Introduction to Machine Learning

Module 3: Unsupervised Learning

Tobias Rebholz

University of Tübingen

Fall 2024, SMiP Workshop

Unsupervised Learning

Unsupervised Learning

- For each individual i in a population, we have:
  - Vector of features, $X_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$
  - **No target**
- Goal: Discover "interesting things" (James et al., 2021)
  - I.e., patterns, structures, or relationships within the data
  - In other words, we are not interested in prediction, because there is no target that could eventually be predicted

Unsupervised Learning Tasks

- **Clustering:** Grouping similar instances together based on certain criteria
  - I.e., discovering unknown subgroups in the dataset
- **Dimensionality reduction:** Reducing the number of features in a dataset while preserving its critical structure and important relationships
  - Used for visualization or data pre-processing prior to applying other ML techniques
- **Anomaly detection:** Identifying outliers or anomalies in the data
  - I.e., looking for significant deviations from the norm
- **Association rule mining:** Discovering interesting relationships or associations between variables in large datasets (not discussed!)
  - Example: If a customer buys strawberries, do they also always buy cream?
- **Latent class models** (not discussed!):
  - Mixture of distributions (e.g. 20% [80%] normal with mean 0 [1])
  - Mixture of regression models (e.g., one subgroup with a positive trend and another with a negative trend, where subgroup memberships are unknown)
- ...

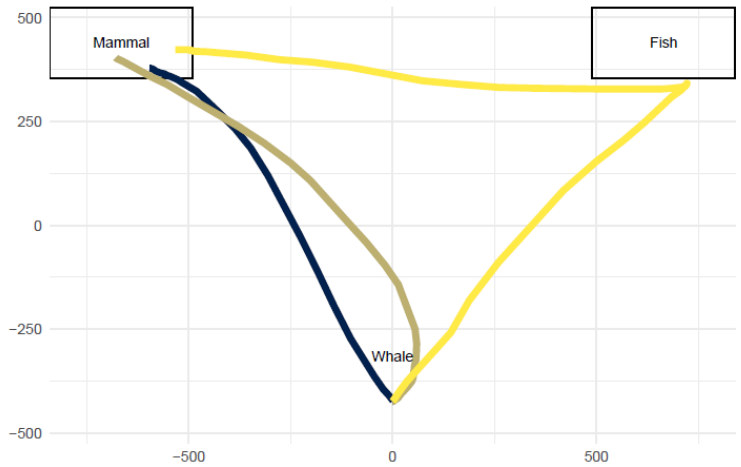
Clustering

Clustering

- Goal: Grouping instances that are similar into a cluster
  - Intra-cluster homogeneity: Instances within a cluster should be similar
  - Inter-cluster heterogeneity: Clusters should be dissimilar to other clusters
- Conditions:
  - Each instance is in one group
  - The groups do not overlap
- Terminology: "clustering" = collection of clusters
- Types:
  - Hierarchical
  - Partitional (e.g., k-means, k-medians)

Applications in Behavioral Science: Process Tracing

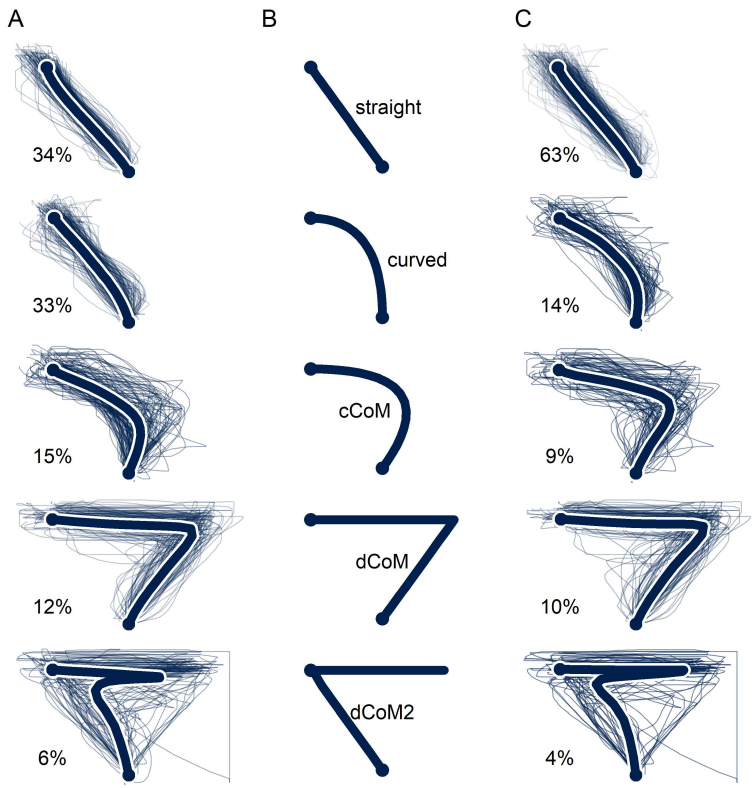
- Movement tracing: Tracking reaching movements of the hand (e.g., using a computer mouse)
  - Participants choose between two spatially separated options in a two-options forced-choice paradigm
  - Movement trajectory: Assumed to reflect the psychological processes that gave rise to the final choice



(Wulff et al., 2023, Figure 1)

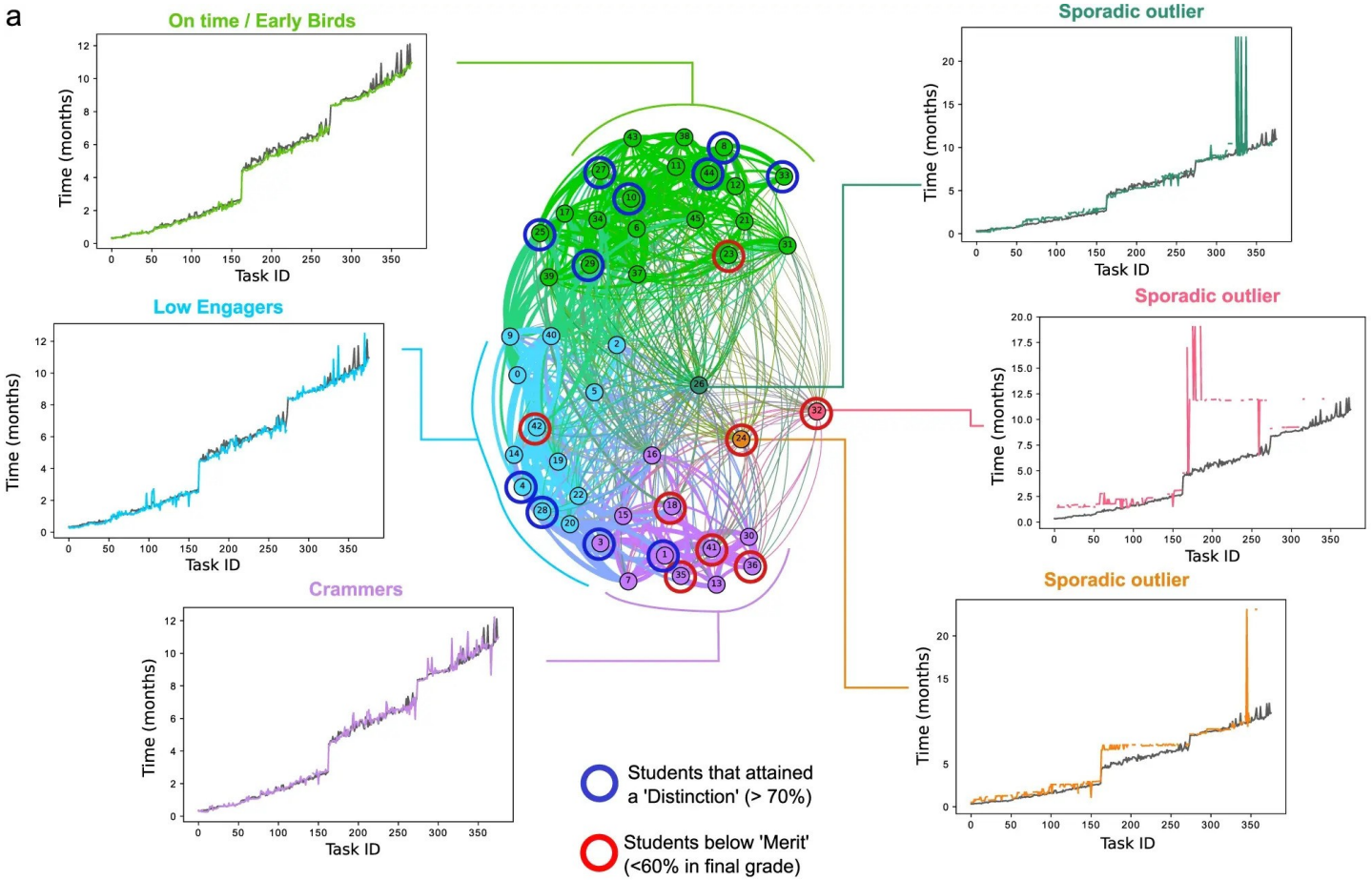
Applications in Behavioral Science: Process Tracing

- With chaotic motion data, how to know what strategy someone used?



