Introduction to Algorithmic Data Analysis

Esther Galbrun

Autumn 2023

UNIVERSITY OF EASTERN FINLAND

Last updated on 2023-07-04 10:51

Part II

Clustering Basics

Problem

A simple example

Consider a bunch of dry beans

A simple example

Consider a bunch of dry beans

We would like to divide them into a small number of groups, such that beans in a group are similar to each other and unlike beans from other groups

A simple example

Consider a bunch of dry beans

We would like to divide them into a small number of groups, such that beans in a group are similar to each other and unlike beans from other groups

A simple example

Consider a bunch of dry beans

We would like to divide them into a small number of groups, such that beans in a group are similar to each other and unlike beans from other groups

A simple example

Let us take a closer look at the beans...

Measurements, in number of pixels, can be extracted automatically from digital images of the beans

data points: Dry beans

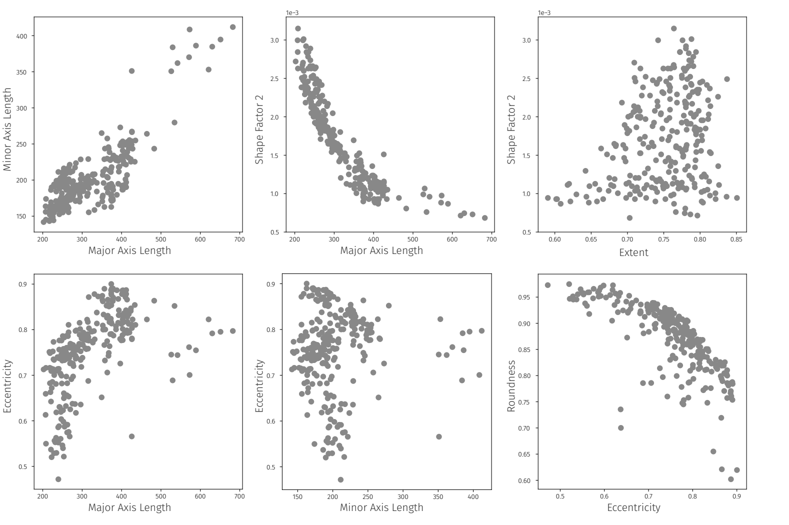
attributes: physical properties

- Major Axis Length
- Minor Axis Length
- Eccentricity
- Roundness
- Extent
- Shape Factor 2

See `https://doi.org/10.1016/j.compag.2020.105507` for details

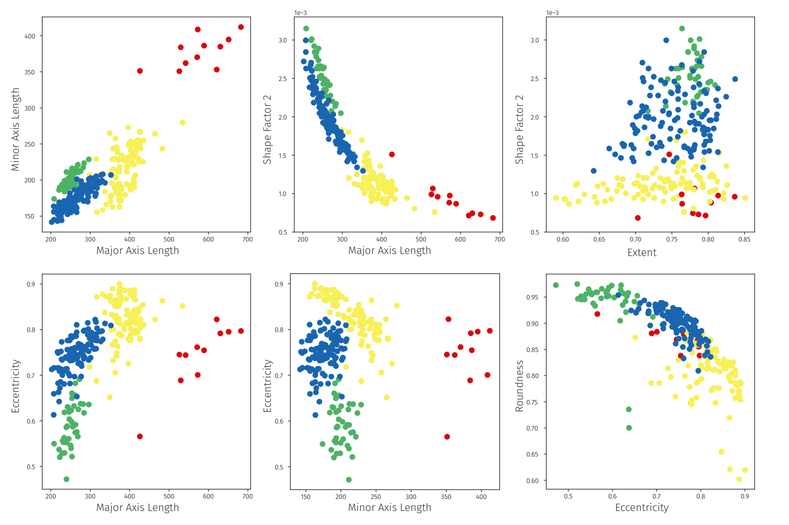
A simple example

Given measurements of physical properties of the beans...



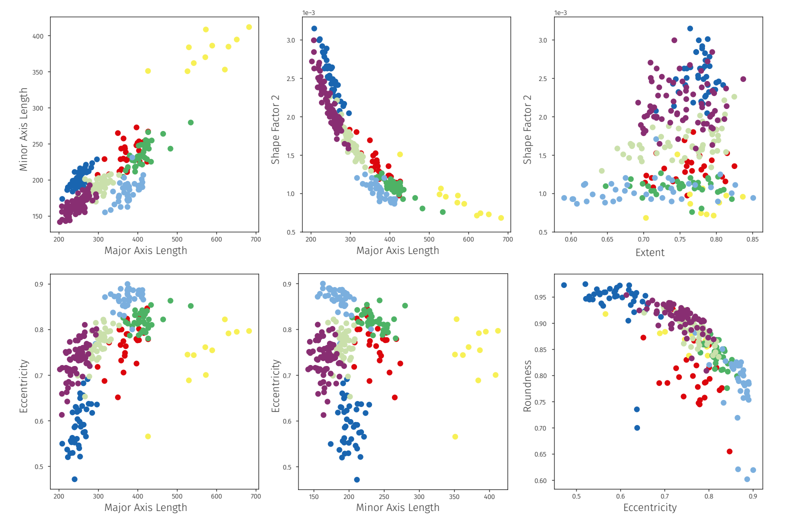
A simple example

...we would like to divide the beans into a few coherent groups



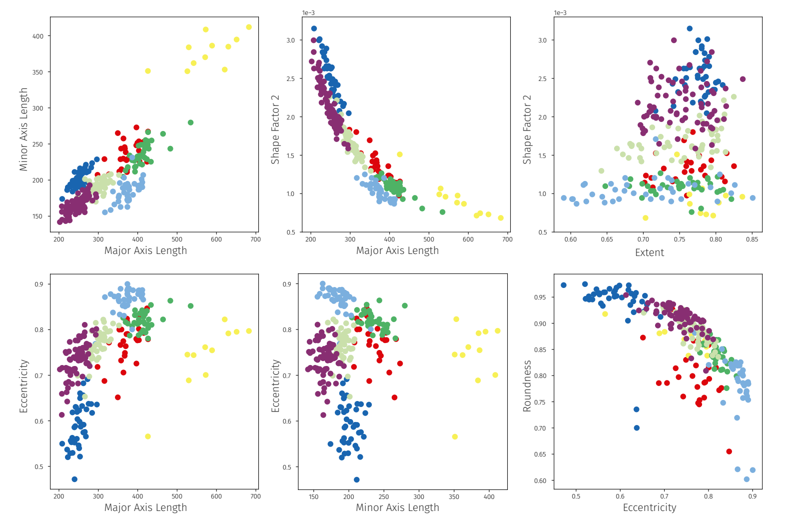
A simple example

…we would like to divide the beans into a few coherent groups



A simple example

→ This is a clustering task



The problem, informally

To put it simply,
the goal of clustering is to divide a collection of objects, or data points, into a small number of groups,
such that the objects *within a group* are *similar* to each other whereas objects *from different groups* are *dissimilar*

The problem, informally

Some obvious questions arise

How many groups?

How to measure similarity?

The problem, informally

Some obvious questions arise

How many groups?
: typically chosen by the user, i.e. input parameter, but determining the most appropriate number of groups can also be seen as part of the problem

How to measure similarity?
: calculating distances is a crucial ingredient in many clustering methods

The problem, informally

Some obvious questions arise

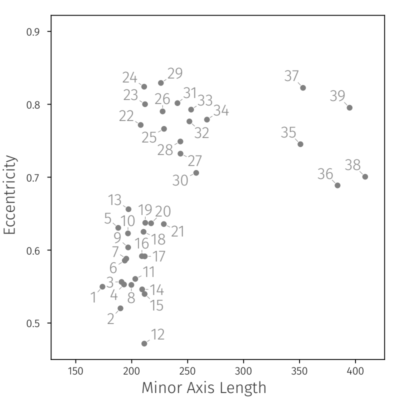
How many groups?
: typically chosen by the user, i.e. input parameter, but determining the most appropriate number of groups can also be seen as part of the problem

How to measure similarity?
: calculating distances is a crucial ingredient in many clustering methods

Next, we will look in turn at different clustering methods
Then, we will look at ways to compare and evaluate clusterings

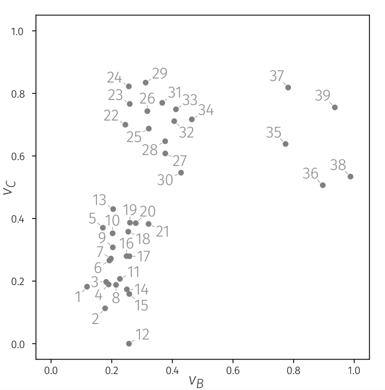
For illustrative purposes, we will take as an example a data set that consists of a handful of the beans

We focus on a pair of measurement variables at a time, so that we can easily visualize our data



A small example data set

- For illustrative purposes, we will take as an example a data set that consists of a handful of the beans
- We normalize the data, rescaling each variable so that its values fall within the unit interval



Some notations

The data set, denoted as $\mathcal{D}$, contains n data points and m attributes, i.e. it is a $n \times m$ matrix

A data point is a m-dimensional vector $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_m \rangle$
We denote $\boldsymbol{x}^{(j)}$ the j^{th} data point of $\mathcal{D}$, i.e. the j^{th} row
Data points are sometimes called *instances* or *examples*

We consider subsets of the data set
For instance, we denote the subset consisting of the first, third and fourth data points as $S = \{x_1, x_3, x_4\}$
For simplicity, when there is no ambiguity about the underlying data set, we might specify a subset by listing the indices of the data points it contains
The size of a subset S, denoted $|S|$, is the number of data points it contains

Some notations

A clustering is a collection of subsets of data points, called clusters

We write $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ to denote a clustering consisting of k clusters

The size of a clustering $\mathcal{C}$, denoted $|\mathcal{C}|$, is the number of clusters it contains

Typically, the clusters form a *partition* of the data set

That is, seeing $\mathcal{D}$ as a set of data points, i.e. ignoring the order, the clusters are such that

(i) they cover the entire data set, i.e. $\bigcup_{C \in \mathcal{C}} C = \mathcal{D}$, and

(ii) they are pairwise disjoint, i.e. $C_i \cap C_j = \emptyset$

for any pair of distinct clusters C_i and C_j from $\mathcal{C}$

Some notations

A clustering is a collection of subsets of data points, called clusters

We write $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ to denote a clustering consisting of k clusters

The size of a clustering $\mathcal{C}$, denoted $|\mathcal{C}|$, is the number of clusters it contains

Typically, the clusters form a *partition* of the data set

A clustering, i.e. an assignment of the data points to k clusters, can be represented as a n-dimensional vector

$$\boldsymbol{y} = \langle y_1, y_2, \ldots, y_n \rangle \in [1..k]^n ,$$

where y_j is the index of the cluster to which data point $\boldsymbol{x}^{(j)}$ is assigned. That is, $y_j = s$ if and only if $\boldsymbol{x}^{(j)} \in C_s$

Methods

Representative-based algorithms

A representative (a.k.a. center) is associated to each cluster

Representatives can be

- synthetic vectors from the domain, or
- existing points from the data set

Given a distance function d, the goal is to find a chosen number k of representatives so that all points are as close as possible from a representative

Find $R = \{\boldsymbol{r}^{(1)}, \boldsymbol{r}^{(2)}, \ldots \boldsymbol{r}^{(k)}\}$ to minimize $\sum_{x \in \mathcal{D}} \min_{r \in R} \mathrm{d}(x, \boldsymbol{r})$

Each data point is assigned to the cluster associated to its closest representative

The representatives and the assignment of data points are unknown a priori, but depend on each other in a circular way

- if the representatives are fixed, it is easy to assign each data point to the closest one
- if the assignment of data points is fixed, it is easy to determine a representative for each group

Such problems are typically solved using an iterative algorithm that alternates between refining the representatives or the assignment, keeping the other one fixed

k-means algorithm

When the chosen distance function is the Euclidean distance (ℓ_2 norm), i.e.

$$\mathrm{d}(x, x') = \sqrt{\sum_{i=1}^{i=m} (x_i - x'_i)^2},$$

it can be shown that the optimal representative is the mean of the data points assigned to it

k-means algorithm

When the chosen distance function is the Euclidean distance (ℓ_2 norm), i.e.

$$d(x, x') = \sqrt{\sum_{i=1}^{i=m} (x_i - x'_i)^2}\,,$$

it can be shown that the optimal representative is the mean of the data points assigned to it

Considering the data points assigned to cluster C_j,
the associated representative is $\boldsymbol{r}^{(j)} = \langle r_1^{(j)}, r_2^{(j)}, \ldots, r_m^{(j)} \rangle$
where

$$r_i^{(j)} = \frac{\sum_{\boldsymbol{x} \in C_j} x_i}{|C_j|}$$

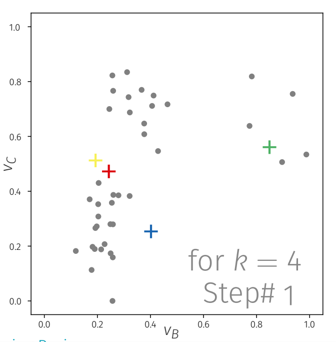
centers $\leftarrow$ k points sampled from the domain

repeat

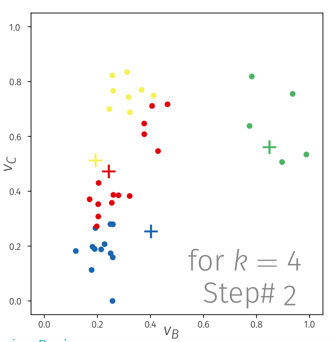
Assign points to closest center

centers $\leftarrow$ **mean** of assigned points

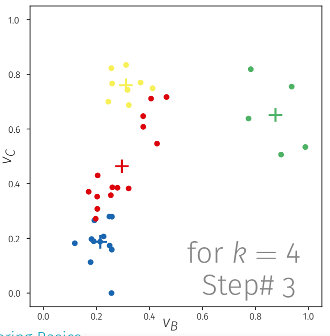
until convergence



centers $\leftarrow$ k points sampled from the domain
repeat
    Assign points to closest center
    centers $\leftarrow$ **mean** of assigned points
until convergence

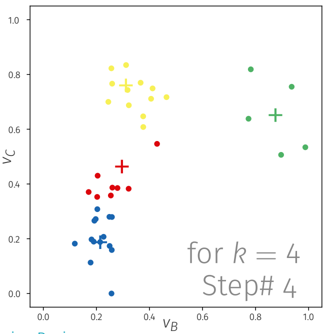


centers $\leftarrow$ k points sampled from the domain
repeat
  Assign points to closest center
  centers $\leftarrow$ **mean** of assigned points
until convergence



k-means algorithm

centers $\leftarrow$ k points sampled from the domain
repeat
    Assign points to closest center
    centers $\leftarrow$ **mean** of assigned points
until convergence



k-means algorithm

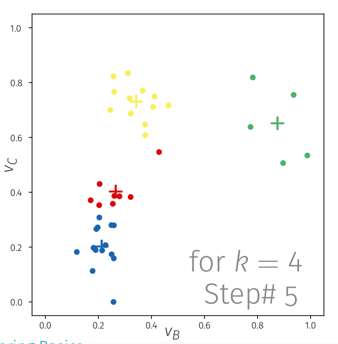
centers ← k points sampled from the domain

repeat

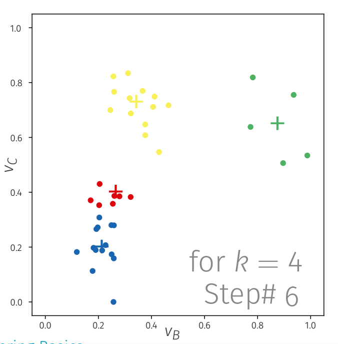
Assign points to closest center

centers ← **mean** of assigned points

until convergence

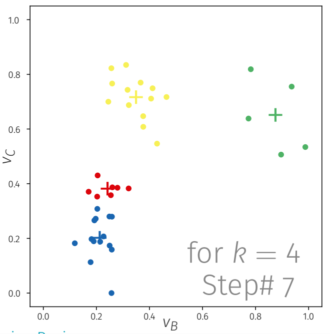


centers ← k points sampled from the domain
repeat
  Assign points to closest center
  centers ← **mean** of assigned points
until convergence



