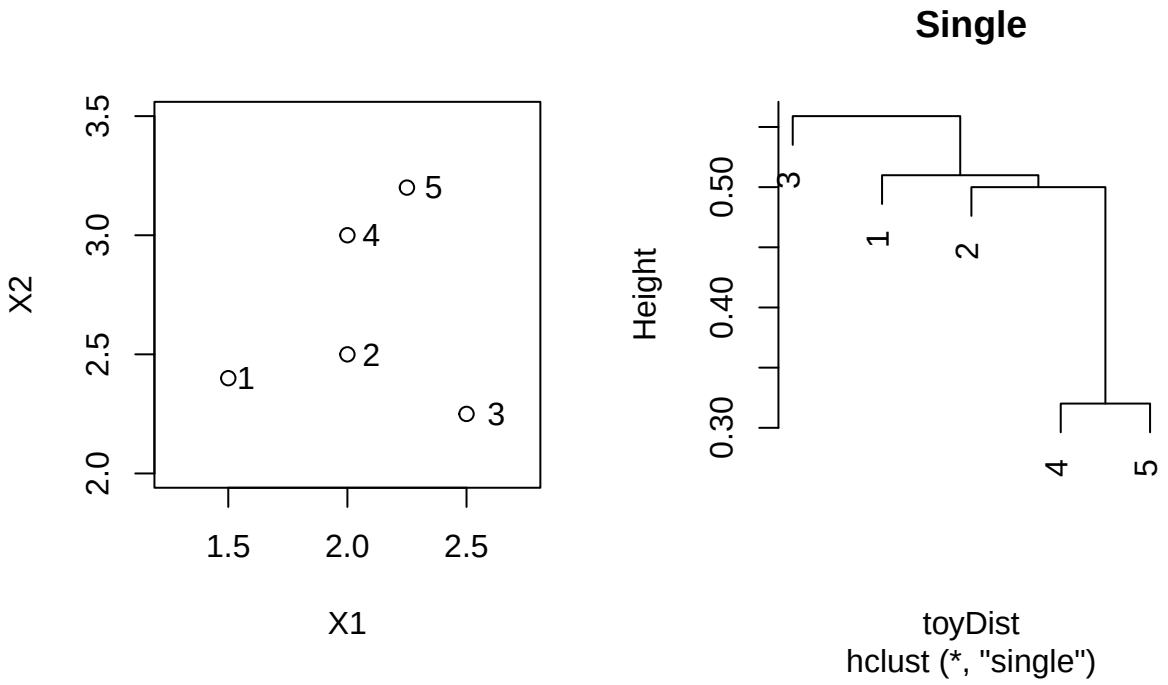



```r
toy[,1:2] %>% dist
```

```
#>           1         2         3         4
#> 2 0.5099020
#> 3 1.0111874 0.5590170
#> 4 0.7810250 0.5000000 0.9013878
#> 5 1.0965856 0.7433034 0.9823441 0.3201562
```

4.1 Single linkage

```r
toyDist <- toy[,1:2] %>% dist
toySingle <- hclust(toyDist, method = "single")
par(mfrow=c(1,2),pty="s")
plot(X2 ~ X1, toy, xlim = c(1.25,2.75),ylim = c(2,3.5))
text(toy$X1*1.05,toy$X2,label=toy$label)
plot(toySingle, main = "Single")
```