(Wulff et al., 2023, Figure 8, where cCoM = continuous change of mind, dCoM = discrete change of mind, and dCoM2 = discrete double change of mind)

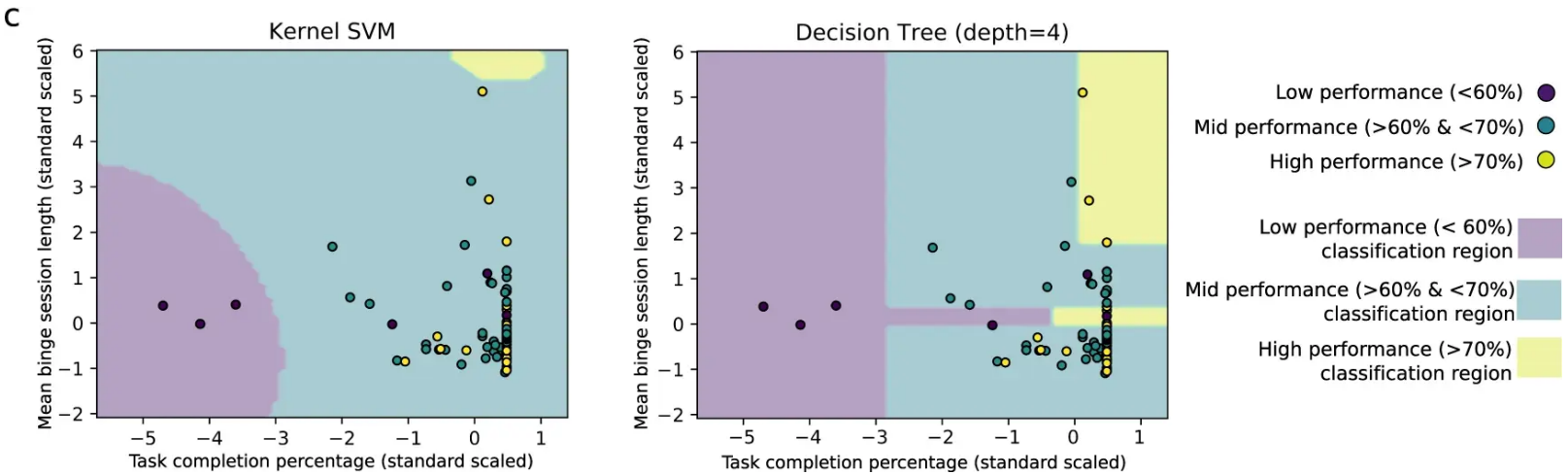
Applications in Behavioral Science: Learning Curves



(Peach et al., 2019, Figure 6a)

Applications in Behavioral Science: Learning Curves

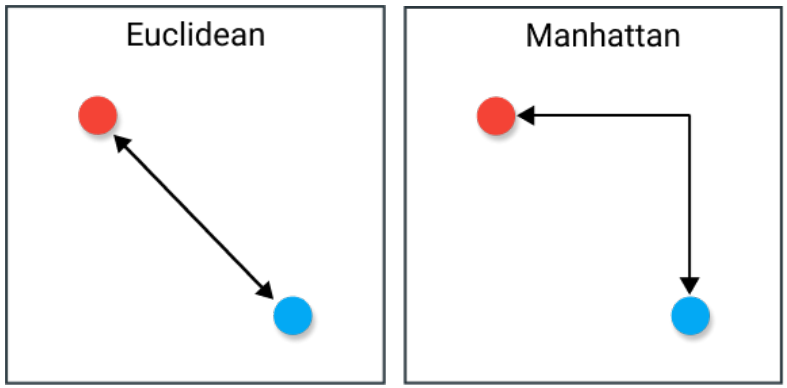
- Note: Learners in Peach et al. (2019) were classifier according to their performance using SVMs and Decision Trees



(Peach et al., 2019, Figure 5c)

Dissimilarity

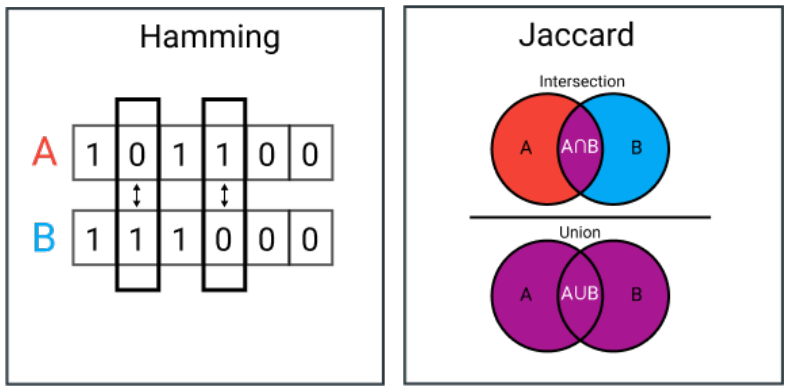
- Measuring dissimilarity between instances through distance:
  - Euclidean distance: $d(X_1, X_2) = \sqrt{(x_{11} - x_{21})^2 + \cdots + (x_{1p} - x_{2p})^2}$
  - Manhattan distance: $d(X_1, X_2) = |x_{11} - x_{21}| + \cdots + |x_{1p} - x_{2p}|$
  - Maximum distance: $d(X_1, X_2) = \max\{|x_{11} - x_{21}|, \ldots, |x_{1p} - x_{2p}|\}$
  - ...



(https://www.maartengrootendorst.com/blog/distances/)

Dissimilarity

- Distances for categorical variables:
  - Hamming distance: $d(X_i, X_j) = |\{s : x_{is} \neq x_{js}\}|$
    - Note: $|.|$ denotes the cardinality (i.e., number of unique levels)
  - Jaccard distance: $d(X_i, X_j) = 1 - \frac{X_i \cap X_j}{X_i \cup X_j}$
  - ...



(https://www.maartengrootendorst.com/blog/distances/)

Dissimilarity

- Dissimilarity is measured as the distance between two instances
  - But, what is the distance (or "linkage") between two clusters (i.e., groups of instances)?

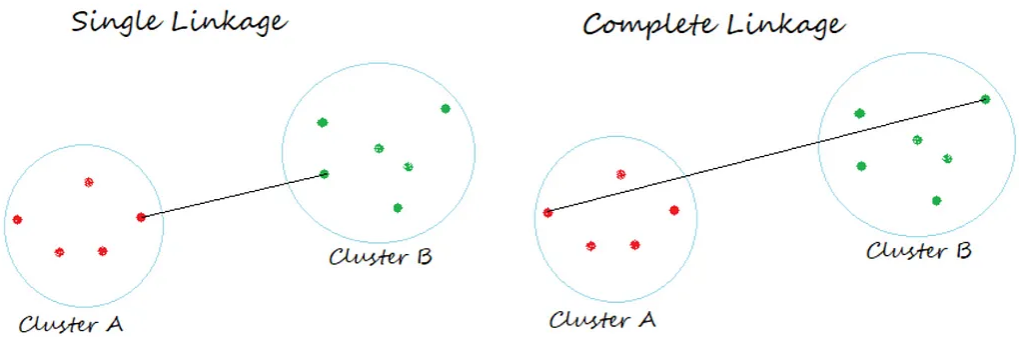
Dissimilarity

- Linkage between sets S_A and S_B:
  - **Single:** Minimal intercluster dissimilarity
    - I.e., smallest pairwise dissimilarity between all instances

$$L(S_A, S_B) = \underset{X_A \in S_A, X_B \in S_B}{\text{minimize}} \; d(X_A, X_B) \tag{1}$$

  - **Complete:** Maximal intercluster dissimilarity
    - I.e., largest pairwise dissimilarity between all instances

$$L(S_A, S_B) = \underset{X_A \in S_A, X_B \in S_B}{\text{maximize}} \; d(X_A, X_B) \tag{2}$$