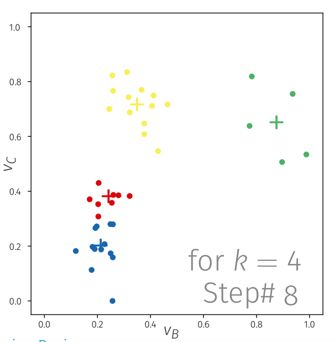
k-means algorithm

centers $\leftarrow$ k points sampled from the domain
repeat
  Assign points to closest center
  centers $\leftarrow$ **mean** of assigned points
until convergence



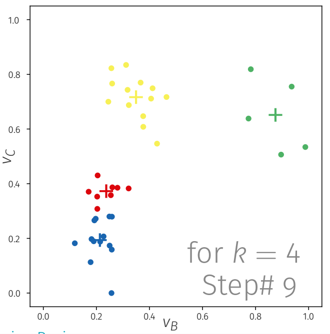
k-means algorithm

centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← **mean** of assigned points
until convergence



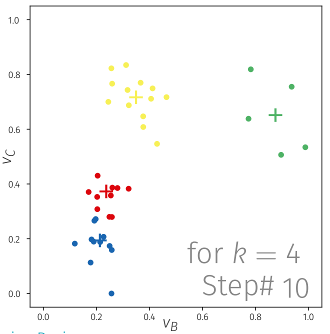
k-means algorithm

centers $\leftarrow k$ points sampled from the domain
repeat
    Assign points to closest center
    centers $\leftarrow$ **mean** of assigned points
until convergence

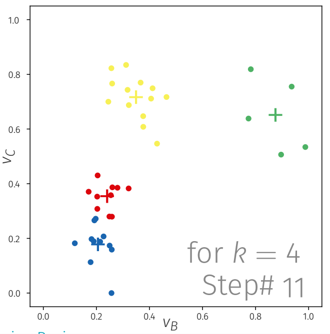


k-means algorithm

centers ← k points sampled from the domain
repeat
  Assign points to closest center
  centers ← **mean** of assigned points
until convergence

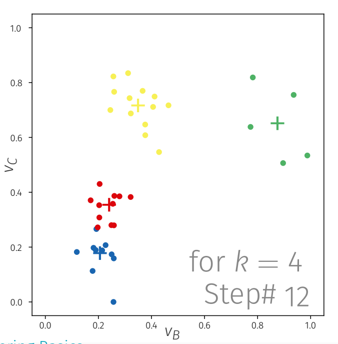


centers $\leftarrow$ k points sampled from the domain
repeat
Assign points to closest center
centers $\leftarrow$ **mean** of assigned points
until convergence

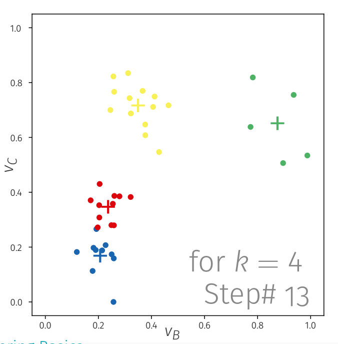


k-means algorithm

centers $\leftarrow$ k points sampled from the domain
repeat
  Assign points to closest center
  centers $\leftarrow$ **mean** of assigned points
until convergence

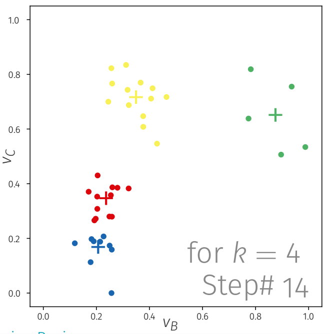


centers $\leftarrow$ k points sampled from the domain
repeat
  Assign points to closest center
  centers $\leftarrow$ **mean** of assigned points
until convergence

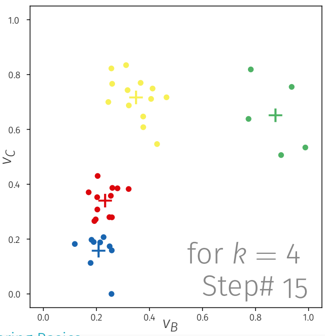


k-means algorithm

centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← **mean** of assigned points
until convergence



centers ← k points sampled from the domain
repeat
  Assign points to closest center
  centers ← **mean** of assigned points
until convergence



k-means algorithm

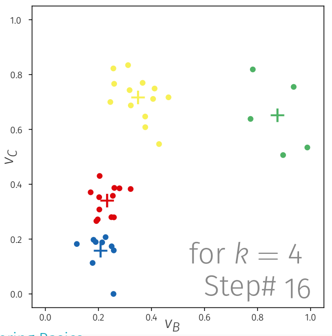
centers $\leftarrow$ k points sampled from the domain

repeat

Assign points to closest center

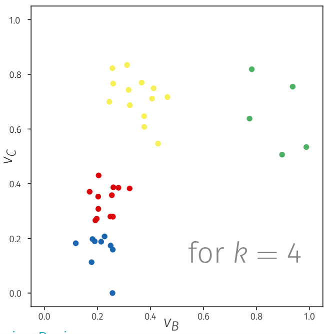
centers $\leftarrow$ **mean** of assigned points

until convergence



k-means algorithm

centers $\leftarrow$ k points sampled from the domain
repeat
  Assign points to closest center
  centers $\leftarrow$ **mean** of assigned points
until convergence



k-medians algorithm

When the chosen distance function is the Manhattan distance (ℓ_1 norm), i.e.

$$d(x, x') = \sum_{i=1}^{i=m} \left| x_i - x'_i \right| ,$$

it can be shown that the optimal representative is the median of the data points assigned to it

k-medians algorithm

When the chosen distance function is the Manhattan distance (ℓ_1 norm), i.e.

$$\mathrm{d}(x, x') = \sum_{i=1}^{i=m} \left| x_i - x'_i \right| ,$$

it can be shown that the optimal representative is the median of the data points assigned to it

Given a set of values A,
we denote $a^{<p>}$ the p^{th} smallest value in A,
that is, $a^{<1>} \leq a^{<2>} \leq \cdots \leq a^{<|A|>}$

Then the median of A is

$$\mathrm{median}(A) = \begin{cases} a^{<(|A|+1)/2>} & \text{if } |A| \text{ is odd} \\ (a^{<|A|/2>} + a^{<|A|/2+1>})/2 & \text{if } |A| \text{ is even} \end{cases}$$

k-medians algorithm

When the chosen distance function is the Manhattan distance (ℓ_1 norm), i.e.

$$\mathrm{d}(x, x') = \sum_{i=1}^{i=m} \left| x_i - x_i' \right| ,$$

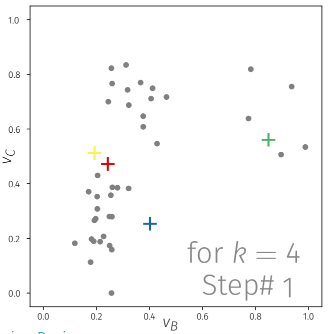
it can be shown that the optimal representative is the median of the data points assigned to it

Considering the data points assigned to cluster C_j,
the associated representative is $\boldsymbol{r}^{(j)} = \langle r_1^{(j)}, r_2^{(j)}, \ldots, r_m^{(j)} \rangle$
where

$$r_i^{(j)} = \mathrm{median}(\{x_i,\ \boldsymbol{x} \in C_j\})$$

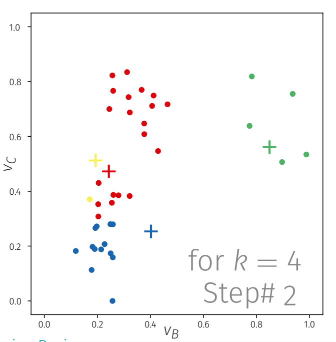
k-medians algorithm

centers ← k points sampled from the domain
repeat
  Assign points to closest center
  centers ← **median** of assigned points
until convergence



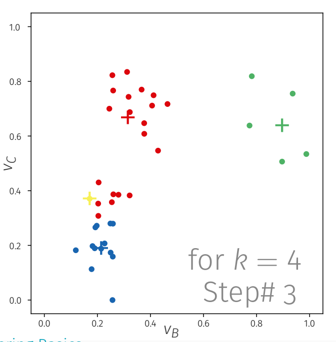
k-medians algorithm

centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← **median** of assigned points
until convergence

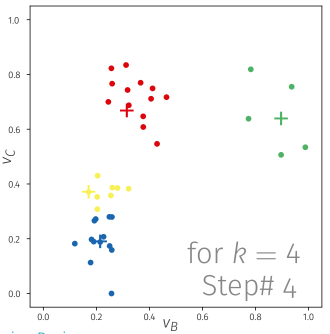


k-medians algorithm

centers ← k points sampled from the domain
repeat
Assign points to closest center
centers ← **median** of assigned points
until convergence

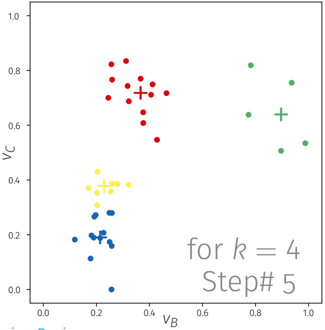


centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← **median** of assigned points
until convergence

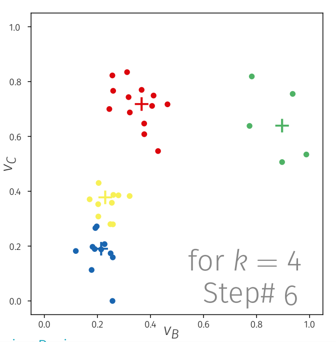


k-medians algorithm

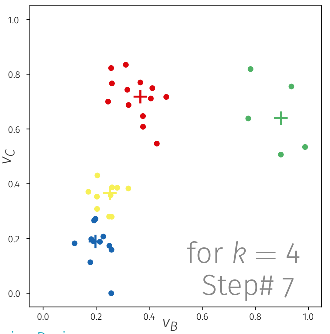
centers ← k points sampled from the domain
repeat
  Assign points to closest center
  centers ← **median** of assigned points
until convergence



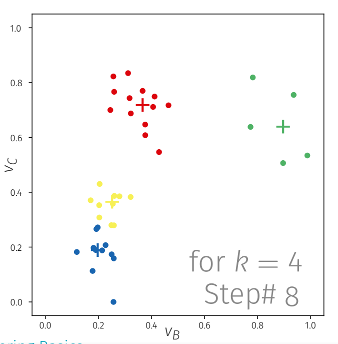
centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← **median** of assigned points
until convergence



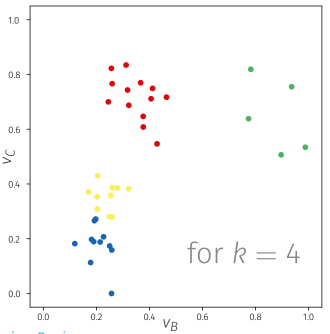
centers ← k points sampled from the domain
repeat
  Assign points to closest center
  centers ← **median** of assigned points
until convergence



centers $\leftarrow$ k points sampled from the domain
repeat
    Assign points to closest center
    centers $\leftarrow$ **median** of assigned points
until convergence



centers ← k points sampled from the domain
repeat
    Assign points to closest center
    centers ← **median** of assigned points
until convergence

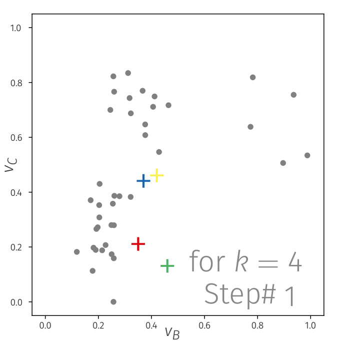


Representative-based algorithms

	Euclidean distance	Manhattan distance
Distance	ℓ_2 norm	ℓ_1 norm
$\mathrm{d}(x, x')$	$\sqrt{\sum_{i=1}^{i=m} (x_i - x'_i)^2}$	$\sum_{i=1}^{i=m} \lvert x_i - x'_i \rvert$
Representative	mean	median
$r_i^{(j)}$	$\sum_{\mathbf{x} \in C_j} x_i / \lvert C_j \rvert$	$\mathrm{median}(\{x_i,\ \mathbf{x} \in C_j\})$
Algorithm	k-means	k-medians

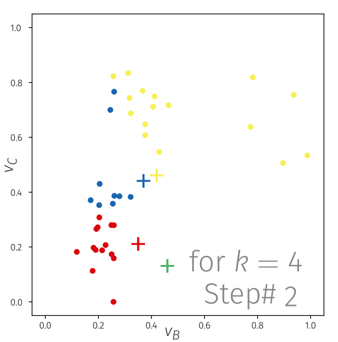
Initialization

A representative might not be assigned any data point, because it is not closest to any one



Initialization

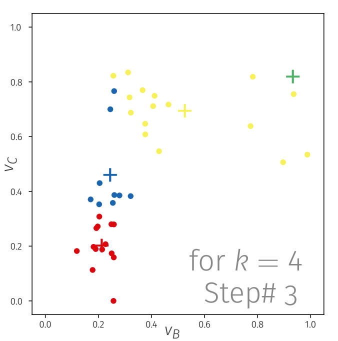
A representative might not be assigned any data point, because it is not closest to any one



Initialization

A representative might not be assigned any data point, because it is not closest to any one

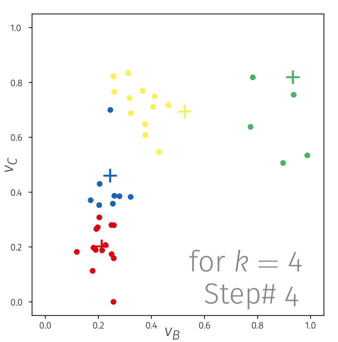
In such cases, the representative might be dropped altogether, resulting in a smaller number of clusters, or re-initialised



Initialization

A representative might not be assigned any data point, because it is not closest to any one

In such cases, the representative might be dropped altogether, resulting in a smaller number of clusters, or re-initialised



Initialization

A representative might not be assigned any data point, because it is not closest to any one

Initialization

The initialization of the representatives is a crucial step

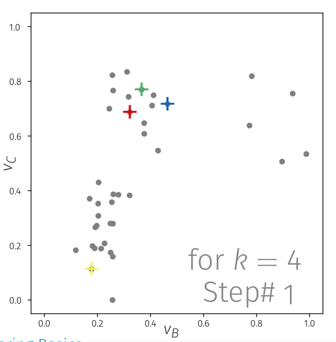
Instead of initializing the representatives as random vectors from the domain of the variables, one might sample existing points from the data set centers $\leftarrow$ k points sampled from the data

repeat

  Assign points to closest center

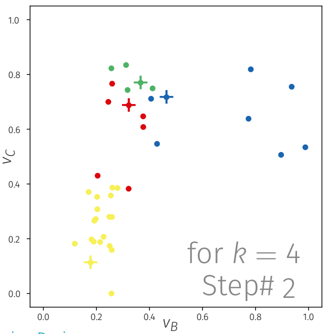
  centers $\leftarrow$ **mean** of assigned points

until convergence



k-means algorithm

centers $\leftarrow$ k points sampled from the data
repeat
    Assign points to closest center
    centers $\leftarrow$ **mean** of assigned points
until convergence



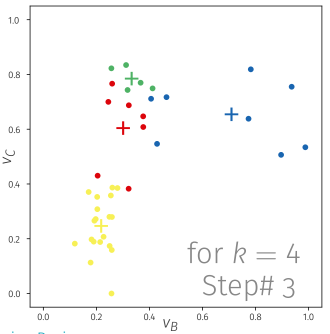
centers $\leftarrow k$ points sampled from the data

repeat

Assign points to closest center

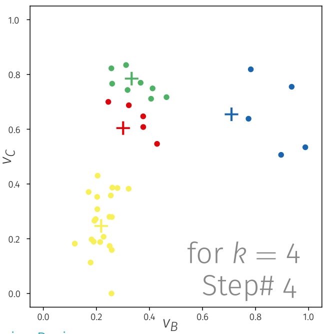
centers $\leftarrow$ **mean** of assigned points

until convergence



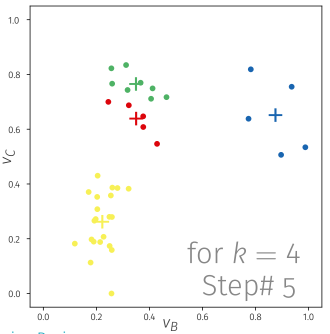
k-means algorithm

centers ← k points sampled from the data
repeat
  Assign points to closest center
  centers ← **mean** of assigned points
until convergence