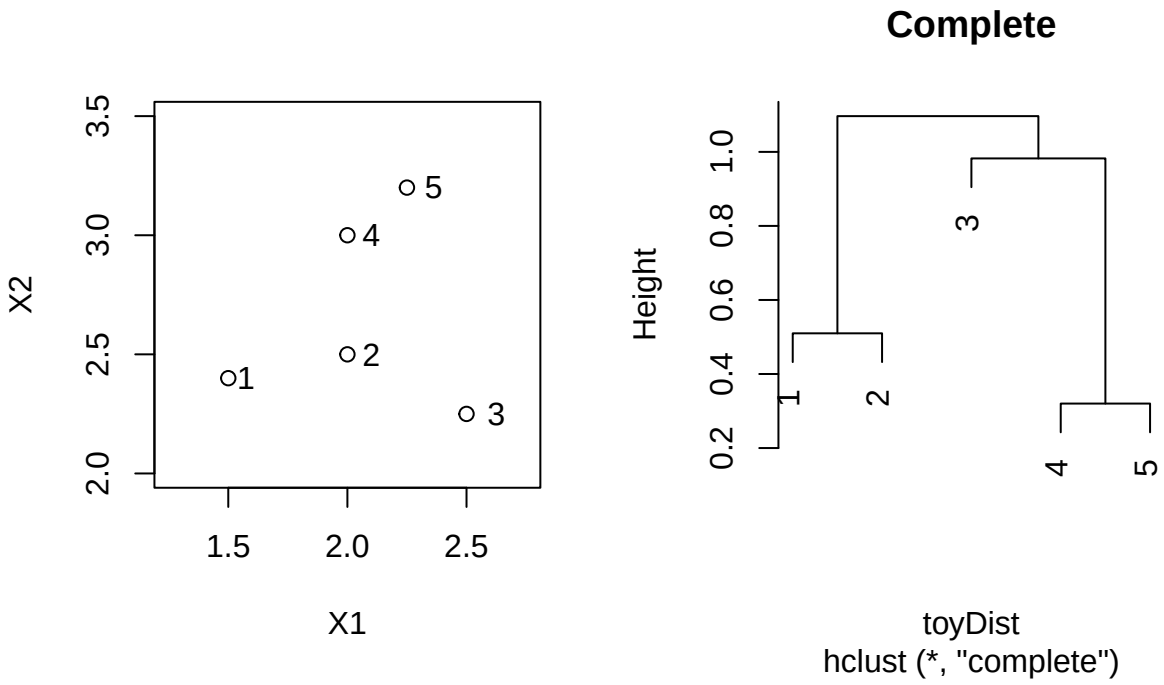



```
toyDist
```

```
#>           1         2         3         4
#> 2 0.5099020
#> 3 1.0111874 0.5590170
#> 4 0.7810250 0.5000000 0.9013878
#> 5 1.0965856 0.7433034 0.9823441 0.3201562
```

4.2 Complete linkage

```
toyComplete <- hclust(toyDist, method = "complete")
par(mfrow=c(1,2),pty="s")
plot(X2 ~ X1, toy, xlim = c(1.25,2.75),ylim = c(2,3.5))
text(toy$X1*1.05,toy$X2,label=toy$label)
plot(toyComplete,  main = "Complete")
```

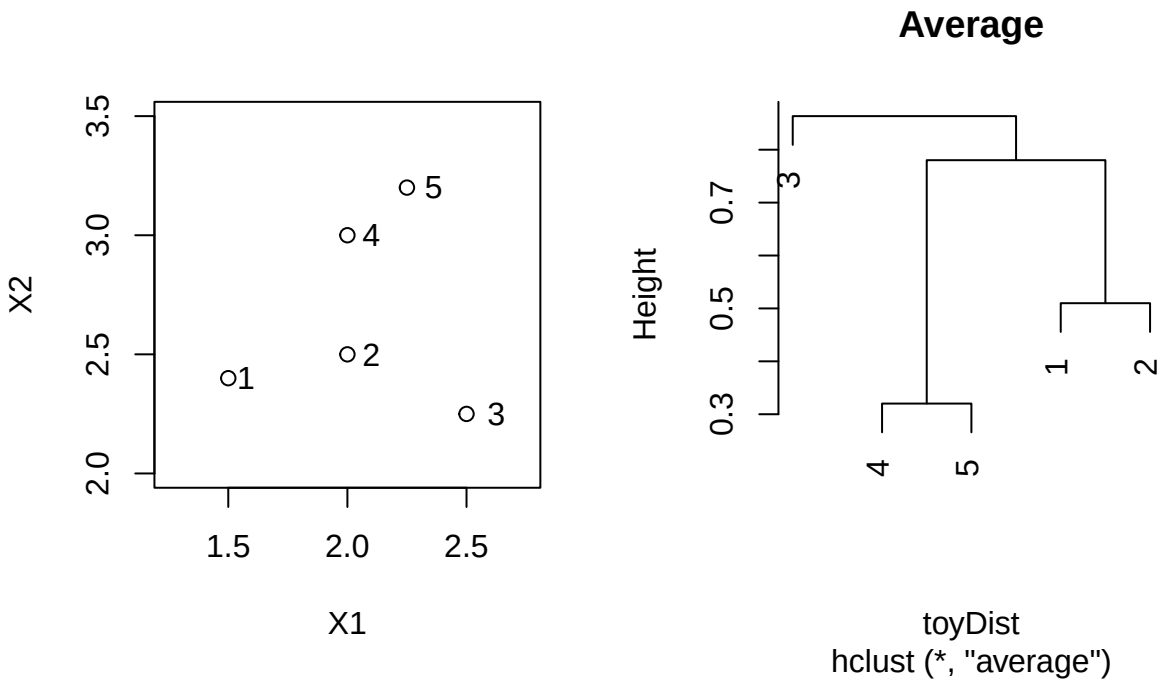



```
toyDist
```

```
#>           1         2         3         4
#> 2 0.5099020
#> 3 1.0111874 0.5590170
#> 4 0.7810250 0.5000000 0.9013878
#> 5 1.0965856 0.7433034 0.9823441 0.3201562
```