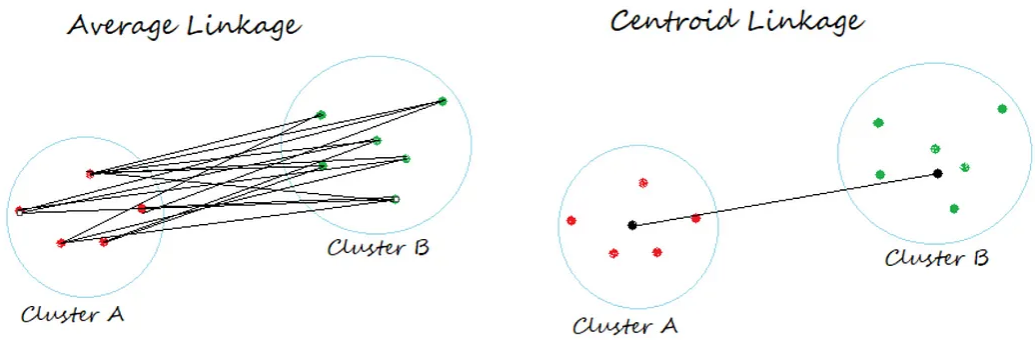
(https://harikabonthu96.medium.com/single-link-clustering-clearly-explained-90dff58db5cb)

Dissimilarity

- Linkage between sets S_A and S_B:
  - **Average:** Mean dissimilarity between all pairs of instances in both sets

$$L(S_A, S_B) = \frac{1}{|S_A| \times |S_B|} \sum_{X_A \in S_A, X_B \in S_B} d(X_A, X_B) \tag{3}$$

  - **Centroid:** Dissimilarity between the centroids (or "prototypes") for clusters A and B
    - Centroid = mean vector of length p (think of a prototipical instance)

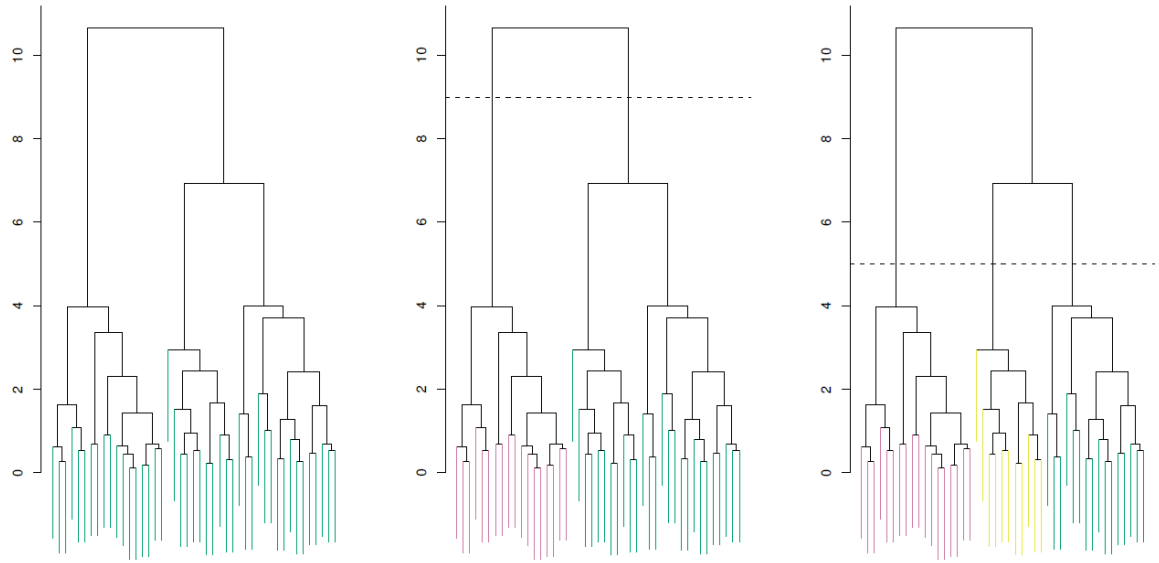


(https://harikabonthu96.medium.com/single-link-clustering-clearly-explained-90dff58db5cb)

Hierarchical Clustering

Hierarchical Clustering

- Idea: There is a hierarchy of clusters
  - Top: All individuals together in one huge cluster (i.e., whole sample)
  - Bottom: Each instance in a cluster by itself (i.e., individual observations)
- Procedure: By giving a horizontal cut, we obtain a clustering
  - Visualization: Dendrogram (cf. decision trees)
  - Different horizontal cuts give different clusterings:



(James et al., 2021, Figure 12.11)

Hierarchical Clustering in mlr3

- From a subset of Kaggle data, we have information on certain features (e.g., income) of customers of a store
  - Note: For simplicity, we exclude discrete variables (i.e., gender)

```
df <- dat_subset %>% select(-Gender)
head(df)
##   CustomerID Age Annual_Income Spending_Score
## 1         20  35            23             98
## 2         24  31            25             73
## 3         47  50            40             55
## 4         49  29            40             42
## 5         65  63            48             51
## 6         71  70            49             55
```

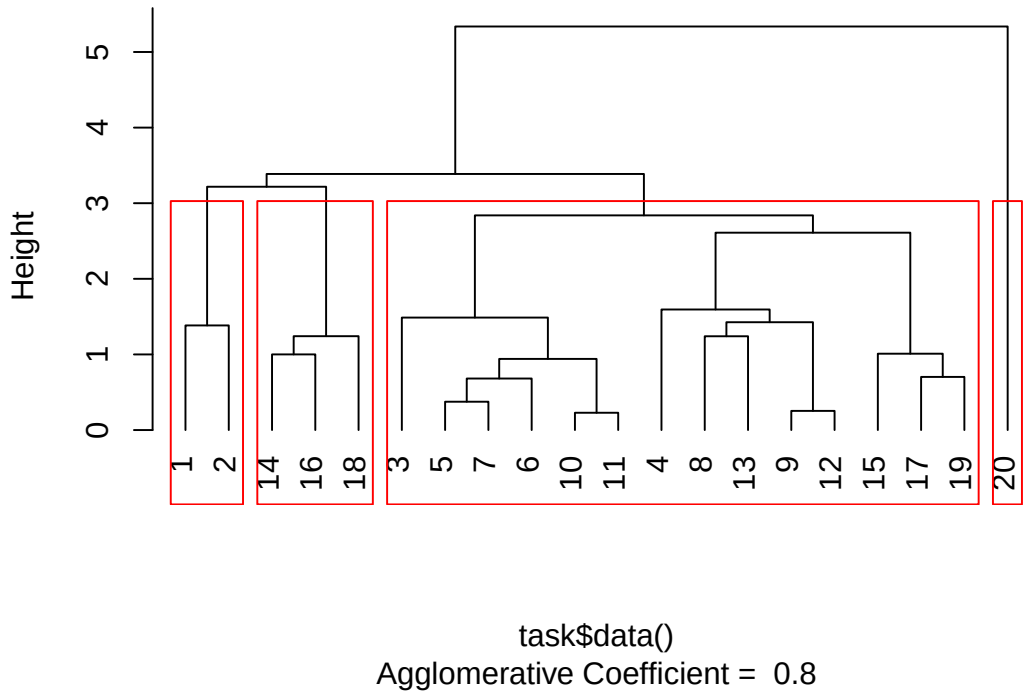
- Goal: Customer segmentation
  - By defining a clustering task using `as_task_clust()`, we want to group similar customers together in a total of `k = 4` groups

```
tsk = as_task_clust(df %>% select(-CustomerID))
k = 4
```

Hierarchical Clustering in `mlr3`

- **Agglomerative:** Start at the bottom, where each instance consitutes an individual cluster $\rightarrow$ **Merge** clusters that are similar

```
mdl_aggl = lrn("clust.agnes", stand = T, k = k)
mdl_aggl$train(tsk)

plot(mdl_aggl$model, hang = -1, which.plots = 2, main = "")
rect.hclust(as.hclust(mdl_aggl$model), k = k, border = "red")
```