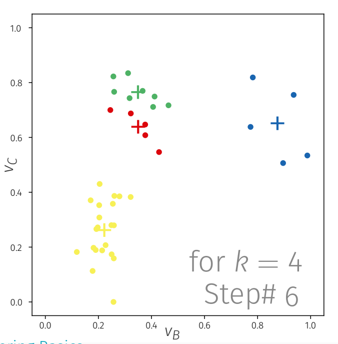
k-means algorithm

centers $\leftarrow$ k points sampled from the data
repeat
    Assign points to closest center
    centers $\leftarrow$ **mean** of assigned points
until convergence



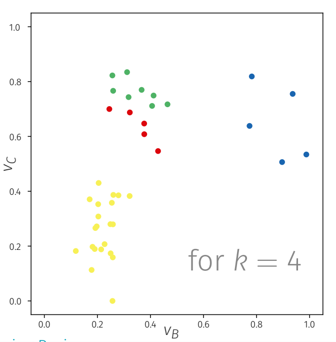
k-means algorithm

centers $\leftarrow k$ points sampled from the data
repeat
  Assign points to closest center
  centers $\leftarrow$ **mean** of assigned points
until convergence



k-means algorithm

centers $\leftarrow$ k points sampled from the data
repeat
    Assign points to closest center
    centers $\leftarrow$ **mean** of assigned points
until convergence



Initialization

The initialization of the representatives is a crucial step

It is desirable that the initial representatives are spread through different regions of the data

When sampling, penalize data points that are close to previously selected representatives

Initialization

The initialization of the representatives is a crucial step

It is desirable that the initial representatives are spread through different regions of the data

When sampling, penalize data points that are close to previously selected representatives

$\rightarrow$ k-means++ algorithm

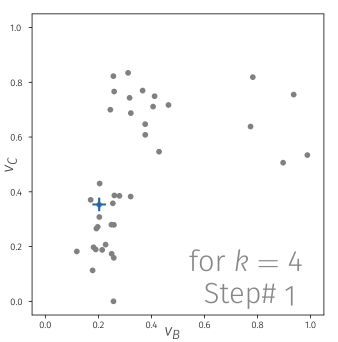
k-means++ algorithm

$r^{(1)} \leftarrow$ sample from data points uniformly at random

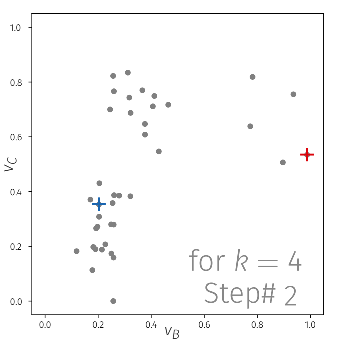
for $j = 2..k$ **do**

$r^{(j)} \leftarrow$ sample from remaining data points with probability $P(x) \propto \min_{i=1..j} \mathrm{d}(x, r_i)$

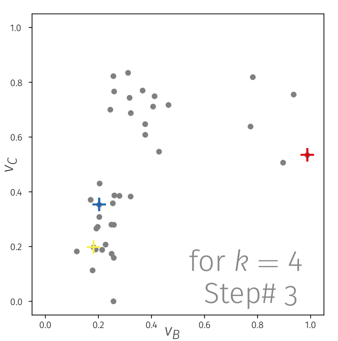
...



$r^{(1)} \leftarrow$ sample from data points uniformly at random
for $j = 2..k$ **do**
$\quad r^{(j)} \leftarrow$ sample from remaining data points with probability $P(x) \propto \min_{i=1..j} d(x, r_i)$
...



$r^{(1)} \leftarrow$ sample from data points uniformly at random
for $j = 2..k$ **do**
$r^{(j)} \leftarrow$ sample from remaining data points with probability $P(\mathbf{x}) \propto \min_{i=1..j} \mathrm{d}(\mathbf{x}, \mathbf{r}_i)$
...

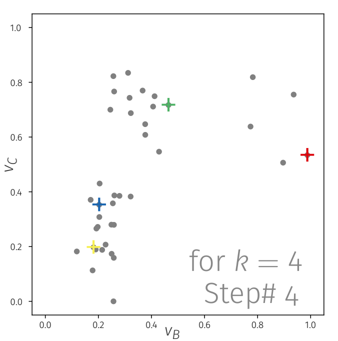


$r^{(1)} \leftarrow$ sample from data points uniformly at random

for $j = 2..k$ **do**

$r^{(j)} \leftarrow$ sample from remaining data points with probability $P(x) \propto \min_{i=1..j} d(x, r_i)$

...



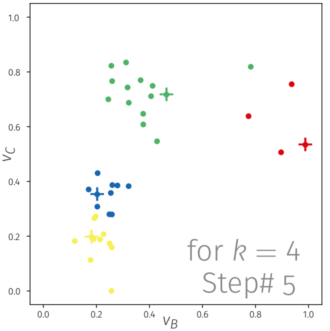
...

repeat

Assign points to closest center

centers ← **mean** of assigned points

until convergence



k-means++ algorithm

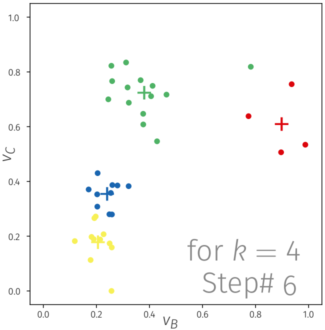
...

repeat

Assign points to closest center

centers ← **mean** of assigned points

until convergence



k-means++ algorithm

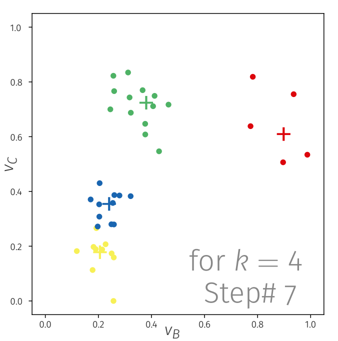
...

repeat

Assign points to closest center

centers ← **mean** of assigned points

until convergence



k-means++ algorithm

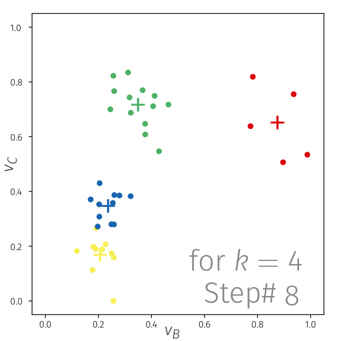
...

repeat

Assign points to closest center

centers ← **mean** of assigned points

until convergence



k-means++ algorithm

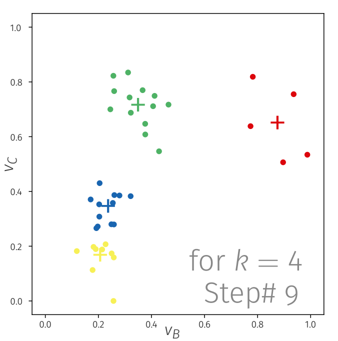
...

repeat

Assign points to closest center

centers $\leftarrow$ **mean** of assigned points

until convergence



k-means++ algorithm

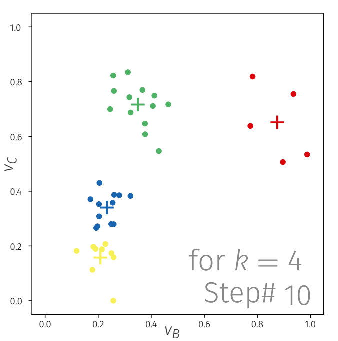
...

repeat

Assign points to closest center

centers $\leftarrow$ **mean** of assigned points

until convergence



k-means++ algorithm

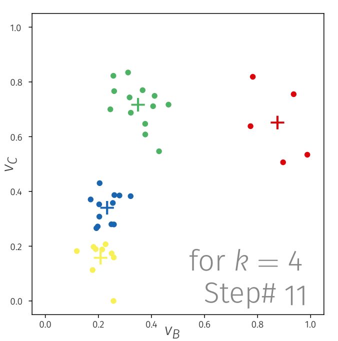
...

repeat

Assign points to closest center

centers $\leftarrow$ **mean** of assigned points

until convergence



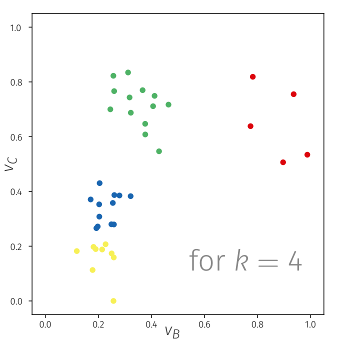
...

repeat

Assign points to closest center

centers ← **mean** of assigned points

until convergence



Initialization

The initialization of the representatives is a crucial step

More careful initialization, as with k-means++ might be more expensive, but the algorithm then typically and more reliably converges in fewer iterations, and to better solutions

Probabilistic model-based algorithms

Underlying assumption: the data was generated from a *mixture* of k probability distributions

A probabilistic model (a.k.a. *mixture component*) is associated to each cluster

Each data point is generated by the mixture model as follows
(i) a mixture component is selected
(ii) the data point is generated from this component

Probabilistic model-based algorithms

Each data point is generated by the mixture model as follows
(i) a mixture component is selected
(ii) the data point is generated from this component

Each mixture component $M^{(j)}$ has a prior probability α_j, a set of parameters θ_j and a probability density function f_{θ_j}

For a probabilistic model $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(k)}\}$, the probability of data point $\boldsymbol{x}$ is

$$p(\boldsymbol{x} \mid \mathcal{M}) = \sum_{j=1}^{j=k} \alpha_j \cdot f_{\theta_j}(\boldsymbol{x})$$

and the probability of the data is

$$p(\mathcal{D} \mid \mathcal{M}) = \prod_{\boldsymbol{x} \in \mathcal{D}} p(\boldsymbol{x} \mid \mathcal{M})$$

Probabilistic model-based algorithms

Each mixture component $M^{(j)}$ has a prior probability α_j, a set of parameters θ_j and a probability density function f_{θ_j}

For a probabilistic model $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(k)}\}$, the probability of the data is

$$p(\mathcal{D} \mid \mathcal{M}) = \prod_{x \in \mathcal{D}} p(x \mid \mathcal{M})$$

It is generally more convenient to work with the log likelihood

$$\mathcal{L}(\mathcal{D} \mid \mathcal{M}) = \log\left(p(\mathcal{D} \mid \mathcal{M})\right) = \sum_{x \in \mathcal{D}} \log \sum_{j=1}^{j=k} \alpha_j \cdot f_{\theta_j}(x)$$

Probabilistic model-based algorithms

Each mixture component $M^{(j)}$ has a prior probability α_j, a set of parameters θ_j and a probability density function f_{θ_j}

For a probabilistic model $\mathcal{M} = \{M^{(1)}, M^{(2)}, \ldots, M^{(k)}\}$, the log likelihood of the data is

$$\mathcal{L}(\mathcal{D} \mid \mathcal{M}) = \sum_{x \in \mathcal{D}} \log \sum_{j=1}^{j=k} \alpha_j \cdot f_{\theta_j}(x)$$

The goal is to find the model parameters that maximize the fit of the model to the data, as measured by the log likelihood

$$\arg\max_{\mathcal{M}} \mathcal{L}(\mathcal{D} \mid \mathcal{M})$$

Probabilistic model-based algorithms

The optimal parameters of the model and the probabilities of data points are unknown a priori, but depend on each other in a circular way

- if the parameters of the model are fixed, it is easy to compute the probabilities of each data point to be generated by each mixture component
- if the probabilities of each data point to be generated by each mixture component are fixed, it is relatively easy to determine the parameters of the model

Probabilistic model-based algorithms

The optimal parameters of the model and the probabilities of data points are unknown a priori, but depend on each other in a circular way

Such problems are typically solved using an iterative algorithm that alternates between two steps

Expectation estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$

Maximization estimate model parameters Θ to maximize the log-likelihood fit under the current assignment

Probabilistic model-based algorithms

The optimal parameters of the model and the probabilities of data points are unknown a priori, but depend on each other in a circular way

Such problems are typically solved using an iterative algorithm that alternates between two steps

Expectation estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$

Maximization estimate model parameters Θ to maximize the log-likelihood fit under the current assignment

→ Expectation-Maximization (or EM) algorithm

Gaussian mixture model

In particular, Gaussian distributions might be used as mixture components

A Gaussian distribution (a.k.a. normal distribution) has two parameters, the *mean* $\boldsymbol{\mu}$ and the *variance* $\boldsymbol{\sigma}^2$, i.e. $\theta = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, and is typically denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$

In particular, Gaussian distributions might be used as mixture components

A Gaussian distribution (a.k.a. normal distribution) has two parameters, the *mean* $\boldsymbol{\mu}$ and the *variance* $\boldsymbol{\sigma}^2$, i.e. $\theta = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, and is typically denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$

The probability of data point $\boldsymbol{x}$ under mixture component M with $\boldsymbol{\mu} = \langle \mu_1, \ldots, \mu_m \rangle$ and $\boldsymbol{\sigma}^2 = \langle \sigma_1^2, \ldots, \sigma_m^2 \rangle$ is

$$f_\theta(\boldsymbol{x}) = \prod_{i=1}^{i=m} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x_i - \mu_i)^2}{2\sigma_i^2}}$$

Gaussian mixture model

Let Θ denote all parameters of the model, including the mean and variance of each mixture component, as well as their prior probabilities

Assuming that the parameter values in Θ are fixed, the posterior probability that data point x was generated by mixture component $M^{(j)}$ is

$$p(M^{(j)} \mid x, \Theta) = \frac{\alpha_j \cdot f_{\theta_j}(x)}{\sum_{s=1}^{s=k} \alpha_s \cdot f_{\theta_s}(x)}$$

This can be interpreted as a *soft assignment* of the data point to the mixture component, and used to weigh the contribution of the data point to the mixture component

Gaussian mixture model

Given a soft assignment of data points to mixture component $M^{(j)}$ as a vector of probabilities over the data points $\mathbf{w}^{(j)} \in [0, 1]^n$, such that $w_q^{(j)} = p(M^{(j)} \mid \mathbf{x}^{(q)}, \Theta)$, the parameters of the mixture component are estimated as

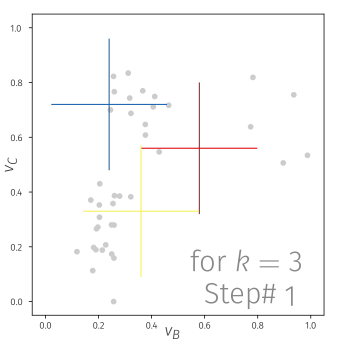
$$\mu_i^{(j)} = \frac{\sum_{q=1}^{q=n} w_q^{(j)} \cdot x_i^{(q)}}{\sum_{q=1}^{q=n} w_q} \quad \text{and} \quad \sigma_i^{(j)} = \sqrt{\frac{\sum_{q=1}^{q=n} w_q^{(j)} \cdot (x_i^{(q)} - \mu_i)^2}{\sum_{q=1}^{q=n} w_q}}$$

And the prior probability of mixture component $M^{(j)}$ is estimated as

$$\alpha_j = \frac{\sum_{q=1}^{q=n} w_q^{(j)}}{n}$$

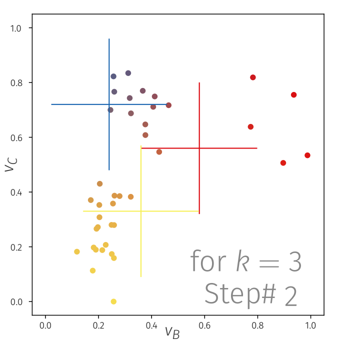
EM clustering with Gaussian mixture model