4.3 Average linkage

```
toyAvg <- hclust(toyDist, method = "average")
par(mfrow=c(1,2),pty="s")
plot(X2 ~ X1, toy, xlim = c(1.25,2.75),ylim = c(2,3.5))
text(toy$X1*1.05,toy$X2,label=toy$label)
plot(toyAvg, main = "Average")
```

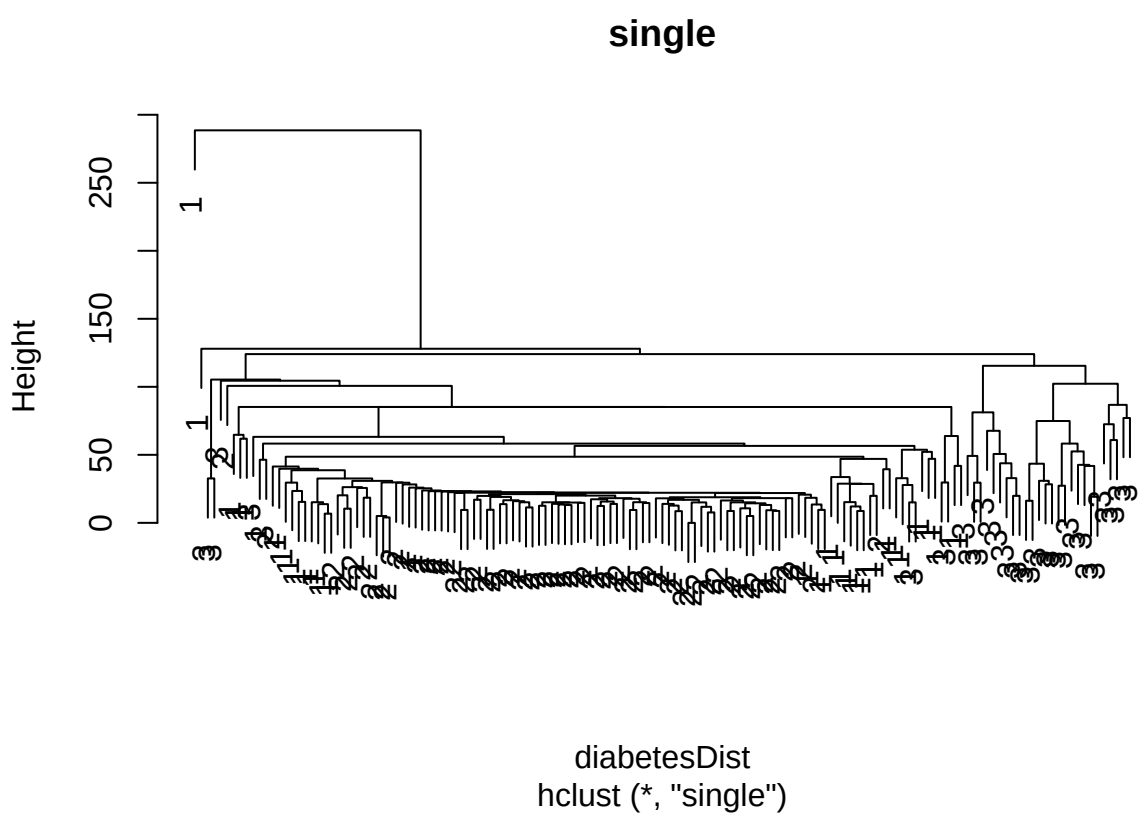



```
toyDist
```

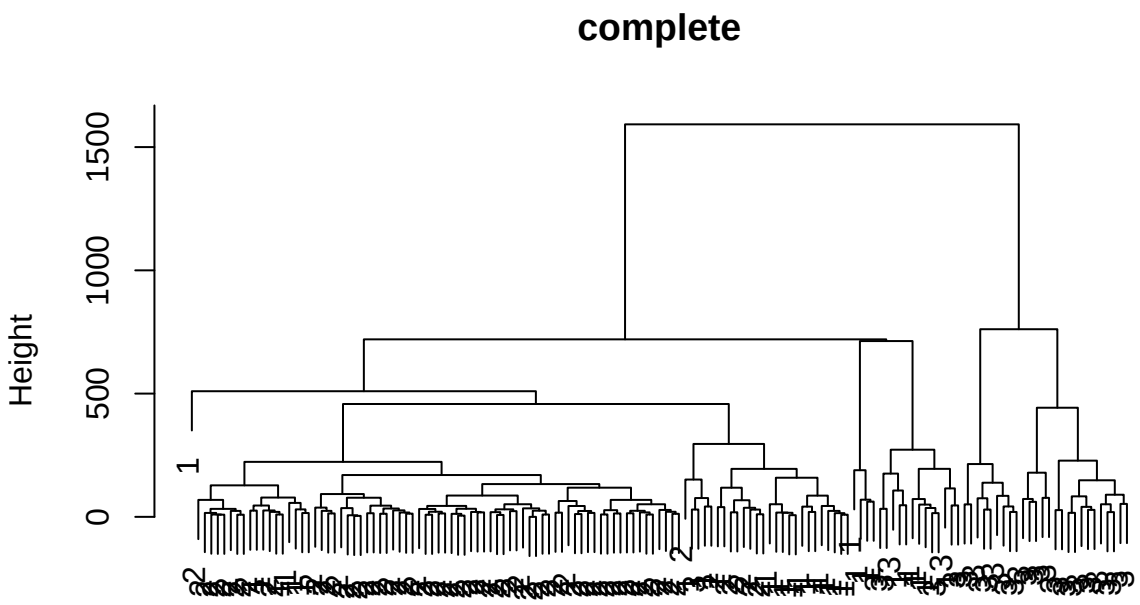
```
#>           1         2         3         4
#> 2 0.5099020
#> 3 1.0111874 0.5590170
#> 4 0.7810250 0.5000000 0.9013878
#> 5 1.0965856 0.7433034 0.9823441 0.3201562
```

4.4 Example