Hierarchical Clustering in mlr3

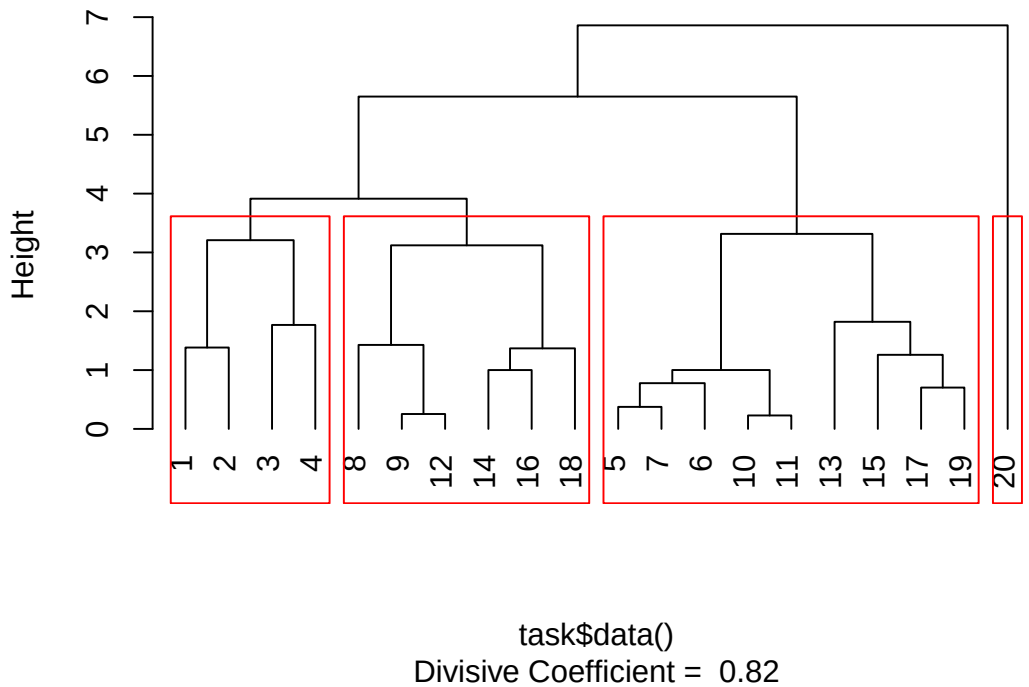
- Agglomerative clustering predictions:

```
pred_aggl <- mdl_aggl$predict(tsk)
## Warning: Learner 'clust.agnes' doesn't predict on new data and predictions may
## not make sense on new data.
pred_aggl
## <PredictionClust> for 20 observations:
##  row_ids partition
##        1         1
##        2         1
##        3         2
##      ---       ---
##       18         3
##       19         2
##       20         4

table(pred_aggl$partition)
##
##  1  2  3  4
##  2 14  3  1
```

Hierarchical Clustering in mlr3

- **Divisive:** Start at the top, where all individuals are in the same cluster (i.e., whole sample) → **Split** clusters that are diverse

```
mdl_divi = lrn("clust.diana", stand = T, k = k)
mdl_divi$train(tsk)

plot(mdl_divi$model, hang = -1, which.plots = 2, main = "")
rect.hclust(as.hclust(mdl_divi$model), k = k, border = "red")
```



Hierarchical Clustering in mlr3

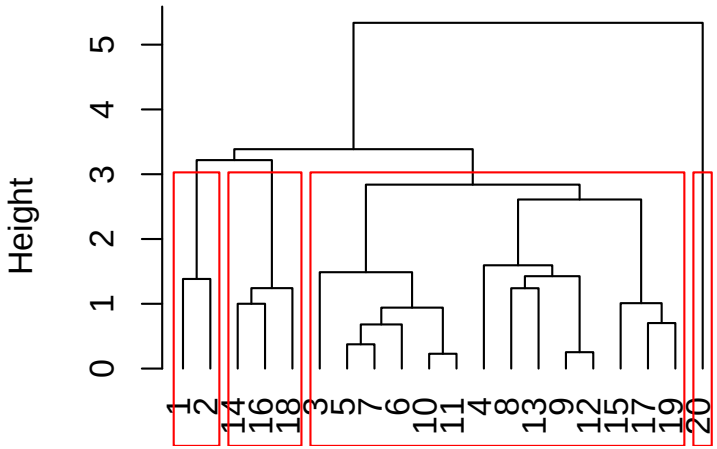
- Divisive clustering predictions:

```
pred_divi <- mdl_divi$predict(tsk)
## Warning: Learner 'clust.diana' doesn't predict on new data and predictions may
## not make sense on new data.
pred_divi
## <PredictionClust> for 20 observations:
##  row_ids partition
##        1         1
##        2         1
##        3         1
##      ---       ---
##       18         3
##       19         2
##       20         4

table(pred_divi$partition)
##
## 1 2 3 4
## 4 9 6 1
```

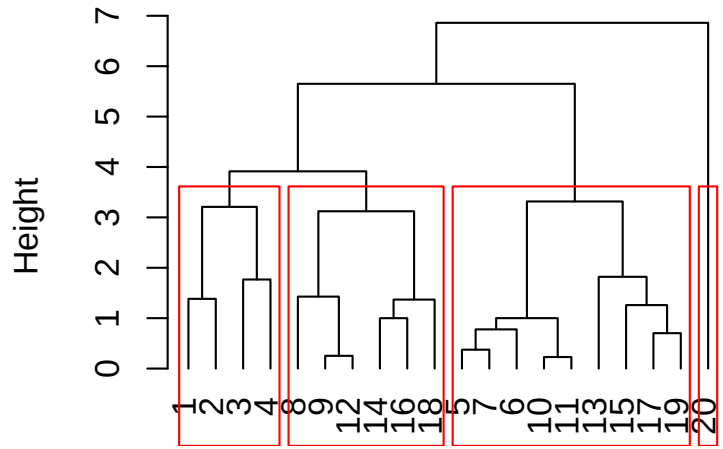
Hierarchical Clustering in mlr3

- Comparison of agglomerative and divisive clustering predictions:

```
table('aggl' = pred_aggl$partition, 'divi' = pred_divi$partition)
##      divi
## aggl  1  2  3  4
##    1  2  0  0  0
##    2  2  9  3  0
##    3  0  0  3  0
##    4  0  0  0  1
```



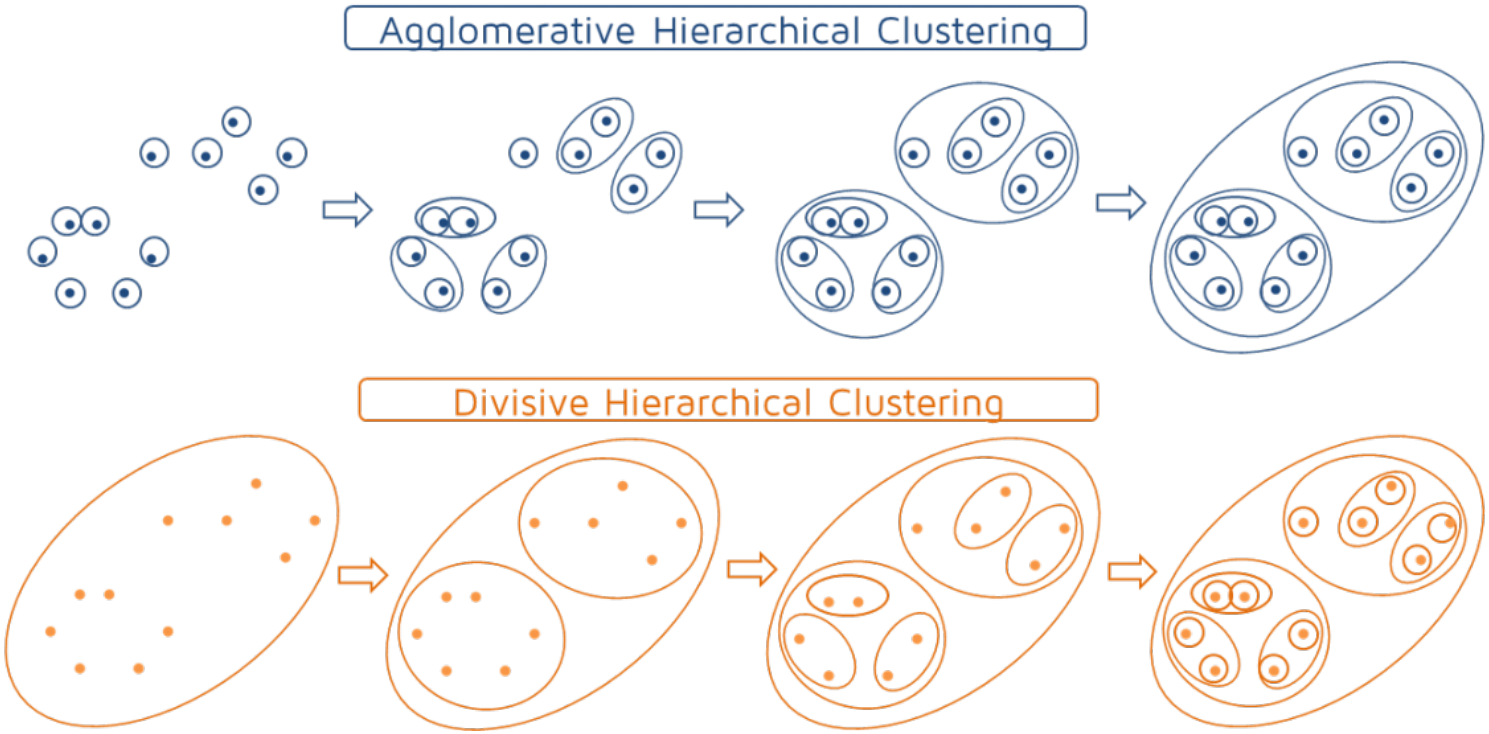
task$data()
Agglomerative Coefficient = 0.8



task$data()
Divisive Coefficient = 0.82

Summary

- Comparison of agglomerative and divisive clustering approaches:

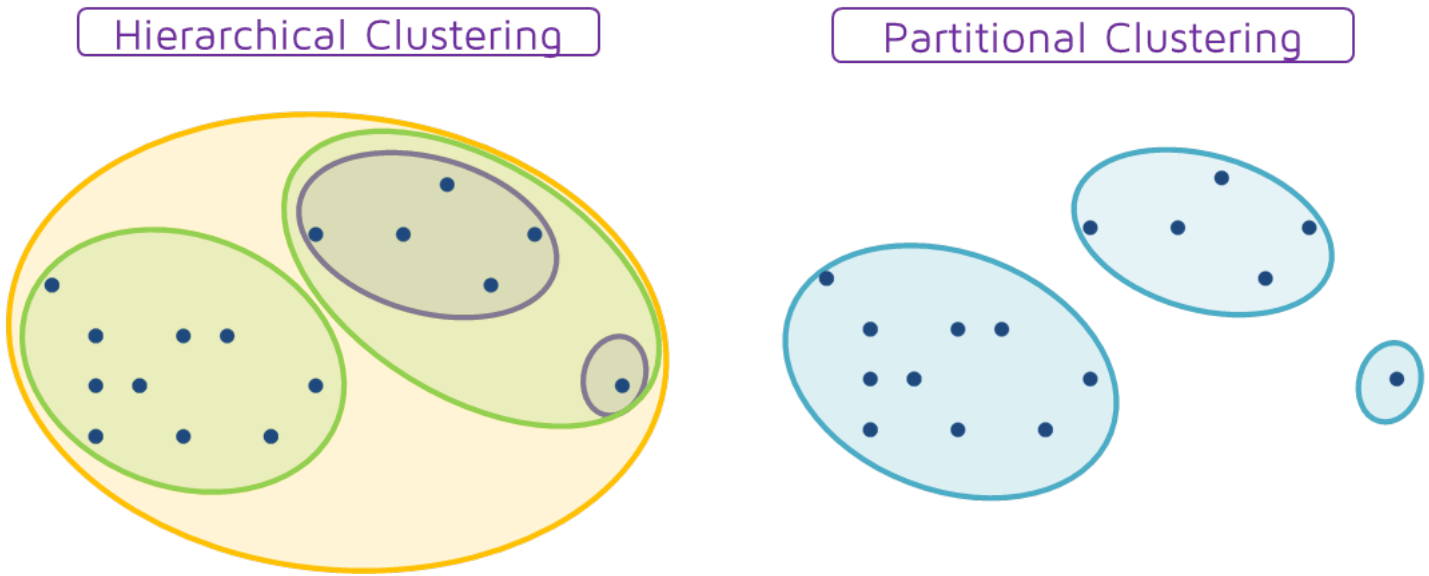


(https://quantdare.com/hierarchical-clustering/)

Partitional Clustering

Partitional Clustering

- The hierarchical method permits clusters to have subclusters (cf. decision tree)
  - Consequence: Once a cluster is formed, it cannot be split or combined with other clusters anymore
- **Partitional clustering:** More flexible in terms of reshaping the clusters



(https://quantdare.com/hierarchical-clustering/)

Partitional Clustering

- Problem: Partitioning a dataset of N individuals into k clusters is combinatorially hard because there are many possible combinations
  - For N individuals and k clusters, the number of possible partitions is:

$$\sum_{l=1}^{k} (-1)^{k-l} \frac{1}{(k-l)!l!} l^N \tag{4}$$

  - E.g., for $N = 20$ instances and $k = 4$ clusters: Extremely large number of more than 4×10^{10} possible partitions
- Goal: Find the best partitioning using iterative procedures that alleviate this computational complexity
  - Across all clusters, minimizing the within cluster distance to its centroid, denoted P_l, serving as a prototypical representative of cluster l, to improve the total intra-cluster homogeneity

$$\text{minimize} \sum_{l=1}^{k} \sum_{i \in C_l} d(X_i, P_l) \tag{5}$$

k-means Clustering

The iterative partitioning algorithm of k-means clustering:

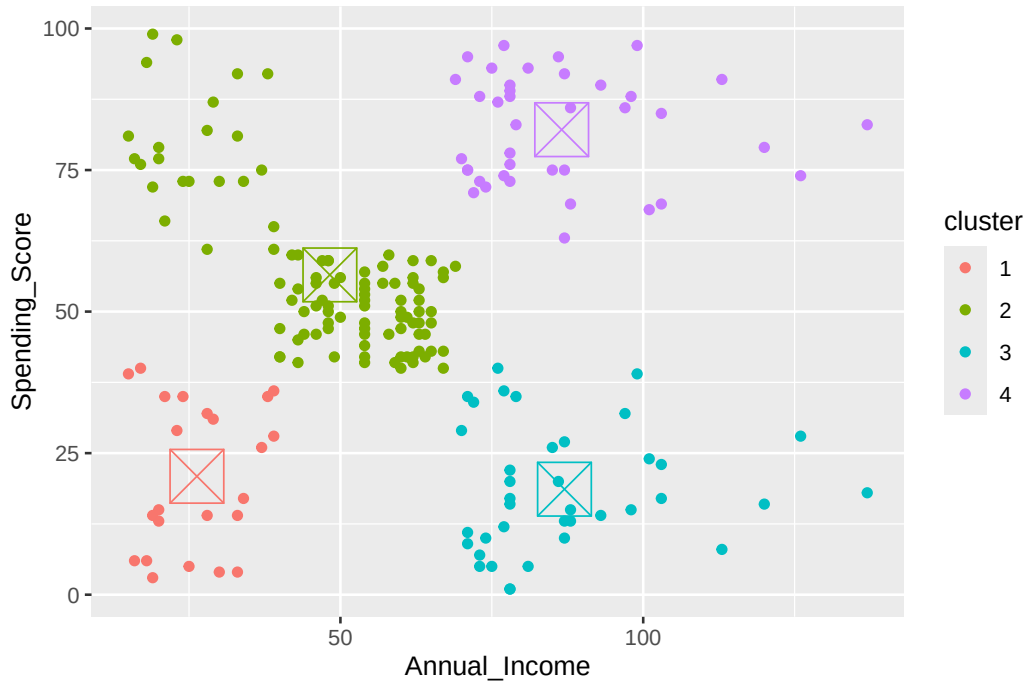
1. Choosing k initial centroids $P_1, \ldots, P_k$
   - $P_1, \ldots, P_k$, which eventually represents the vector of means of each cluster, can be randomly selected data points or even be randomly generated numbers
   - Multistart $\Rightarrow$ Avoids getting trapped in local optima
2. Clustering around the centroids from Step 1: Assigning each data point the cluster of its closest centroid
   - Goal: Minimizing the total within sum of squares (i.e., improving the total intra-cluster homogeneity)
3. Calculate the new centroids: Taking the mean of all data points assigned to each centroid's cluster: $P_l = \mu_l = \frac{1}{|C_l|} \sum_{i \in C_l} X_i$
4. Repeat steps 2 and 3 until the centroids do not change significantly anymore, or a maximum number of iterations is reached

k-means Clustering in `mlr3`

```
set.seed(42)
df <- dat %>% select(-c(CustomerID, Gender, Age))
tsk = as_task_clust(df)

mdl = lrn("clust.kmeans", centers = k)
mdl$train(tsk)
mdl$model
## K-means clustering with 4 clusters of sizes 23, 100, 38, 39
##
## Cluster means:
##   Annual_Income Spending_Score
## 1      26.30435       20.91304
## 2      48.26000       56.48000
## 3      87.00000       18.63158
## 4      86.53846       82.12821
##
## Clustering vector:
##   [1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1
##  [38] 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##  [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [112] 2 2 2 2 2 2 2 2 2 2 2 2 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
## [149] 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3
## [186] 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4
##
## Within cluster sum of squares by cluster:
## [1]  5098.696 40932.200 14204.842 13444.051
##  (between_SS / total_SS =  72.7 %)
```

k-means Clustering in `mlr3`

```
ggplot() +
  # Plot data:
  geom_point(data = df %>% mutate(cluster = factor(mdl$model$cluster)),
             aes(x = Annual_Income, y = Spending_Score, col = cluster)) +
  # Add prototypes:
  geom_point(data = as.data.frame(mdl$model$centers) %>% rownames_to_column('cluster'),
             aes(x = Annual_Income, y = Spending_Score, col = cluster),
             size = 10, shape = 7, show.legend = F)
```