Initialize mixture components: sample k normal distribution parameter pairs and set prior probabilities



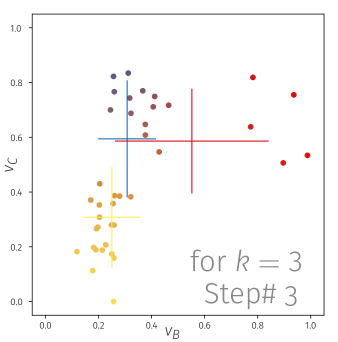
$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
    compute points assignment
    compute parameters Θ
until convergence

E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
    compute points assignment
    compute parameters Θ
until convergence

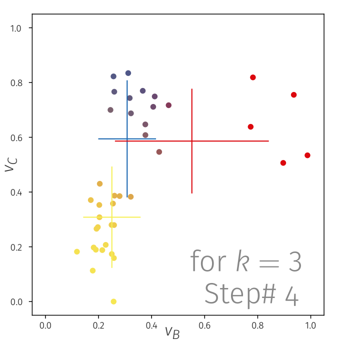
M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
 compute points assignment
 compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$

$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$

repeat

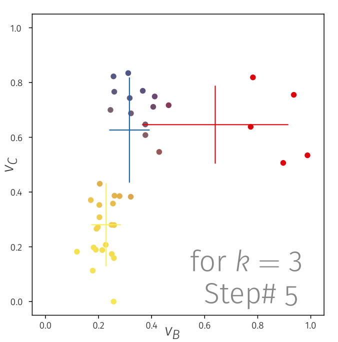
  compute points assignment

  compute parameters Θ

until convergence

EM clustering with Gaussian mixture model

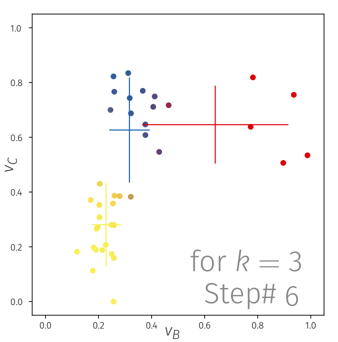
M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
    compute points assignment
    compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$

$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$

repeat

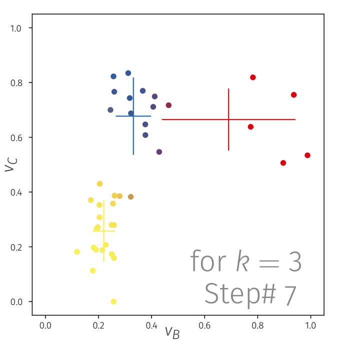
compute points assignment

compute parameters Θ

until convergence

EM clustering with Gaussian mixture model

M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$

$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$

repeat

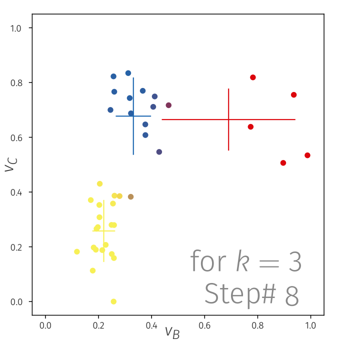
compute points assignment

compute parameters Θ

until convergence

EM clustering with Gaussian mixture model

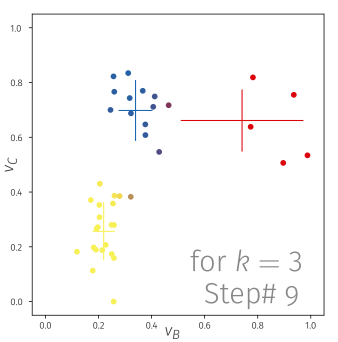
E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
  compute points assignment
  compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

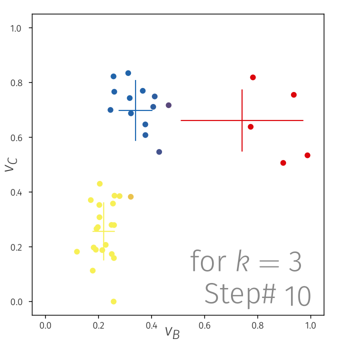
M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
compute points assignment
compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$

$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$

repeat

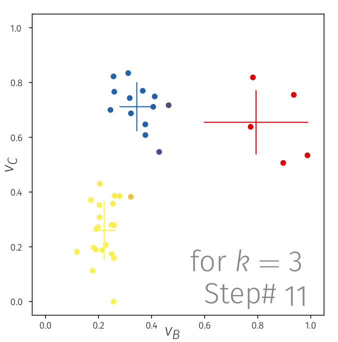
  compute points assignment

  compute parameters Θ

until convergence

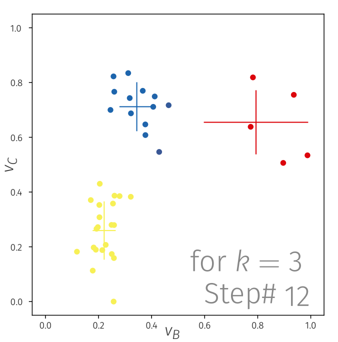
EM clustering with Gaussian mixture model

M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
    compute points assignment
    compute parameters Θ
until convergence

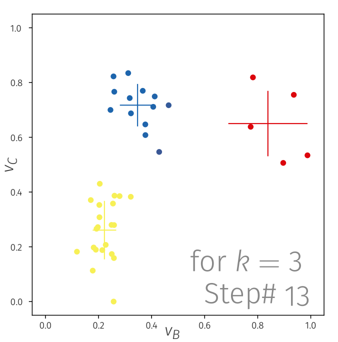
E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
    compute points assignment
    compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

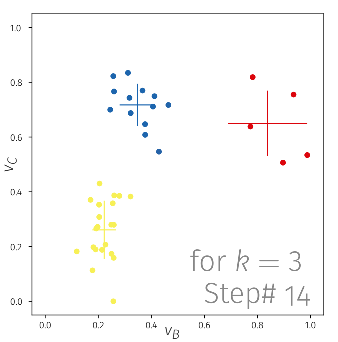
M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
    compute points assignment
    compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

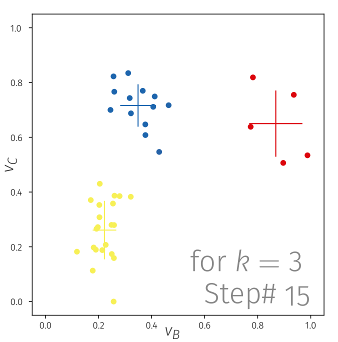
E(xpectation) step: estimate the posterior probability that point x was generated by each mixture component $M^{(j)}$



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
  compute points assignment
  compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

M(aximization) step: estimate model parameters Θ to maximize the log-likelihood fit under the current assignment



$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$

$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$

repeat

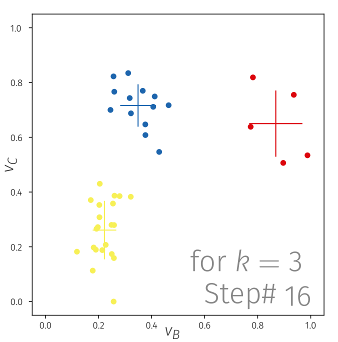
  compute points assignment

  compute parameters Θ

until convergence

EM clustering with Gaussian mixture model

Convergence: the parameters of the model stabilize

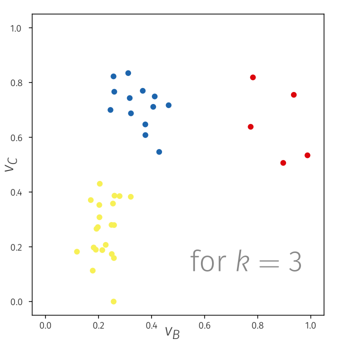


$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
  compute points assignment
  compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

Soft assignment: compute the final posterior probabilities for each data point $\boldsymbol{x}$

$$p(M^{(j)} \mid \boldsymbol{x}, \Theta), M^{(j)} \in \mathcal{M}$$

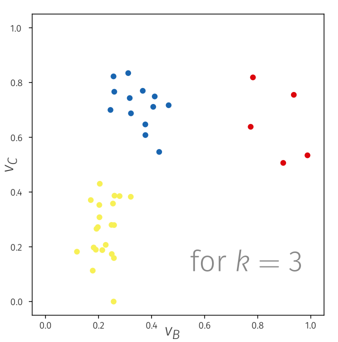


$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
  compute points assignment
  compute parameters Θ
until convergence

EM clustering with Gaussian mixture model

Hard assignment: assign each data point to the model under which it has the highest probability

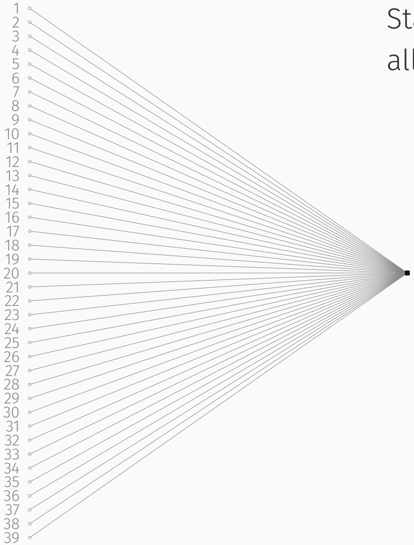
$$\arg\max_{M^{(j)} \in \mathcal{M}} p(M^{(j)} \mid \mathbf{x}, \Theta)$$



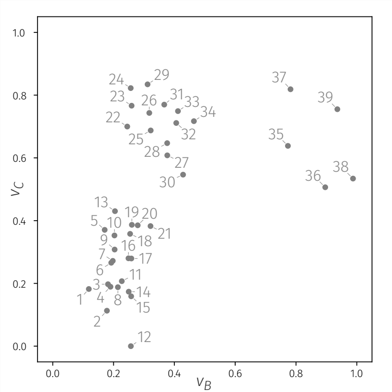
$\theta_j \leftarrow$ initialize $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\quad j \in [1..k]$
$\alpha_j \leftarrow 1/k$, $\quad j \in [1..k]$
repeat
  compute points assignment
  compute parameters Θ
until convergence

Bisecting k-means

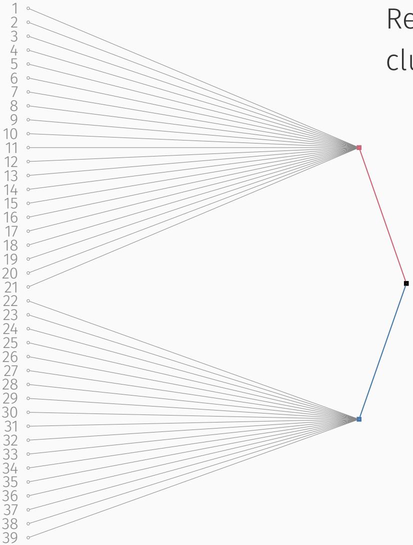
The bisecting k-means algorithm, as the name suggests, works by *bisecting* clusters, i.e. splitting them into two, recursively by applying the *k-means* algorithm with $k = 2$

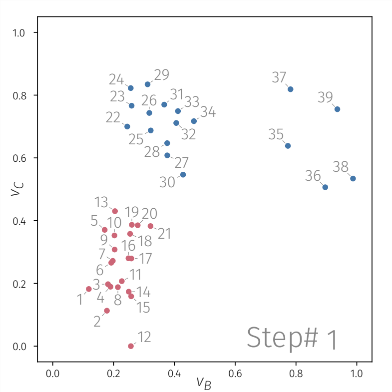


Start with a single cluster containing all data points

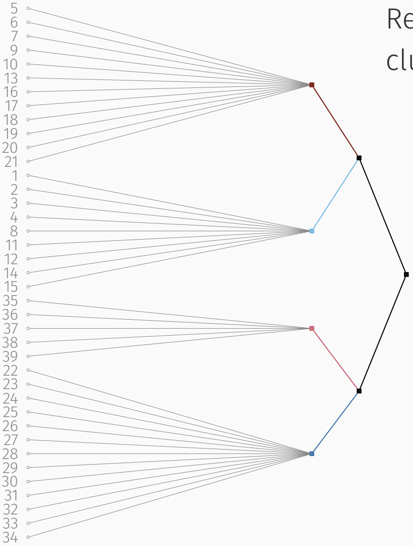


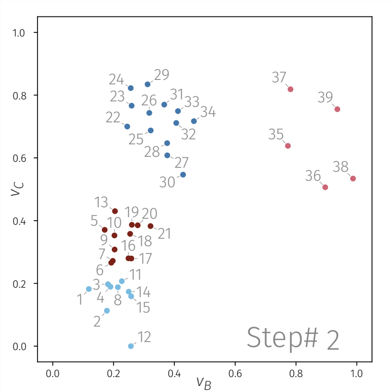
Recursively split non-singleton clusters into two

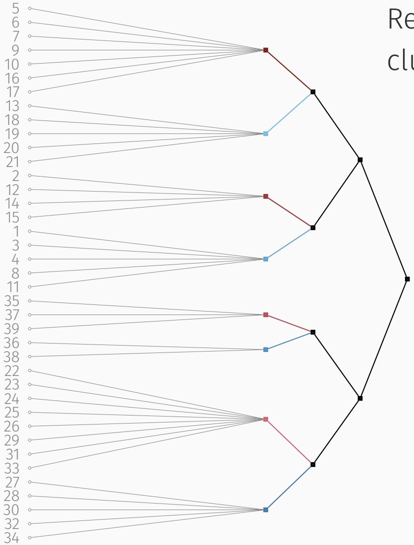




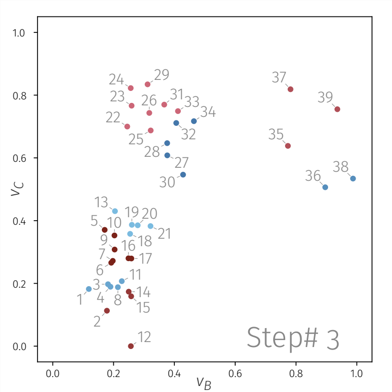
Recursively split non-singleton clusters into two



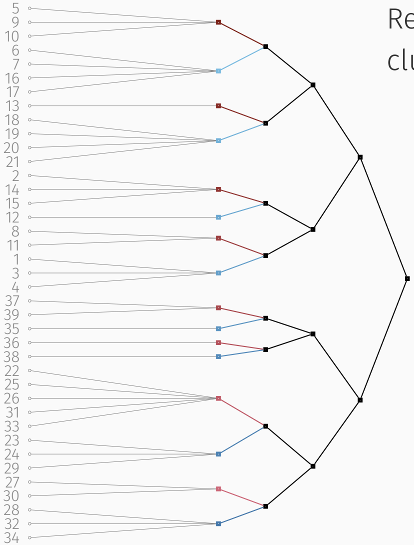




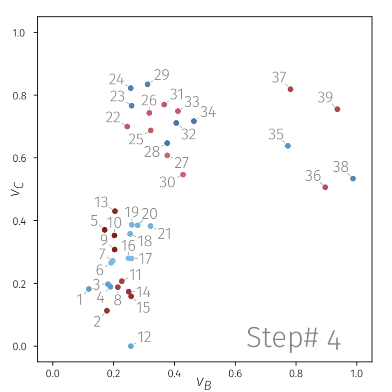
Recursively split non-singleton clusters into two



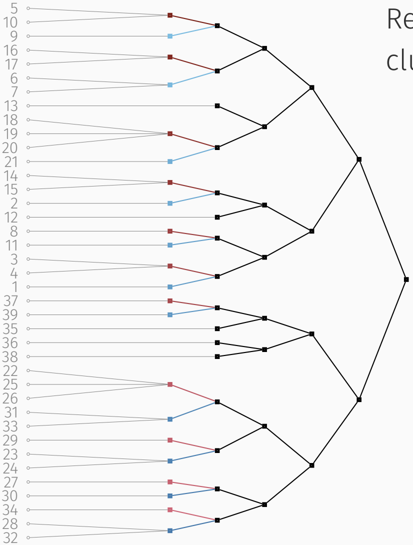
Bisecting k-means



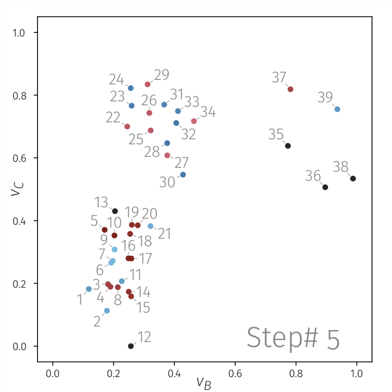
Recursively split non-singleton clusters into two