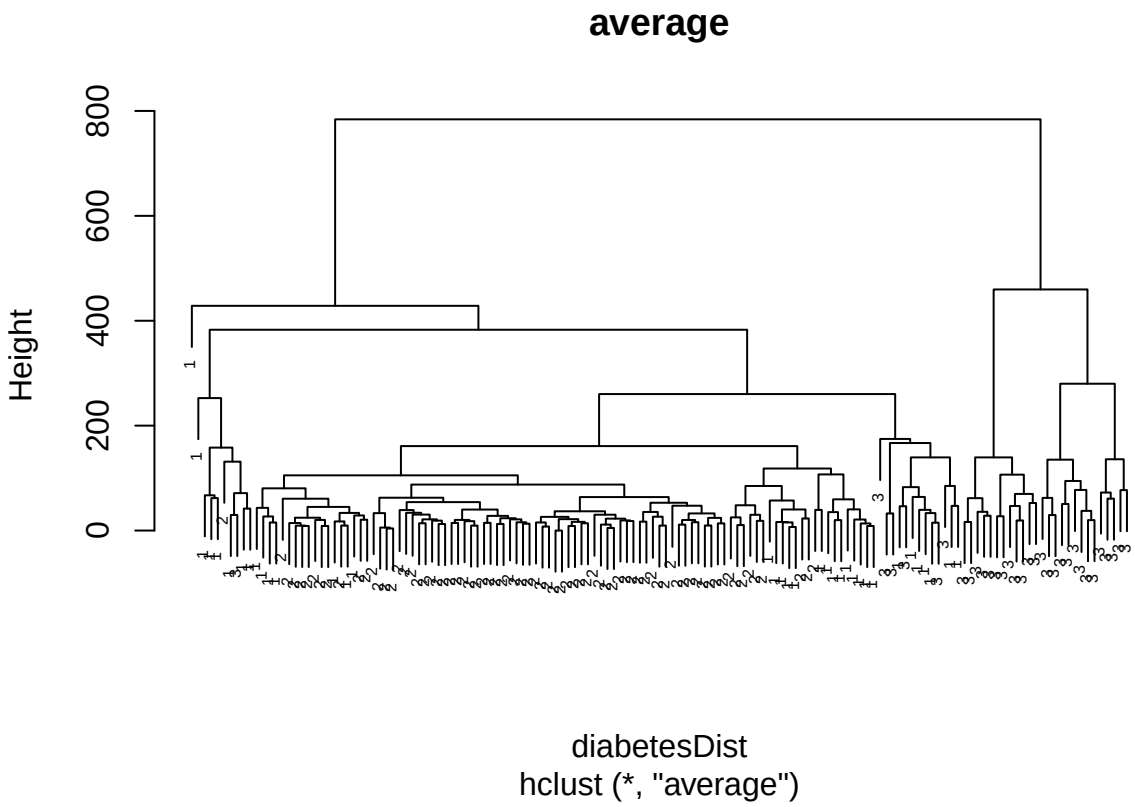
```
diabetesDist <- dist(diabetes[,-1])
diabetesSingle <- hclust(diabetesDist, method = "single")
plot(diabetesSingle, labels = as.double(diabetes$class), main="single")
```




```
diabetesComplete <- hclust(diabetesDist, method = "complete")
plot(diabetesComplete, labels = as.double(diabetes$class), main="complete")
```




```
diabetesAverage <- hclust(diabetesDist, method = "average")
plot(diabetesAverage, labels = as.double(diabetes$class), main = "average",cex=0.5)
```