k-means Clustering in `mlr3`

- Hyperparameter tuning: Number of clusters k

```
k_cv <- seq(2,10)

mdl_cv = auto_tuner(
  learner = lrn("clust.kmeans", centers = to_tune(levels = k_cv)),
  resampling = rsmp("cv", folds = 5),
  measure = msr("clust.dunn"),
  tuner = tnr("grid_search"),
  terminator = trm("none")
)

set.seed(42)
mdl_cv$train(tsk)
## INFO  [09:31:21.137] [bbotk] Starting to optimize 1 parameter(s) with '<OptimizerBatchGr
## INFO  [09:31:21.219] [bbotk] Evaluating 1 configuration(s)
## INFO  [09:31:21.323] [mlr3] Running benchmark with 5 resampling iterations
## INFO  [09:31:21.416] [mlr3] Applying learner 'clust.kmeans' on task 'df' (iter 1/5)
## INFO  [09:31:21.454] [mlr3] Applying learner 'clust.kmeans' on task 'df' (iter 2/5)
## INFO  [09:31:21.493] [mlr3] Applying learner 'clust.kmeans' on task 'df' (iter 3/5)
## INFO  [09:31:21.529] [mlr3] Applying learner 'clust.kmeans' on task 'df' (iter 4/5)
## INFO  [09:31:21.564] [mlr3] Applying learner 'clust.kmeans' on task 'df' (iter 5/5)
## INFO  [09:31:21.592] [mlr3] Finished benchmark
## INFO  [09:31:21.703] [bbotk] Result of batch 1:
## INFO  [09:31:21.714] [bbotk]  centers clust.dunn warnings errors runtime_learners
## INFO  [09:31:21.714] [bbotk]        4  0.1452742        0      0             0.05
```

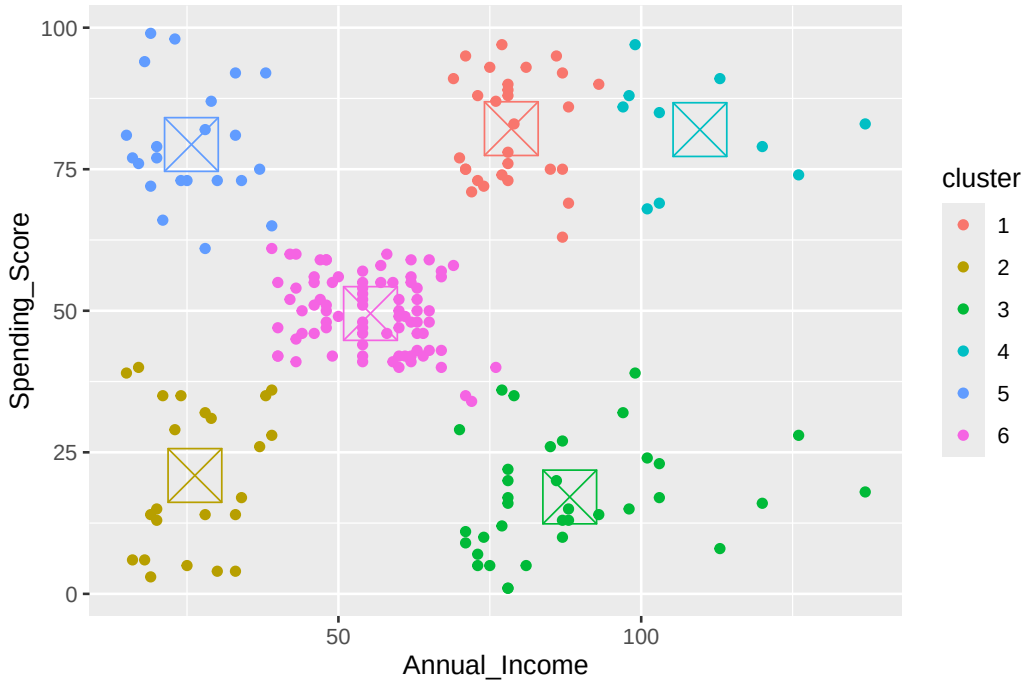
k-means Clustering in `mlr3`

- Higher Dunn index = Better clustering
  - **Compact:** Small variance between members
  - **Well separated:** Means of different clusters far apart from each other (as compared to the within cluster variance)

```
mdl_cv$archive %>%
  as.data.table() %>%
  select(centers, clust.dunn) %>%
  arrange(as.numeric(centers))
##    centers clust.dunn
##     <char>      <num>
## 1:       2 0.05212731
## 2:       3 0.14921873
## 3:       4 0.14527417
## 4:       5 0.16567293
## 5:       6 0.19283202
## 6:       7 0.13185604
## 7:       8 0.11390857
## 8:       9 0.11732031
## 9:      10 0.10450908

mdl_cv$tuning_result
##    centers learner_param_vals  x_domain clust.dunn
##     <char>             <list>    <list>      <num>
## 1:       6          <list[1]> <list[1]>   0.192832
```

k-means Clustering in `mlr3`

- For optimal $k = 6$:
  - The low versus high spending score clusters for low annual income are also successfully separated
  - In addition, the medium and high annual income clusters are separated only for high but not low spending score

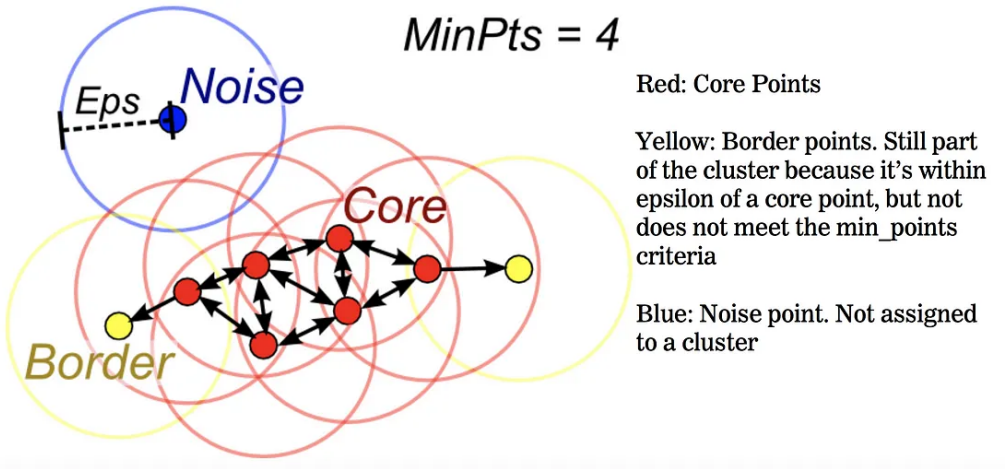


Excurse: Alternative Approaches

- k-means clustering is applicable only for continuous data
  - Uses the Euclidean distance as measure of dissimilarity
- More outlier robust: k-medians clustering
  - Uses the Manhattan distance as measure of dissimilarity and medians as cluster centroids
- For data including categorical variables: k-mediods
  - Any distance can be used to measure dissimilarity

Excurse: Anomaly Detection

- Clustering techniques can be used to detect anomalies in the data:

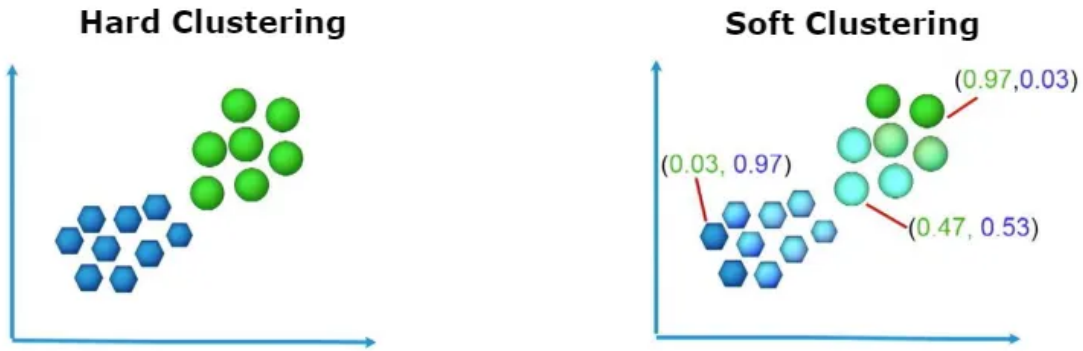


(https://towardsdatascience.com/best-clustering-algorithms-for-anomaly-detection-d5b7412537c8)

- Note: k-means is not suitable for anomaly detection, as it is only good when clusters are expected to have fairly regular shapes

Excurse: Fuzzy Clustering

- Fuzzy clustering: Allow data points to belong to **multiple** clusters to varying degrees
  - I.e., 0-100% membership in all available clusters
    - Cf. SVMs with soft margins: Allow for some classification uncertainty
  - E.g. ambiversion: Most people have both extroverted and introverted tendencies, rather than falling clearly into one category or the other
- Fuzzy c-means clustering: Assigning a vector of membership levels to each instance based on its distance to all available mean centroids