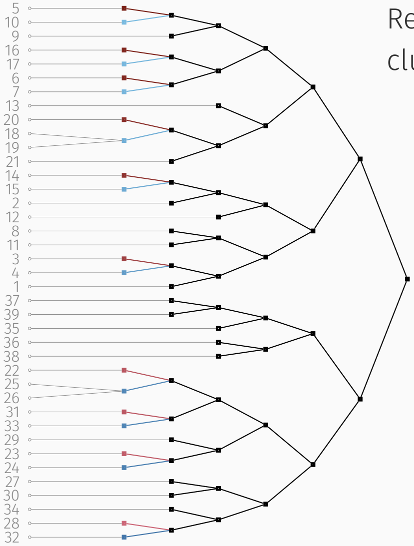
Bisecting k-means



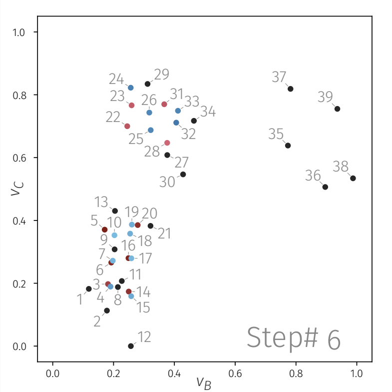
Recursively split non-singleton clusters into two



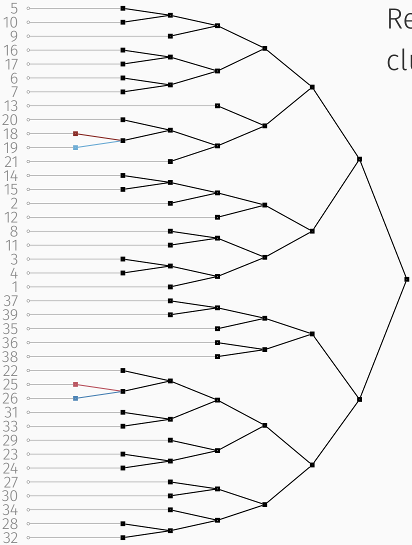
Bisecting k-means



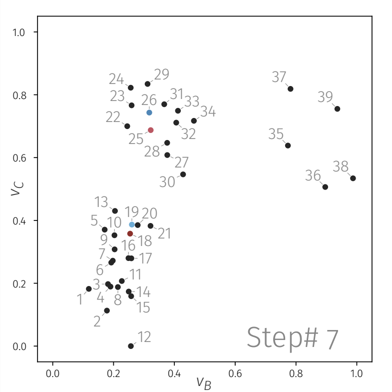
Recursively split non-singleton clusters into two



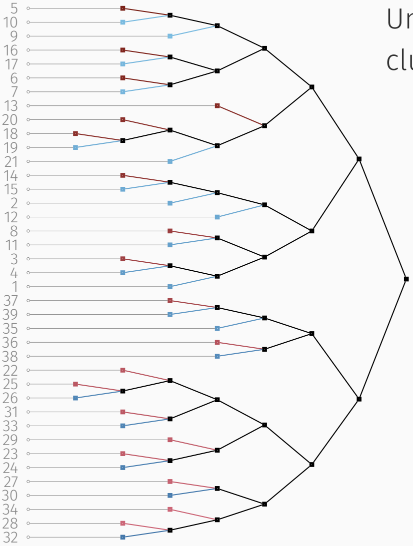
Bisecting k-means



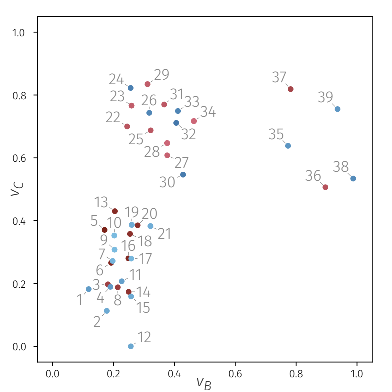
Recursively split non-singleton clusters into two



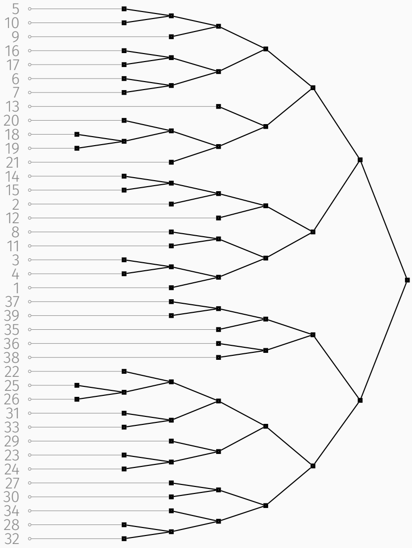
Bisecting k-means



Until each data point is in its own cluster



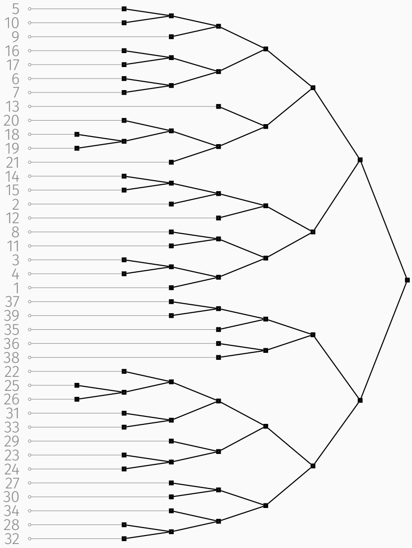
Bisecting k-means



The clusters obtained through the successive iterations of the algorithm form a hierarchy, with higher-level clusters containing lower-level clusters

→ Bisecting k-means is a top-down divisive hierarchical clustering algorithm

Bisecting k-means



The clusters obtained through the successive iterations of the algorithm form a hierarchy, with higher-level clusters containing lower-level clusters

The tree diagram depicting this hierarchical structure is called a dendrogram

Bisecting k-means

Rather than further splitting all current non-singleton clusters, we can split one cluster at a time

In particular, the *least cohesive* current cluster can be selected to be split next

At each step, the number of clusters increases by one
The process can be repeated until
(i) the desired number of clusters is obtained, or
(ii) a desired cohesiveness threshold is reached for all clusters

Bisecting k-means

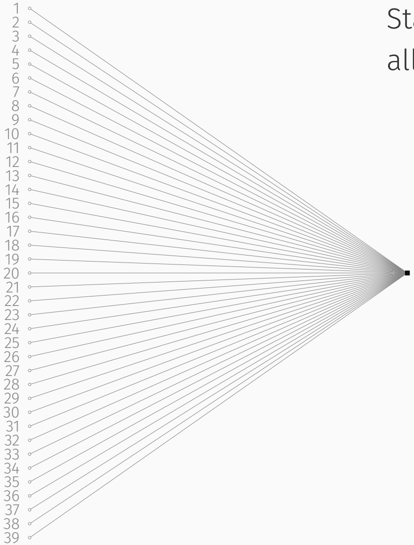
The *cohesiveness* of a cluster can be evaluated using an aggregate of the distances between pairs of points in the cluster, such as the maximum of pairwise distances

Let $\mathcal{C}_m$ denote the current collection of non-singleton clusters
The next cluster to split can be selected as

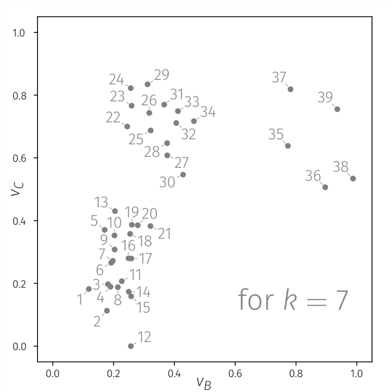
$$\arg\max_{C \in \mathcal{C}_m} \max_{(x,x') \in C^2} \mathrm{d}(x, x')$$

This way, the cluster containing the pair of nodes furthest apart will be selected to be split in the next step

Bisecting k-means

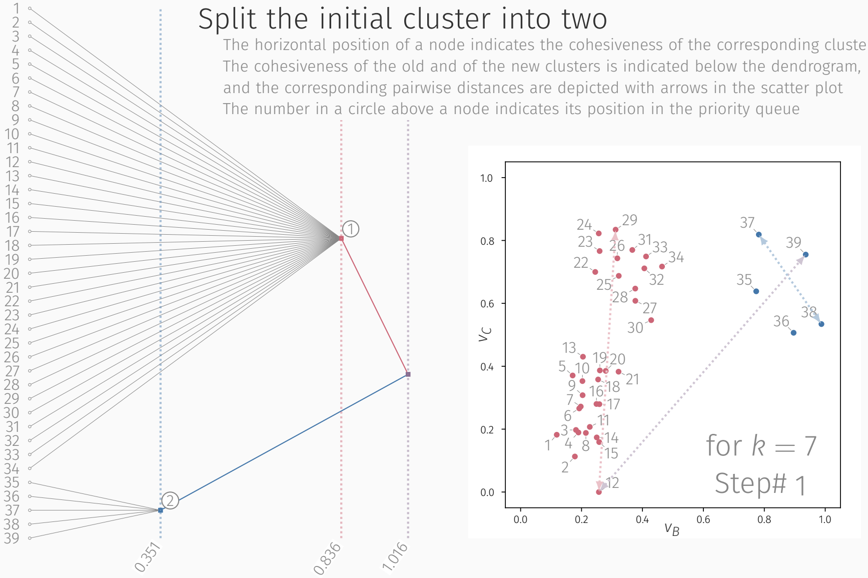


Start with a single cluster containing all data points

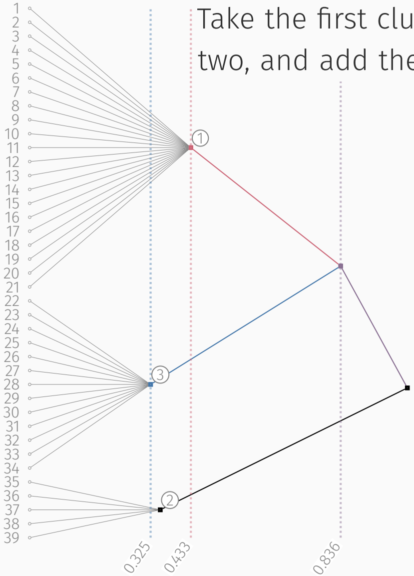


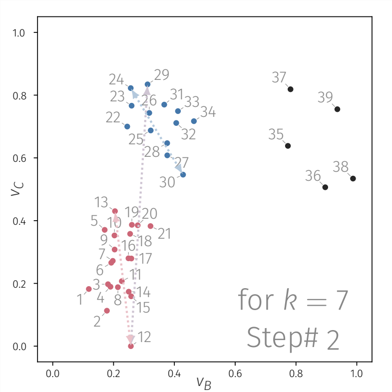
Split the initial cluster into two

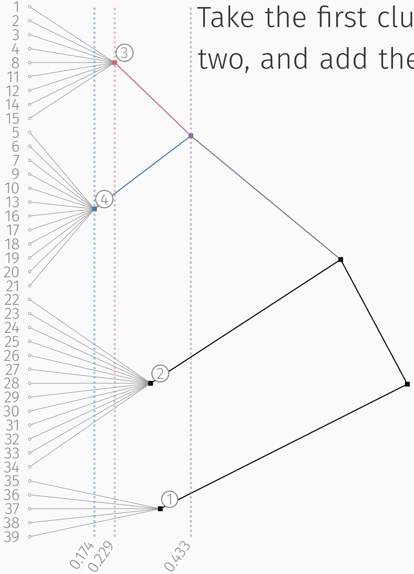
- The horizontal position of a node indicates the cohesiveness of the corresponding cluster
- The cohesiveness of the old and of the new clusters is indicated below the dendrogram, and the corresponding pairwise distances are depicted with arrows in the scatter plot
- The number in a circle above a node indicates its position in the priority queue



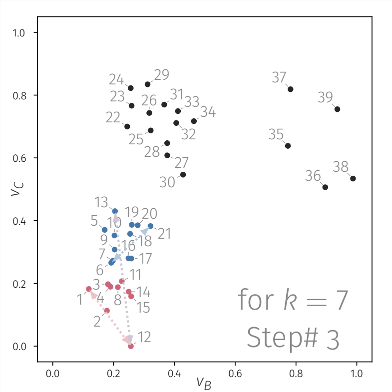
Take the first cluster from the queue, split it into two, and add the new clusters to the queue

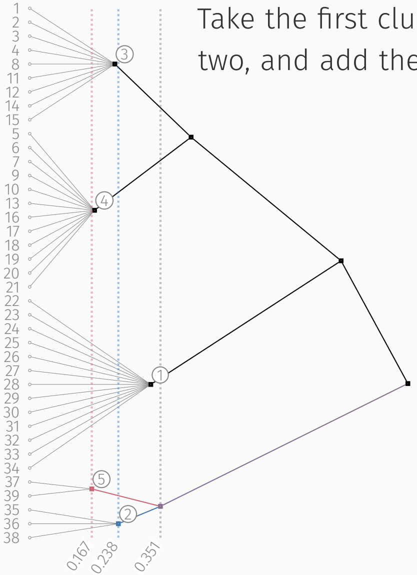




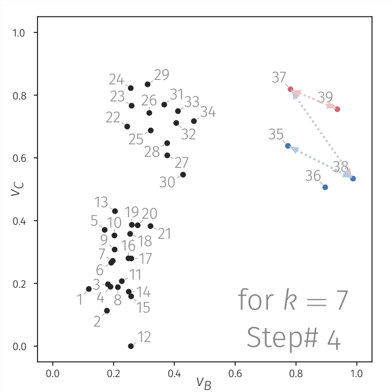


Take the first cluster from the queue, split it into two, and add the new clusters to the queue



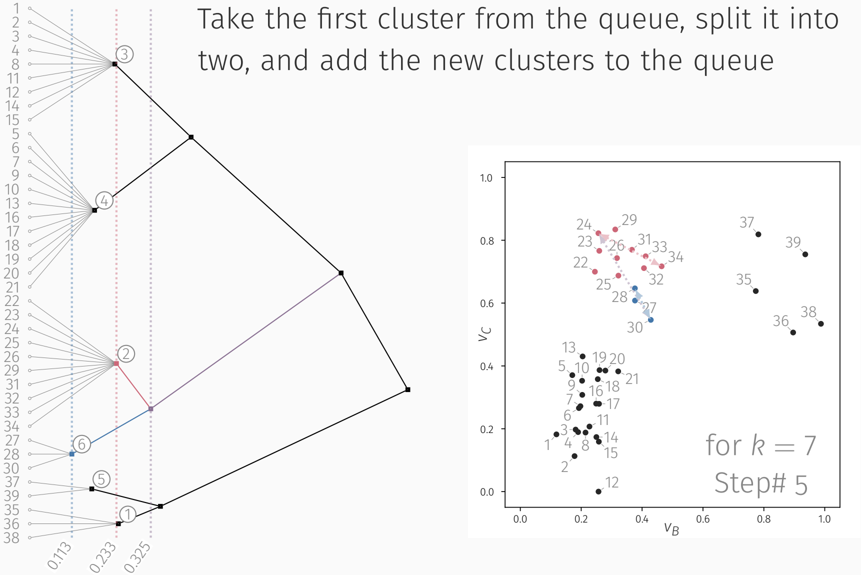


Take the first cluster from the queue, split it into two, and add the new clusters to the queue