5 Model-based clustering

- Paper: Fraley and Raftery (1998). How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. The Computer Journal, (41)8:578-588.
- EM algorithm [PDF]
- Example: see tutorial session

Session info

Session info

```
#> [1] "2024-10-03 06:22:04 UTC"

#> - Session info ---------------------------------------------------------------
#>  setting  value
#>  version  R version 4.1.3 (2022-03-10)
#>  os       Ubuntu 22.04.5 LTS
#>  system   x86_64, linux-gnu
#>  ui       X11
#>  language (EN)
#>  collate  C.UTF-8
```

```
#>  ctype    C.UTF-8
#>  tz       UTC
#>  date     2024-10-03
#>  pandoc   3.1.11 @ /opt/hostedtoolcache/pandoc/3.1.11/x64/ (via rmarkdown)
#>
#> - Packages -------------------------------------------------------------------
#>  ! package     * version date (UTC) lib source
#>  P assertthat    0.2.1   2019-03-21 [?] CRAN (R 4.1.3)
#>  P backports     1.4.1   2021-12-13 [?] CRAN (R 4.1.3)
#>  P BiocManager   1.30.16 2021-06-15 [?] CRAN (R 4.1.3)
#>  P bookdown      0.24    2021-09-02 [?] CRAN (R 4.1.3)
#>  P broom         0.7.11  2022-01-03 [?] CRAN (R 4.1.3)
#>  P bslib         0.3.1   2021-10-06 [?] CRAN (R 4.1.3)
#>  P cellranger    1.1.0   2016-07-27 [?] CRAN (R 4.1.3)
#>  P cli           3.1.1   2022-01-20 [?] CRAN (R 4.1.3)
#>  P colorspace    2.0-2   2021-06-24 [?] CRAN (R 4.1.3)
#>  P crayon        1.4.2   2021-10-29 [?] CRAN (R 4.1.3)
#>  P DBI           1.1.2   2021-12-20 [?] CRAN (R 4.1.3)
#>  P dbplyr        2.1.1   2021-04-06 [?] CRAN (R 4.1.3)
#>  P digest        0.6.29  2021-12-01 [?] CRAN (R 4.1.3)
#>  P dplyr       * 1.0.7   2021-06-18 [?] CRAN (R 4.1.3)
#>  P ellipsis      0.3.2   2021-04-29 [?] CRAN (R 4.1.3)
#>  P evaluate      0.14    2019-05-28 [?] CRAN (R 4.1.3)
#>  P fansi         1.0.2   2022-01-14 [?] CRAN (R 4.1.3)
#>  P farver        2.1.0   2021-02-28 [?] CRAN (R 4.1.3)
#>  P fastmap       1.1.0   2021-01-25 [?] CRAN (R 4.1.3)
#>  P forcats     * 0.5.1   2021-01-27 [?] CRAN (R 4.1.3)
#>  P fs            1.5.2   2021-12-08 [?] CRAN (R 4.1.3)
#>  P generics      0.1.1   2021-10-25 [?] CRAN (R 4.1.3)
#>  P ggplot2     * 3.3.5   2021-06-25 [?] CRAN (R 4.1.3)
#>  P glue          1.6.1   2022-01-22 [?] CRAN (R 4.1.3)
#>  P gtable        0.3.0   2019-03-25 [?] CRAN (R 4.1.3)
#>  P haven         2.4.3   2021-08-04 [?] CRAN (R 4.1.3)
#>  P highr         0.9     2021-04-16 [?] CRAN (R 4.1.3)
#>  P hms           1.1.1   2021-09-26 [?] CRAN (R 4.1.3)
#>  P htmltools     0.5.2   2021-08-25 [?] CRAN (R 4.1.3)
#>  P httr          1.4.2   2020-07-20 [?] CRAN (R 4.1.3)