(https://medium.com/geekculture/fuzzy-c-means-clustering-fcm-algorithm-in-machine-learning-c2e51e586fff)

Excurse: Fuzzy Clustering

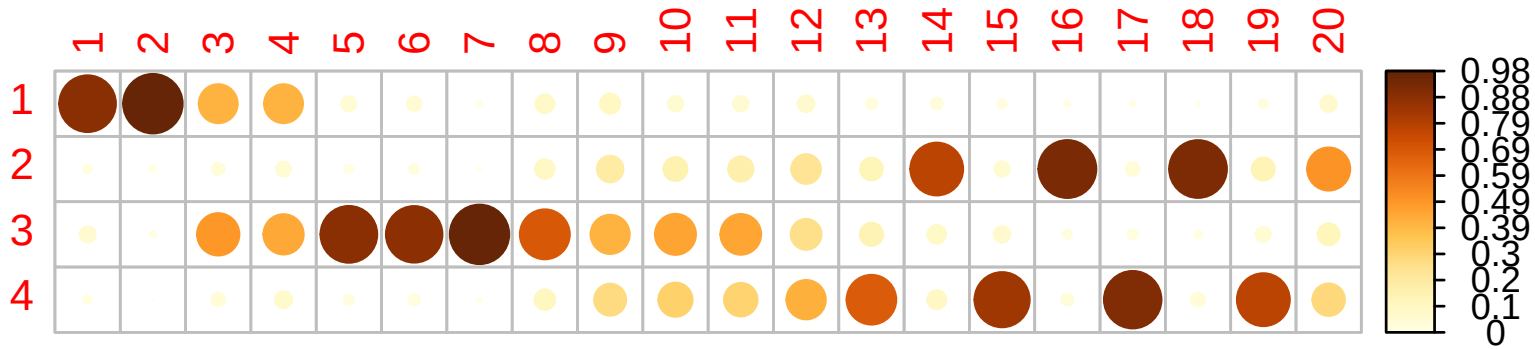
- Hyperparameter: The degree of fuzziness m of the clustering
  - Very large values of $m \Rightarrow$ All instances tend to belong to all clusters (i.e., blurred clustering)

```
tsk <- as_task_clust(dat_subset %>% select(-Gender))

mdl = lrn("clust.cmeans", centers = k, m = 2)
mdl$train(tsk)
mdl$model
## Fuzzy c-means clustering with 4 clusters
##
## Cluster centers:
##        Age Annual_Income CustomerID Spending_Score
## 1 34.02734      27.72076   28.45933       77.80776
## 2 34.58278      80.72659  147.22529       84.85540
## 3 55.29154      51.79518   77.07137       53.59007
## 4 43.79381      75.32525  140.24231       27.85364
##
## Memberships:
##                  1           2          3           4
##  [1,] 0.888235866 0.022667297 0.06898531 0.020111530
##  [2,] 0.980654153 0.003167280 0.01297717 0.003201394
##  [3,] 0.416587032 0.041222918 0.49283076 0.049359288
##  [4,] 0.416041938 0.059373950 0.44629867 0.078285444
##  [5,] 0.060993232 0.020596727 0.89084781 0.027562229
##  [6,] 0.056094434 0.025024580 0.88346145 0.035419533
```

Excurse: Fuzzy Clustering

- Visualization of the clustering:

```
# mlr3:
#autoplot(mdl$model)

# Other packages:
library(corrplot)
corrplot(t(mdl$model$membership), is.corr = FALSE)
```



Hands-on Practical Tutorial

- Now it's your turn:
  - Go to https://tobiasrebholz.github.io/teaching
  - Select "Introduction to Machine Learning" > "Materials"
    - Password: **smip24**
  - Download the "Clustering" tutorial
  - Work through the tasks

Summary

Summary

- Clustering:
  - **Hierarchical:** Appealing for users, because it's visual and shows a collection of clusterings
  - **Partitional:** Relies on the concept of prototypes (i.e., representatives of clusters)
- Hyperparameters in clustering analyses:
  - Number of clusters
  - But also: Distance between objects and linkage between sets (not discussed!)
- In order to visualize multivariate datasets, we can use dimensionality reduction techniques (see Appendix)
  - **Principal Component Analysis (PCA):** Projecting the data into a lower-dimensional space
    - For 2D (3D) visualizations, we can take the first two (three) PCs, capturing the largest proportion of total variance in the data

Appendix: Dimensionality Reduction

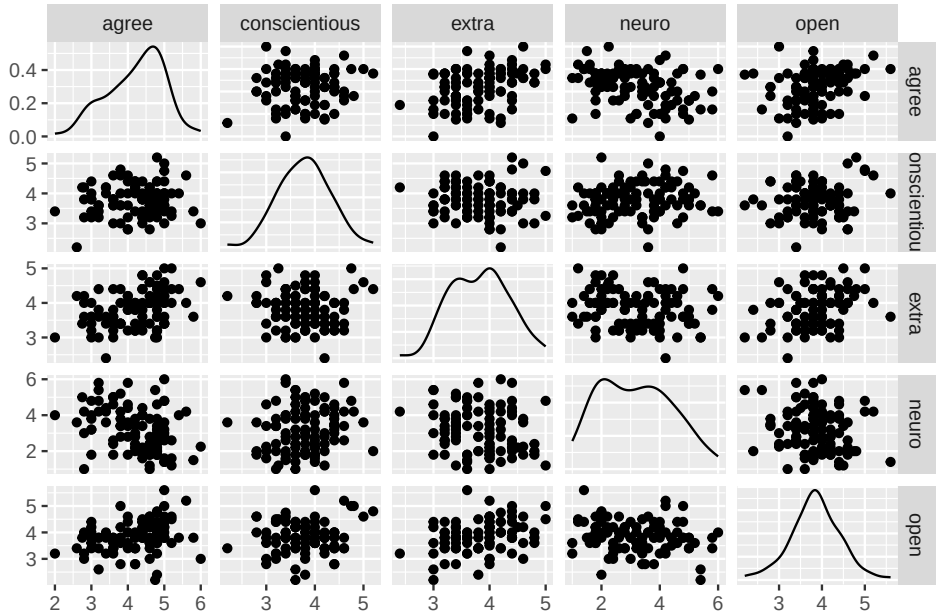
Visualizations

- 1D:
  - Histogram
  - Pie Charts
  - ...
- 2D:
  - xy-scatter plots
  - Bar charts
  - ...
- 3D:
  - xyz-scatter plots
  - Contour plots
  - ...
- $>$ 4D:
  - ???

Visualizations

- What if we have many (>4) dimensions?
  - E.g., pairwise plots:

```
dat <- read.csv('subfiles/data/bfi.csv', header = TRUE)

dat %>%
  select(agree, conscientious, extra, neuro, open) %>%
  ggpairs(upper = list(continuous = "points", combo = "facethist", discrete = "facetbar", r
```



Dimensionality Reduction

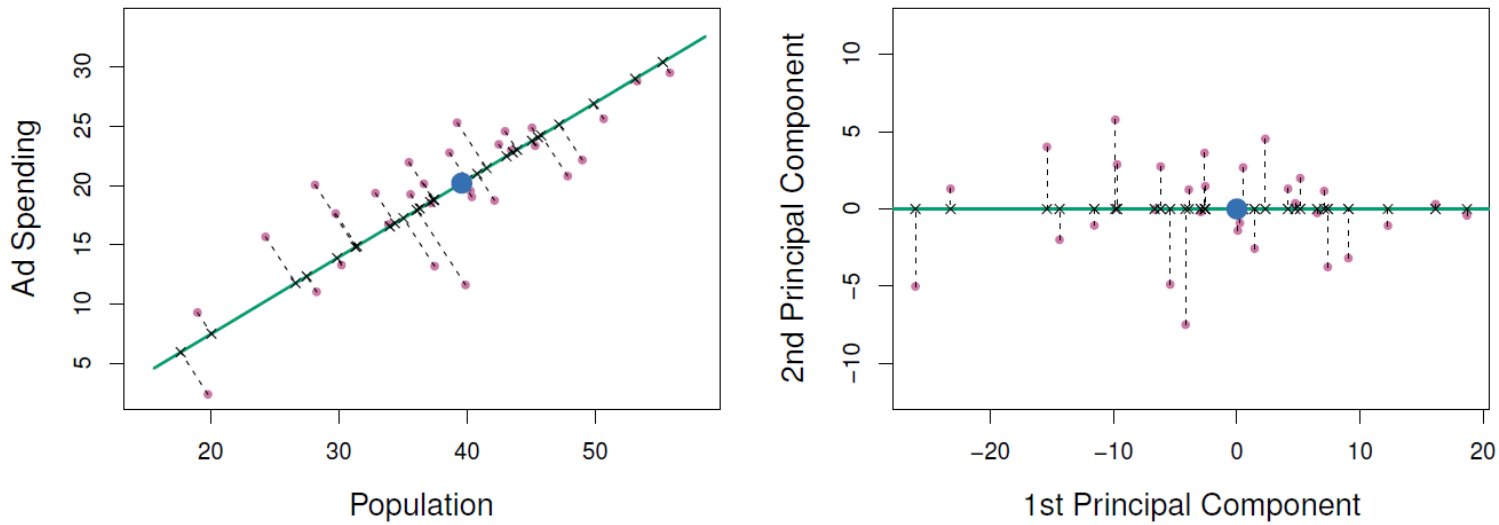
- Dimension of a dataset: $N \times p$
  - The number of instances: N
  - The number of features: p
- Dimension after reduction: $N \times l$, where $l < p$
  - Goal: Represent the most valuable information in the lower dimension l
- Why should we reduce dimensionality?
  - Visualization: If $l = 2$ or 3, we can plot in 2/3D
  - Computational costs: Computations are more affordable with $l < p$ features