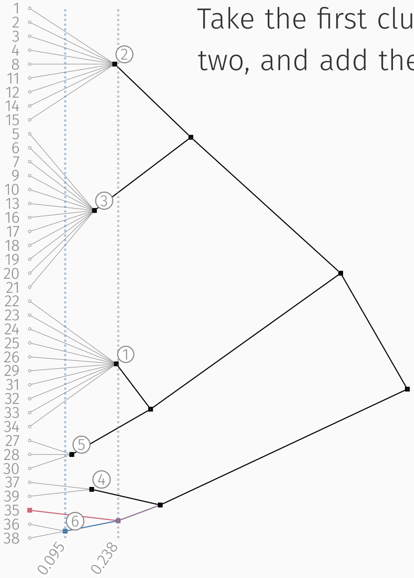


Bisecting k-means

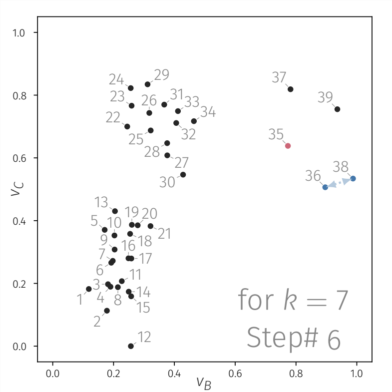
Take the first cluster from the queue, split it into two, and add the new clusters to the queue



Bisecting k-means

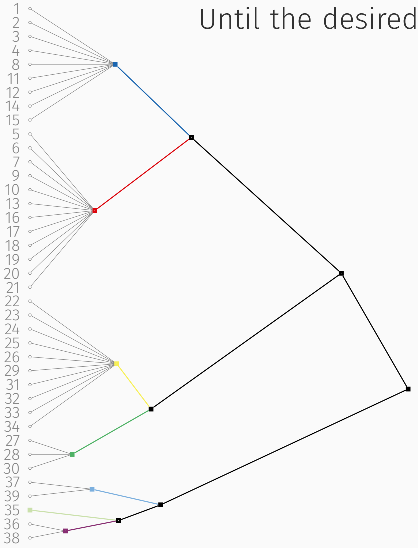


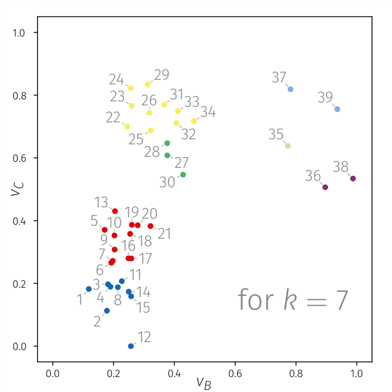
Take the first cluster from the queue, split it into two, and add the new clusters to the queue



Bisecting k-means

Until the desired number of clusters is obtained





Hierarchical agglomerative algorithms

Contrary to *divisive* clustering methods that proceed in a *top-down* manner, hierarchical *agglomerative* clustering methods proceed from the *bottom up*

They start with each data point in its own cluster,
and iteratively merge the pair of clusters that are the closest,
until a single cluster containing all data points is obtained

Hierarchical agglomerative algorithms

Contrary to *divisive* clustering methods that proceed in a *top-down* manner, hierarchical *agglomerative* clustering methods proceed from the *bottom up*

They start with each data point in its own cluster,
and iteratively merge the pair of clusters that are the closest,
until a single cluster containing all data points is obtained

Which clusters are considered to be the closest and selected to be merged in the next step depends on the chosen inter-cluster distance, called the linkage function

Different linkage functions correspond to algorithm variants of agglomerative clustering

Inter-cluster distances

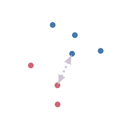
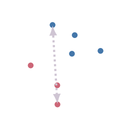
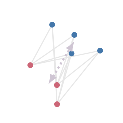
Considering two clusters C_u and C_v, the inter-cluster distance between them, $\mathrm{d}(C_u, C_v)$, is often defined as a function of the pairwise point distances, that is, of the distances between all pairs of points $\boldsymbol{x}$ and $\boldsymbol{x}'$ respectively from C_u and C_v

Furthermore, the distance between cluster C_{uv}, resulting from the merger of C_u and C_v, and any other cluster C_s, $\mathrm{d}(C_{uv}, C_v)$, can often accordingly be computed as a function of the distances between C_u and C_s and between C_v and C_s

Among the most common linkage functions are

single linkage minimum of pairwise point distances
complete linkage maximum of pairwise point distances
average linkage average of pairwise point distances

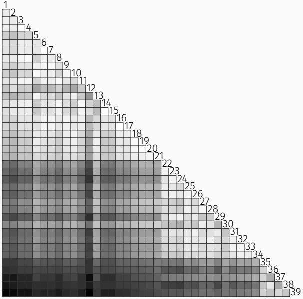
Inter-cluster distances, linkage functions

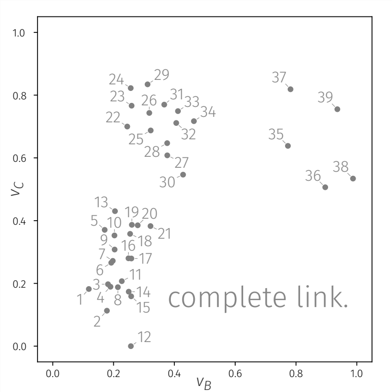
Linkage function	single a.k.a. minimum	complete a.k.a. maximum	average
			
Inter-cluster distance (as function of pairwise point distances)			
$\mathrm{d}(C_u, C_v)$	$\min_{(x,x')\in C_u \times C_v} \mathrm{d}(x, x')$	$\max_{(x,x')\in C_u \times C_v} \mathrm{d}(x, x')$	$\frac{\sum_{(x,x')\in C_u \times C_v} \mathrm{d}(x, x')}{\lvert C_u\rvert \cdot \lvert C_v\rvert}$
Distance merging (for $C_{uv} = C_u \cup C_v$ and any other cluster C_s)			
$\mathrm{d}(C_{uv}, C_s)$	$\min_{C\in\{C_u,C_v\}} \mathrm{d}(C, C_s)$	$\max_{C\in\{C_u,C_v\}} \mathrm{d}(C, C_s)$	$\frac{\sum_{C\in\{C_u,C_v\}} \lvert C\rvert \cdot \mathrm{d}(C, C_s)}{\lvert C_u\rvert + \lvert C_v\rvert}$

1 ◦
2 ◦
3 ◦
4 ◦
5 ◦
6 ◦
7 ◦
8 ◦
9 ◦
10 ◦
11 ◦
12 ◦
13 ◦
14 ◦
15 ◦
16 ◦
17 ◦
18 ◦
19 ◦
20 ◦
21 ◦
22 ◦
23 ◦
24 ◦
25 ◦
26 ◦
27 ◦
28 ◦
29 ◦
30 ◦
31 ◦
32 ◦
33 ◦
34 ◦
35 ◦
36 ◦
37 ◦
38 ◦
39 ◦

Start with each data point in its own cluster

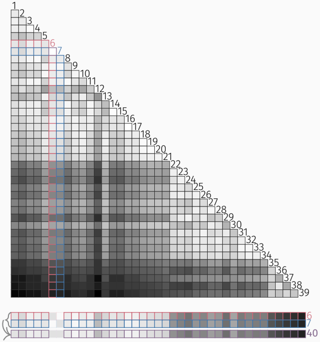
- Distances must be computed for all pairs of points
- The lower triangle of the symmetric distance matrix is depicted, with darker shades of gray indicating larger values

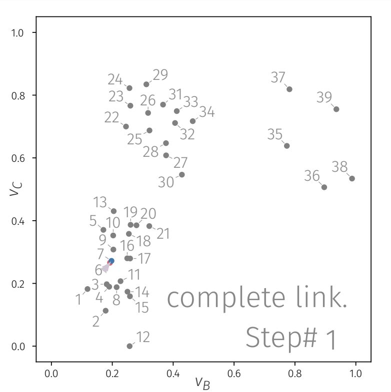




Select the two clusters that are closest and merge them

- The clusters corresponding to the smallest value in the distance matrix are selected (blue and red)
- They are merged into a new cluster (purple), and the distance matrix is updated
- Specifically, the distance vectors for the two clusters are aggregated according to the linkage function, to compute distances for the new cluster (as depicted below the distance matrix)
- The two vectors are removed and the new one is added to the matrix

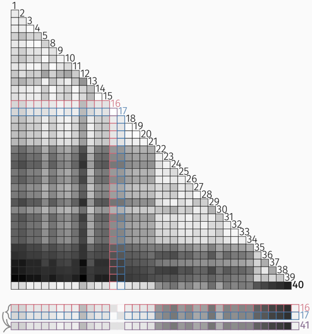


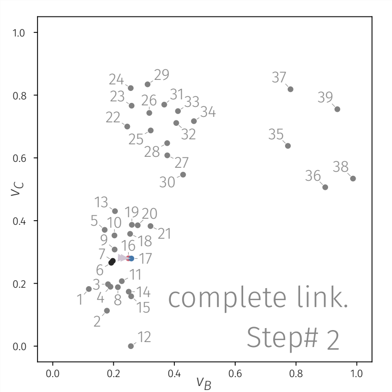


Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them

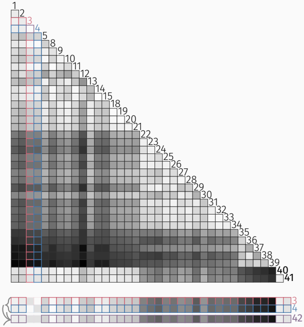
Iterate...

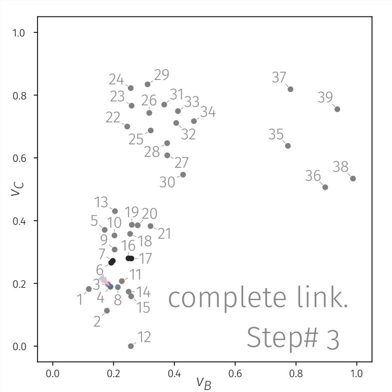


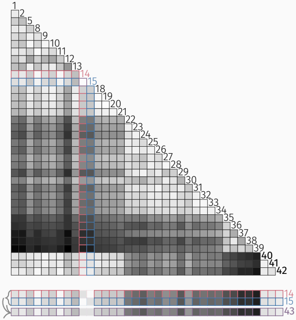


Select the two clusters that are closest and merge them

Iterate...

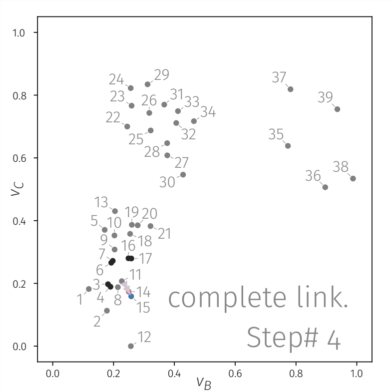






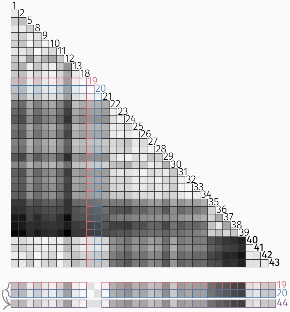
Select the two clusters that are closest and merge them

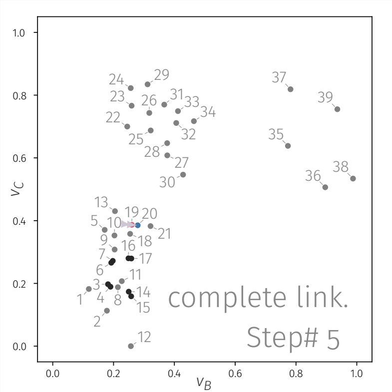
Iterate...



Select the two clusters that are closest and merge them

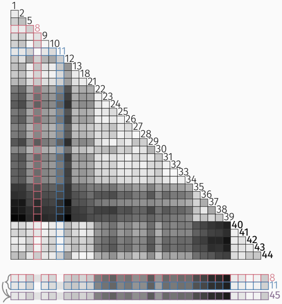
Iterate…

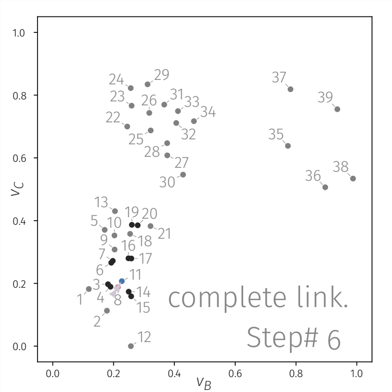




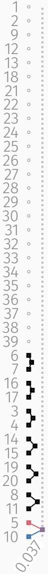
Select the two clusters that are closest and merge them

Iterate...



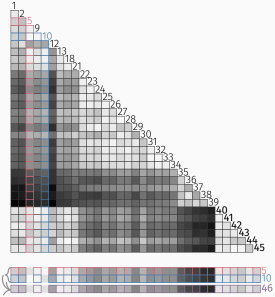


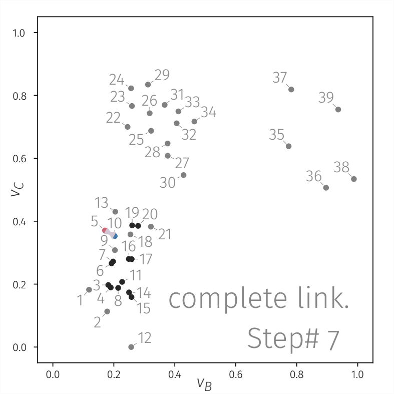
Agglomerative clustering with complete linkage



Select the two clusters that are closest and merge them

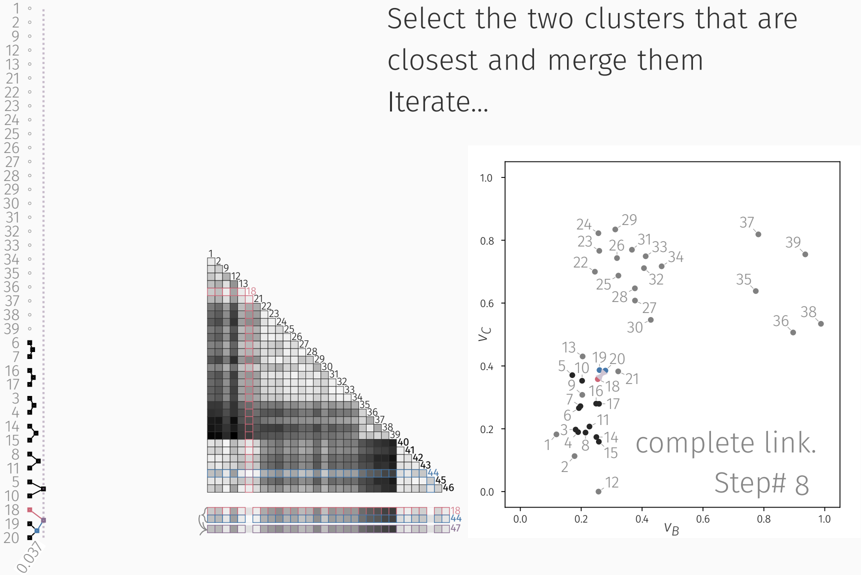
Iterate...





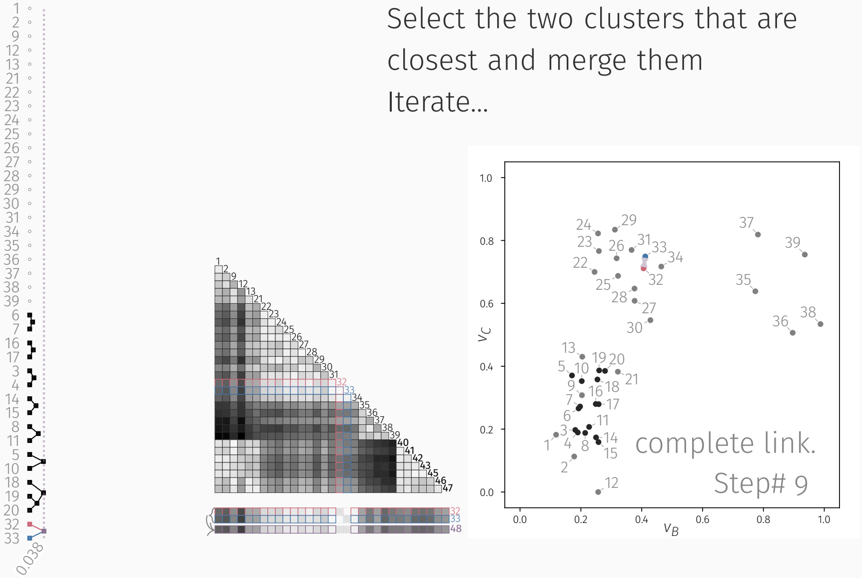
Select the two clusters that are closest and merge them

Iterate...