#>  P jquerylib     0.1.4   2021-04-26 [?] CRAN (R 4.1.3)
#>  P jsonlite      1.7.3   2022-01-17 [?] CRAN (R 4.1.3)
#>  P knitr         1.37    2021-12-16 [?] CRAN (R 4.1.3)
#>  P labeling      0.4.2   2020-10-20 [?] CRAN (R 4.1.3)
#>  P lifecycle     1.0.1   2021-09-24 [?] CRAN (R 4.1.3)
#>  P lubridate     1.8.0   2021-10-07 [?] CRAN (R 4.1.3)
#>  P magrittr      2.0.2   2022-01-26 [?] CRAN (R 4.1.3)
#>  P mclust        5.4.9   2021-12-17 [?] CRAN (R 4.1.3)
#>  P modelr        0.1.8   2020-05-19 [?] CRAN (R 4.1.3)
#>  P munsell       0.5.0   2018-06-12 [?] CRAN (R 4.1.3)
#>  P pillar        1.6.5   2022-01-25 [?] CRAN (R 4.1.3)
#>  P pkgconfig     2.0.3   2024-10-02 [?] Github (r-lib/pkgconfig@b81ae03)
#>  P purrr       * 0.3.4   2020-04-17 [?] CRAN (R 4.1.3)
#>  P R6            2.5.1   2021-08-19 [?] CRAN (R 4.1.3)
#>  P Rcpp          1.0.8   2022-01-13 [?] CRAN (R 4.1.3)
#>  P readr       * 2.1.1   2021-11-30 [?] CRAN (R 4.1.3)
#>  P readxl        1.3.1   2019-03-13 [?] CRAN (R 4.1.3)
```

```
#>    renv          0.15.2  2022-01-24 [1] CRAN (R 4.1.3)
#>  P reprex        2.0.1   2021-08-05 [?] CRAN (R 4.1.3)
#>  P rlang         1.0.0   2022-01-26 [?] CRAN (R 4.1.3)
#>  P rmarkdown     2.11    2021-09-14 [?] CRAN (R 4.1.3)
#>  P rstudioapi    0.13    2020-11-12 [?] CRAN (R 4.1.3)
#>  P rvest         1.0.2   2021-10-16 [?] CRAN (R 4.1.3)
#>  P sass          0.4.0   2021-05-12 [?] CRAN (R 4.1.3)
#>  P scales        1.1.1   2020-05-11 [?] CRAN (R 4.1.3)
#>  P sessioninfo   1.2.2   2021-12-06 [?] CRAN (R 4.1.3)
#>  P stringi       1.7.6   2021-11-29 [?] CRAN (R 4.1.3)
#>  P stringr     * 1.4.0   2019-02-10 [?] CRAN (R 4.1.3)
#>  P tibble      * 3.1.6   2021-11-07 [?] CRAN (R 4.1.3)
#>  P tidyr       * 1.1.4   2021-09-27 [?] CRAN (R 4.1.3)
#>  P tidyselect    1.1.1   2021-04-30 [?] CRAN (R 4.1.3)
#>  P tidyverse   * 1.3.1   2021-04-15 [?] CRAN (R 4.1.3)
#>  P tzdb          0.2.0   2021-10-27 [?] CRAN (R 4.1.3)
#>  P utf8          1.2.2   2021-07-24 [?] CRAN (R 4.1.3)
#>  P vctrs         0.3.8   2021-04-29 [?] CRAN (R 4.1.3)
#>  P withr         2.4.3   2021-11-30 [?] CRAN (R 4.1.3)
#>  P xfun          0.29    2021-12-14 [?] CRAN (R 4.1.3)
#>  P xml2          1.3.3   2021-11-30 [?] CRAN (R 4.1.3)
#>  P yaml          2.2.2   2022-01-25 [?] CRAN (R 4.1.3)
#>
#>  [1] /home/runner/work/HDDA23/HDDA23/renv/library/R-4.1/x86_64-pc-linux-gnu
#>  [2] /opt/R/4.1.3/lib/R/library
#>
#>  P -- Loaded and on-disk path mismatch.
#>
#> ------------------------------------------------------------------------------
```