Dimensionality Reduction Techniques

- Multidimensional Scaling
- Factorial Analysis
- Principal Components Analysis (our focus here!)
- ...

Principal Component Analysis

- In Principal Component Analysis (PCA), we look for the best linear representation of the feature space X in a lower dimension $l < p$
  - In the new space, the features are called "principal components"



(James et al., 2021, Figure 6.15)

Principal Component Analysis

- Total variance in X is equal to $\sum_{j=1}^{p} \sigma_j^2$
  - **First PC:** Explains the largest proportion of total variance in the data
  - **Second PC:** Explains the second largest proportion of total variance in the data
  - ...
  - p**-the PC:** Explains the smallest proportion of total variance in the data
- Any two PCs are uncorrelated
  - Reduced risk of multicollinearity if using the PCs instead of the original variables as features (e.g., in a regression model)
- There exists a strict hierarchy in the predictive value of all PCs
  - Using only a small subset of PCs as features: A reduced feature set that contains most of the signal (i.e., the first couple of PCs) might increase the predictive performance of our ML model because it is not "distracted" by other, noisy features

Principal Component Analysis in mlr3

```
df <- dat %>% select(agree, conscientious, extra, neuro, open)
head(df, 10)
##    agree conscientious extra neuro open
## 1    5.8           3.4   3.6   1.5  3.8
## 2    4.6           3.6   4.0   1.0  3.6
## 3    3.0           3.2   4.0   3.8  4.4
## 4    4.8           5.2   4.4   2.0  4.8
## 5    5.2           3.4   4.4   2.6  4.6
## 6    5.4           4.0   4.4   4.0  3.8
## 7    3.8           4.8   4.4   4.2  5.0
## 8    5.0           5.0   4.6   3.6  4.6
## 9    4.6           4.0   4.6   1.8  3.6
## 10   2.8           3.8   4.0   3.0  3.0

cor(as.matrix(df)) %>% round(., 2)
##                agree conscientious extra neuro  open
## agree           1.00          0.10  0.35 -0.35  0.30
## conscientious   0.10          1.00 -0.05  0.15  0.12
## extra           0.35         -0.05  1.00 -0.19  0.33
## neuro          -0.35          0.15 -0.19  1.00 -0.17
## open            0.30          0.12  0.33 -0.17  1.00
```

Principal Component Analysis in mlr3

- PCA is "only" a data preprocessing method (cf. z-standardization)
  - Therefore, we need to combine it with a task (e.g., using the PCs as features in a clustering task) in a pipeline (i.e., a predefined sequence of modeling steps)

```
tsk <- as_task_clust(df)

#combine pipe operation ("po()") and learner ("lrn()") into a pipeline:
mdl <- as_learner(po("pca", scale. = T) %>>% lrn('clust.agnes'))
mdl$train(tsk)

summary(mdl$model$pca)
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5
## Standard deviation     1.3631 1.0622 0.9080 0.8161 0.7234
## Proportion of Variance 0.3716 0.2256 0.1649 0.1332 0.1047
## Cumulative Proportion  0.3716 0.5972 0.7621 0.8953 1.0000
```

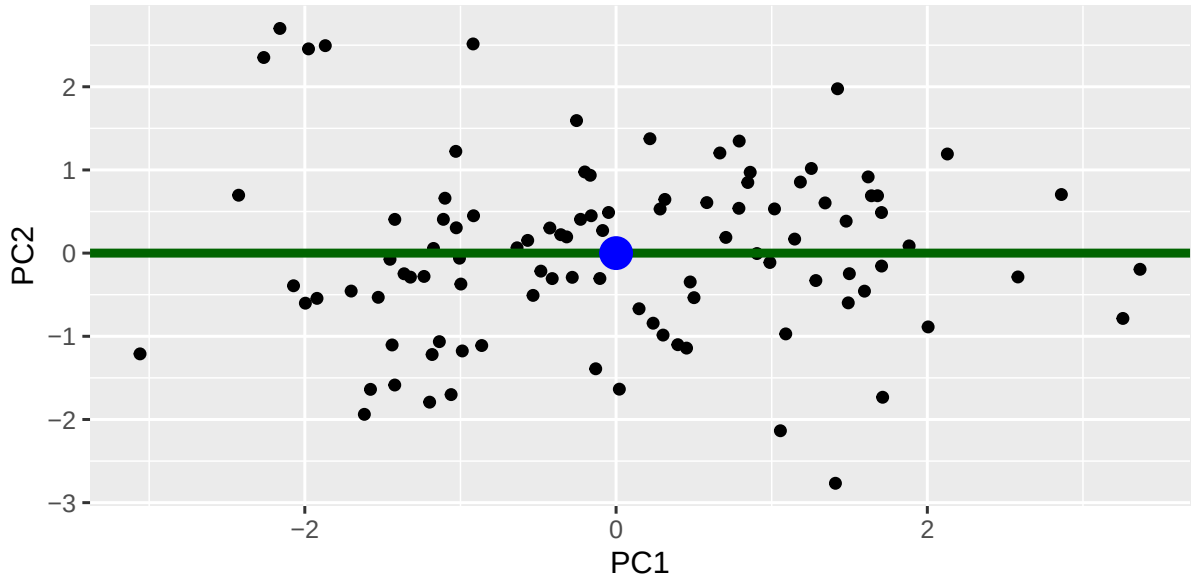
Principal Component Analysis in mlr3

- It is a bit complicated to access the output of pipeline operations in mlr3
  - Because they are only meant to be passed on to further modeling steps
  - But: We can manually calculate the PCs using the rotation from the model training:

```
X <- scale(as.matrix(df)) %*% mdl$model$pca$rotation #manually calculate the PCs using the

head(X)
##              PC1        PC2        PC3        PC4        PC5
## [1,] -1.4391681 -1.1041463  1.4377659  0.2242995 -1.1270828
## [2,] -0.9880684 -1.1761600  0.9940427  0.1826513  0.5969875
## [3,]  0.5008055 -0.5354200 -1.6978715 -1.1637931  0.3734138
## [4,] -2.2636213  2.3516355  0.6476358 -0.2932248  1.1614972
## [5,] -2.0725738 -0.3923845 -0.6305872 -0.1421267 -0.5807443
## [6,] -1.0994266  0.6600485 -0.3329577  1.3947503 -0.6462026

cor(as.matrix(X)) %>% round(., 2)
##     PC1 PC2 PC3 PC4 PC5
## PC1   1   0   0   0   0
## PC2   0   1   0   0   0
## PC3   0   0   1   0   0
## PC4   0   0   0   1   0
## PC5   0   0   0   0   1
```

Principal Component Analysis in mlr3

```
#autoplot(mdl$predict(tsk), type = 'pca')
ggplot(X, aes(x = PC1, y = PC2)) +
  geom_point() +
  geom_hline(yintercept = mean(X['PC1']),
             col = 'darkgreen', linewidth = 1.5) +
  geom_point(data = data.frame('PC1' = mean(X['PC1']), 'PC2' = mean(X['PC2'])),
             col = 'blue', size = 5)
```



Excurse: Scree Plots

- How many dimensions to choose?
  - Most reasonable: The smallest number of PCs that are required in order to explain a sizable amount of the variation in the data
  - Eyeballing the scree plot for an **"elbow"**: The point at which the proportion of variance explained by each subsequent PC drops off

```
ggplot(prop_var, aes(x = PC, y = prop_var)) +
  geom_line() + geom_point() +
  labs(y = "Prop. Variance Explained")
```