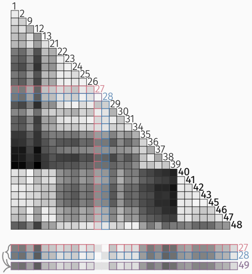


Select the two clusters that are closest and merge them

Iterate…

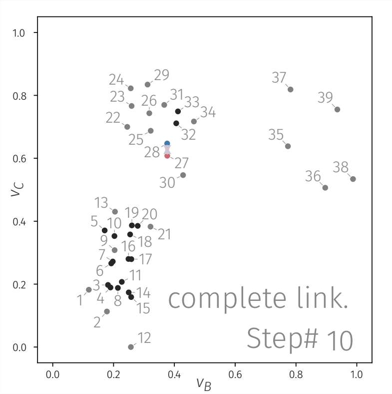


Agglomerative clustering with complete linkage



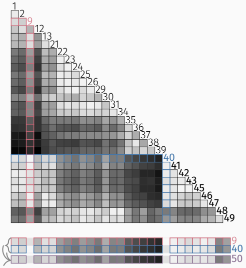
Select the two clusters that are closest and merge them

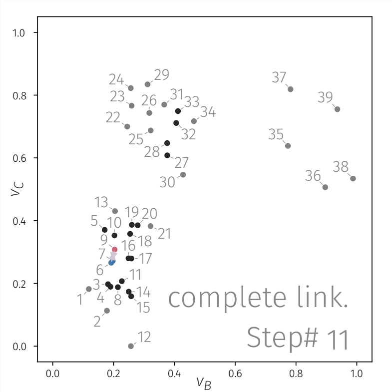
Iterate...



Select the two clusters that are closest and merge them

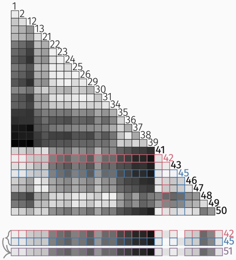
Iterate...

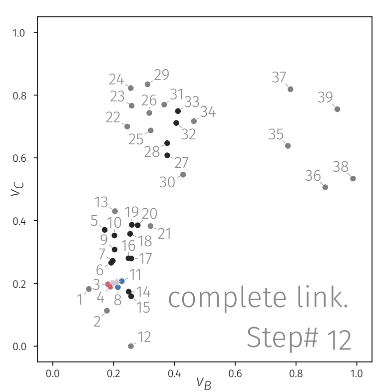




Select the two clusters that are closest and merge them

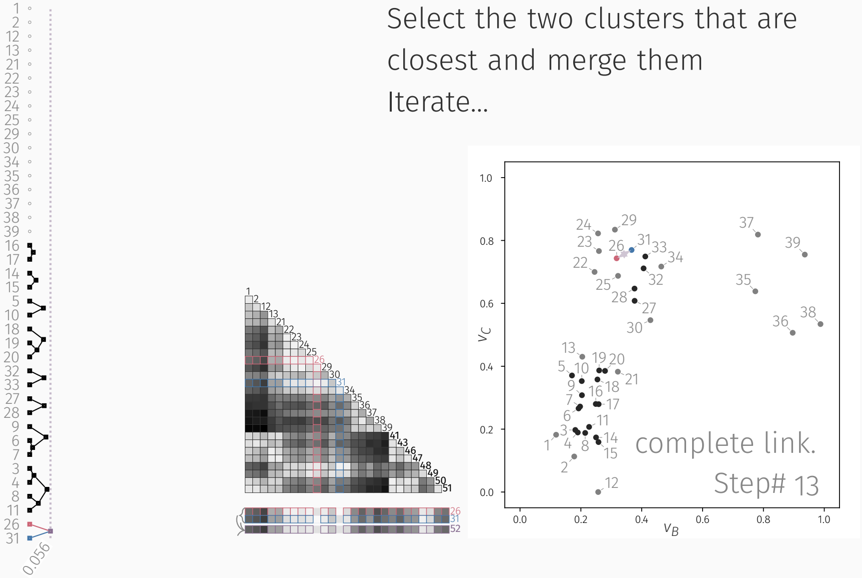
Iterate...





Select the two clusters that are closest and merge them

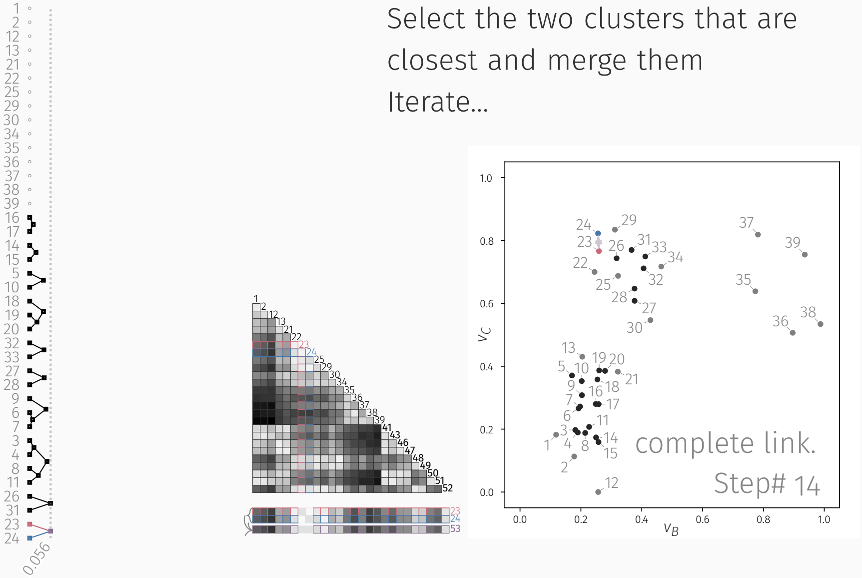
Iterate...



Agglomerative clustering with complete linkage

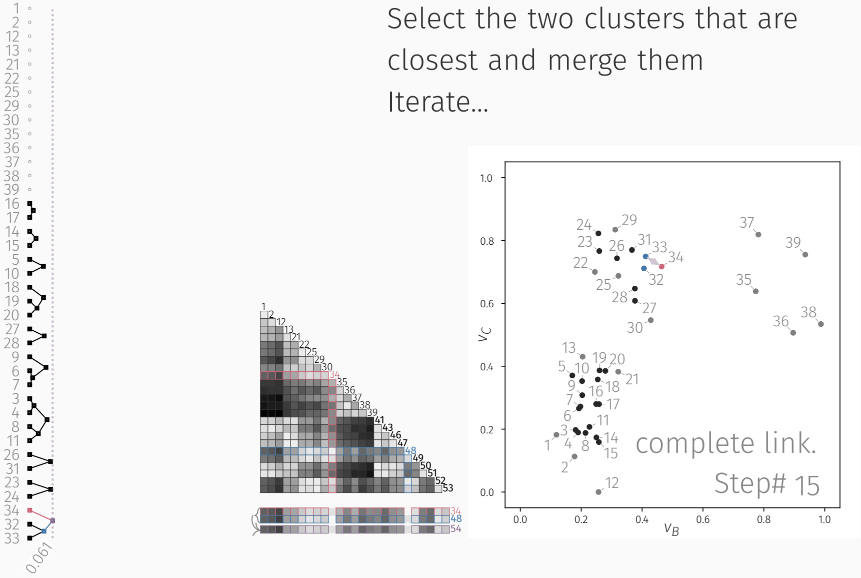
Select the two clusters that are closest and merge them

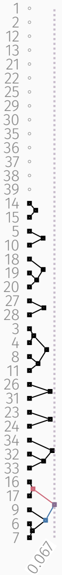
Iterate...



Select the two clusters that are closest and merge them

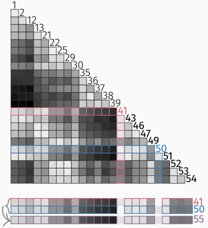
Iterate...

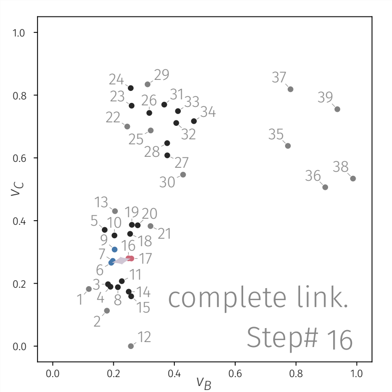




Select the two clusters that are closest and merge them

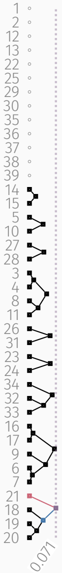
Iterate...

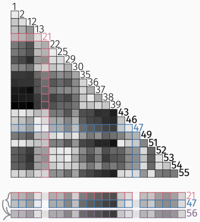


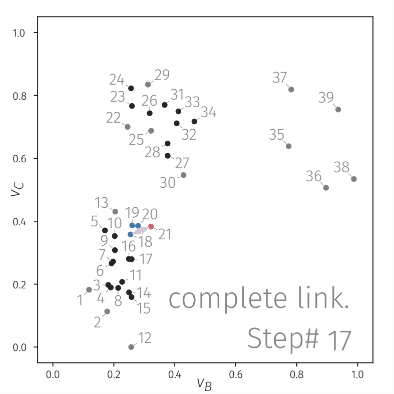


Select the two clusters that are closest and merge them

Iterate...

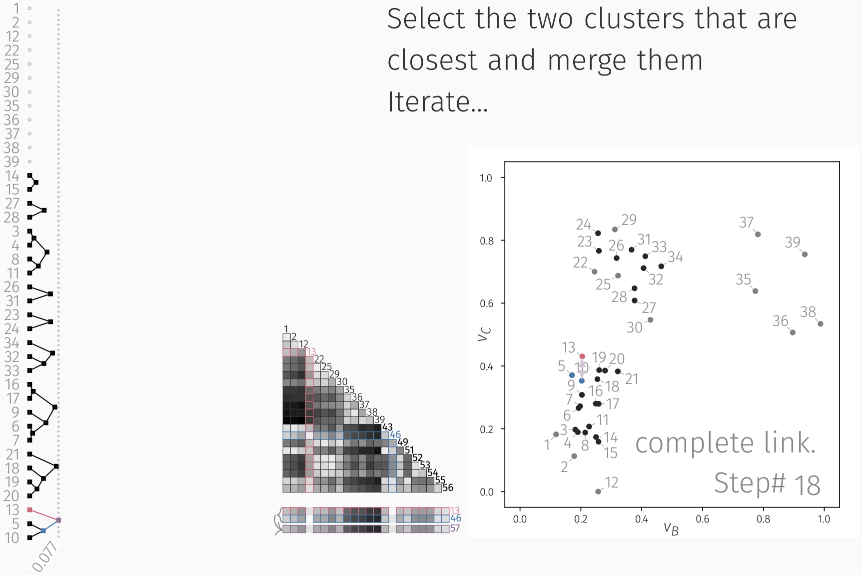






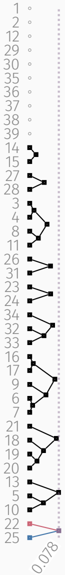
Select the two clusters that are closest and merge them

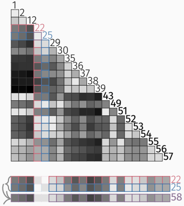
Iterate…

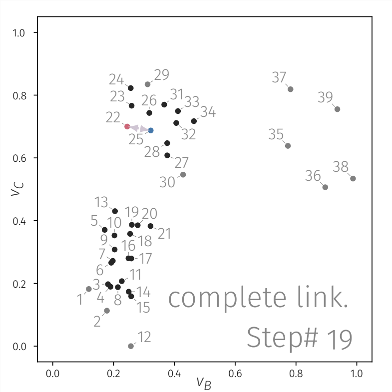


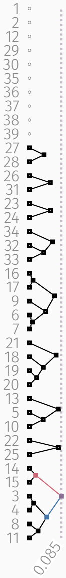
Select the two clusters that are closest and merge them

Iterate...



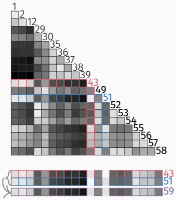


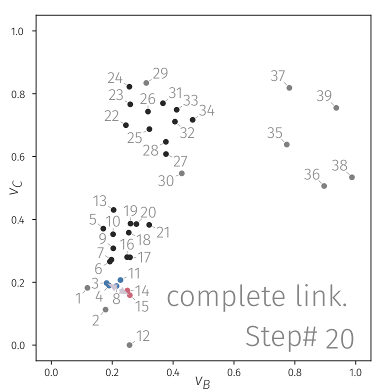




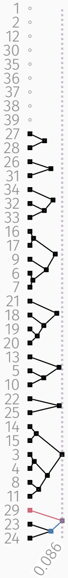
Select the two clusters that are closest and merge them

Iterate...



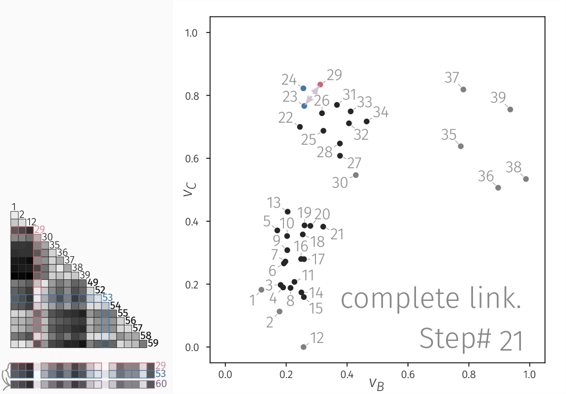


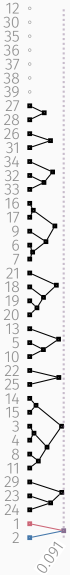
Agglomerative clustering with complete linkage



Select the two clusters that are closest and merge them

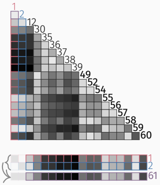
Iterate...

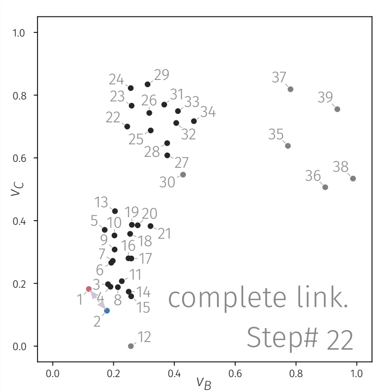




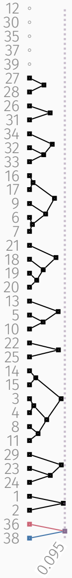
Select the two clusters that are closest and merge them

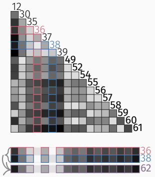
Iterate...





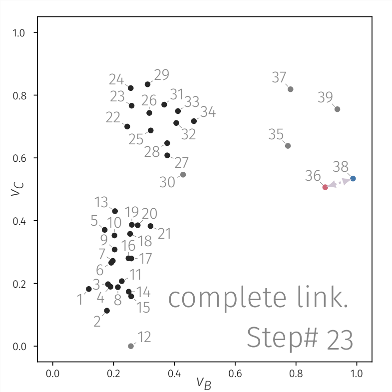
Agglomerative clustering with complete linkage





Select the two clusters that are closest and merge them

Iterate...

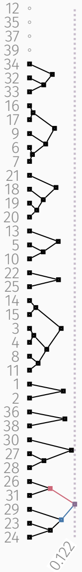


Agglomerative clustering with complete linkage

Select the two clusters that are closest and merge them

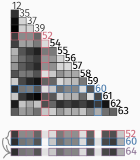
Iterate...

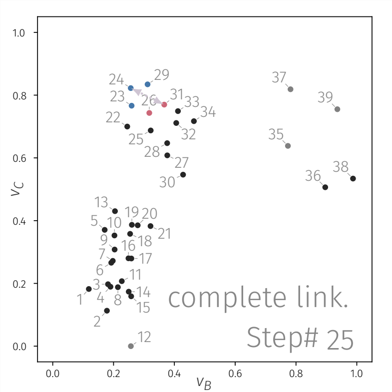
Agglomerative clustering with complete linkage

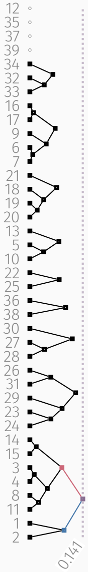


Select the two clusters that are closest and merge them

Iterate…

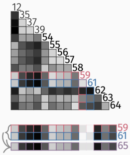


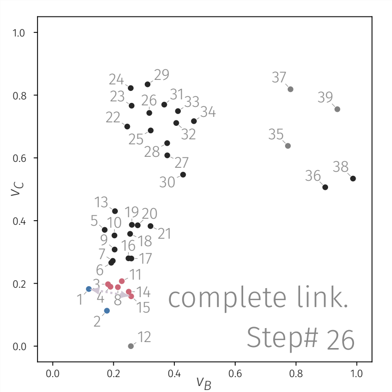




Select the two clusters that are closest and merge them

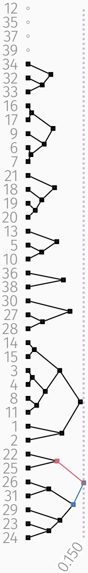
Iterate...

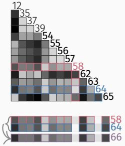


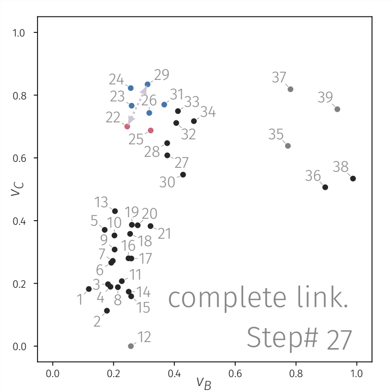


Select the two clusters that are closest and merge them

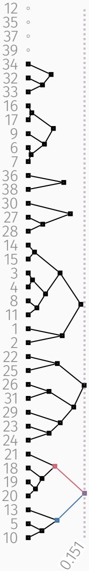
Iterate...





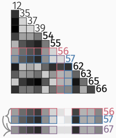


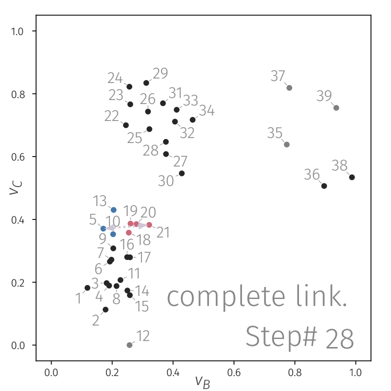
Agglomerative clustering with complete linkage

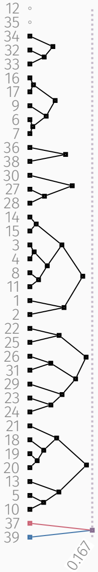


Select the two clusters that are closest and merge them

Iterate...

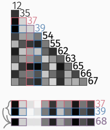


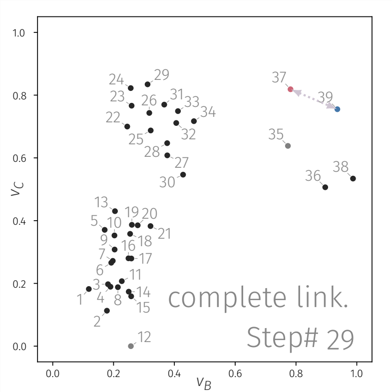


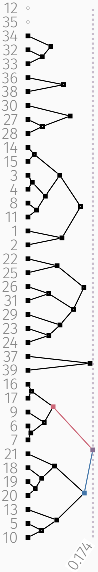


Select the two clusters that are closest and merge them

Iterate...

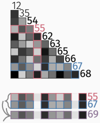


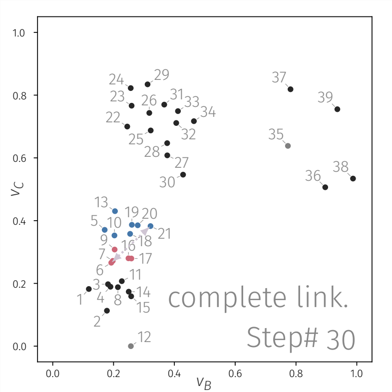




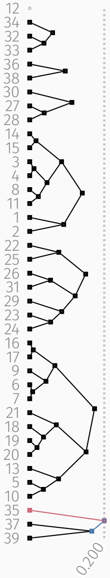
Select the two clusters that are closest and merge them

Iterate...



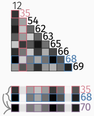


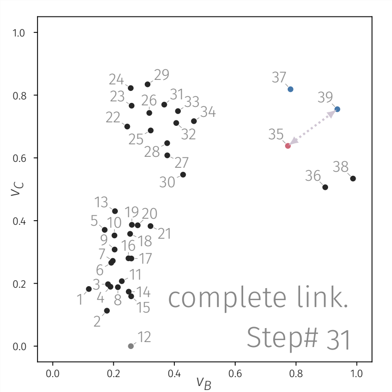
Agglomerative clustering with complete linkage



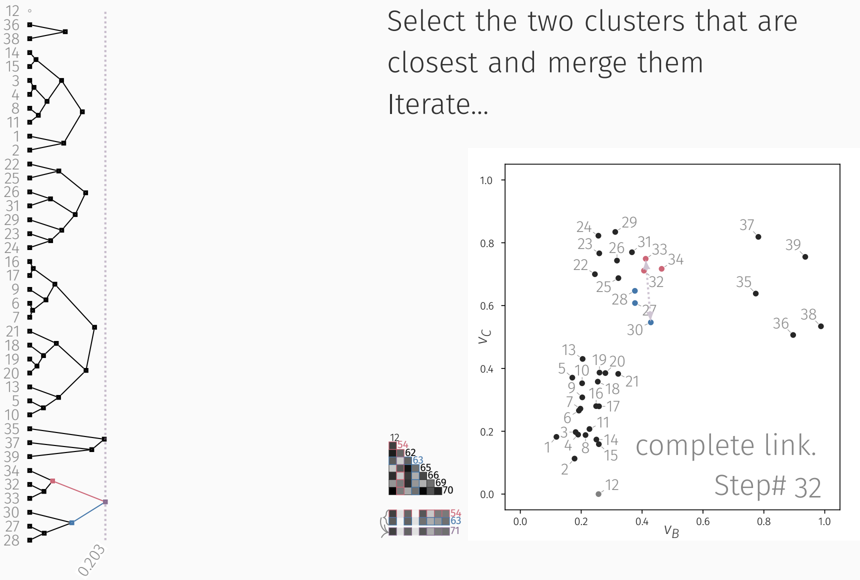
Select the two clusters that are closest and merge them

Iterate...



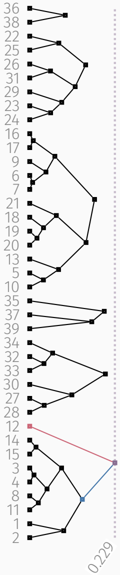


Agglomerative clustering with complete linkage



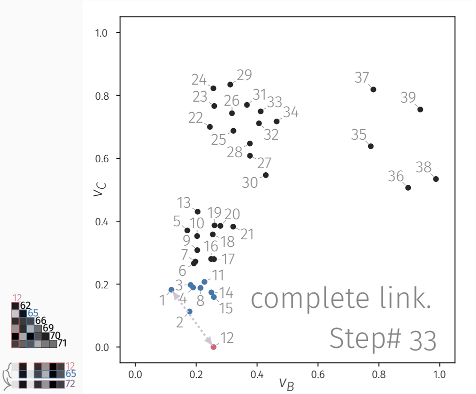
Select the two clusters that are closest and merge them

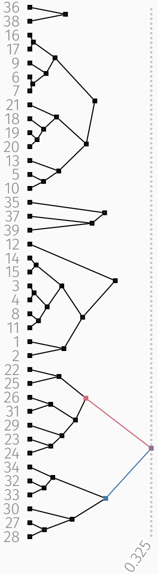
Iterate...



Select the two clusters that are closest and merge them

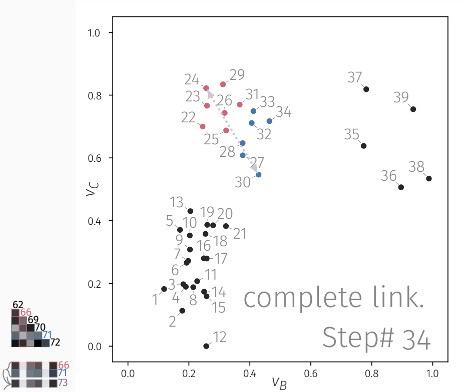
Iterate...



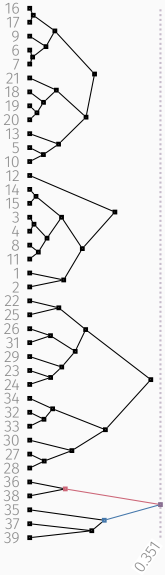


Select the two clusters that are closest and merge them

Iterate...

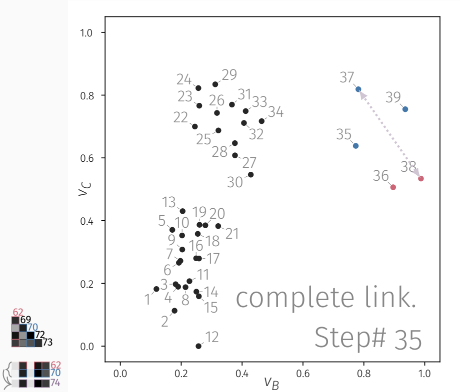


Agglomerative clustering with complete linkage

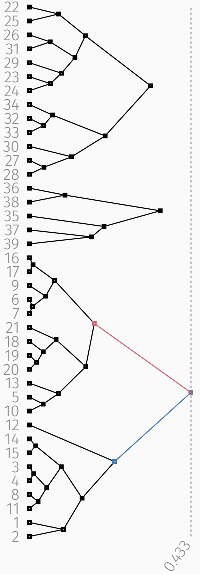


Select the two clusters that are closest and merge them

Iterate...

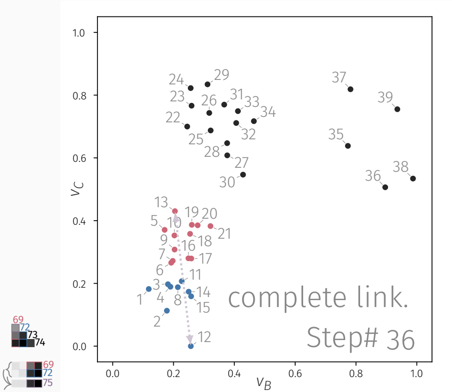


Agglomerative clustering with complete linkage

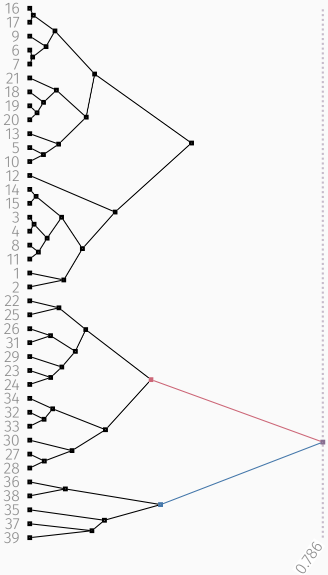


Select the two clusters that are closest and merge them

Iterate...

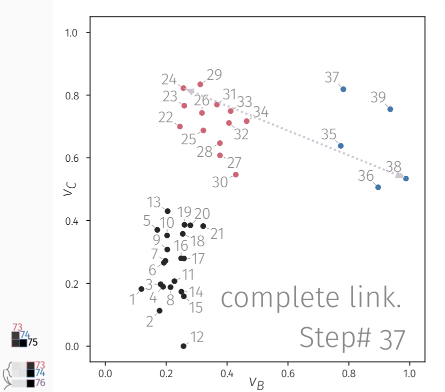


Agglomerative clustering with complete linkage

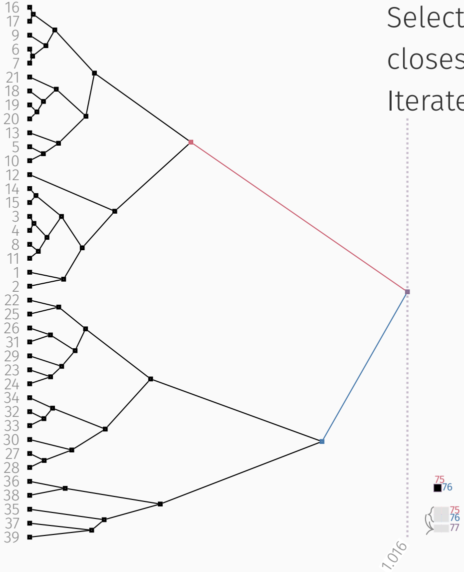


Select the two clusters that are closest and merge them

Iterate...

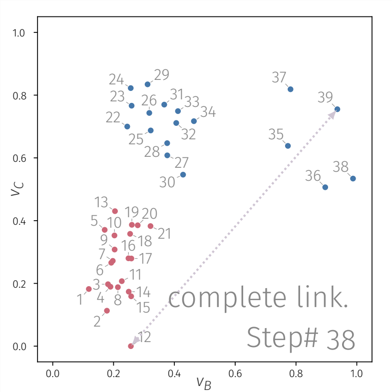


Agglomerative clustering with complete linkage

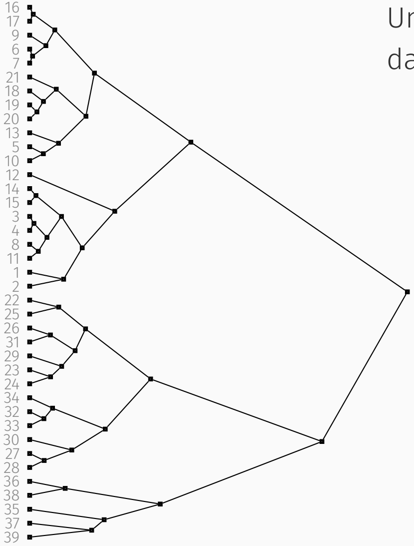


Select the two clusters that are closest and merge them

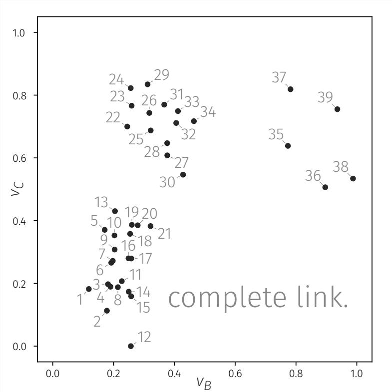
Iterate...

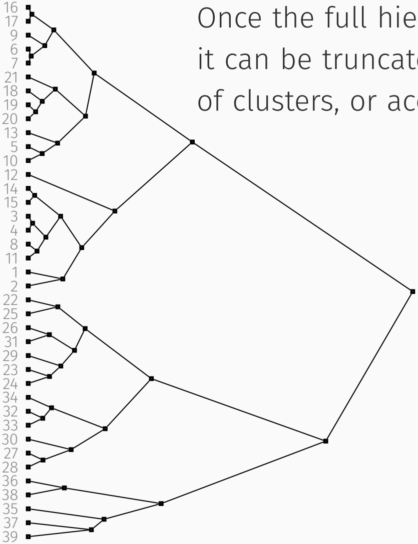


Agglomerative clustering with complete linkage

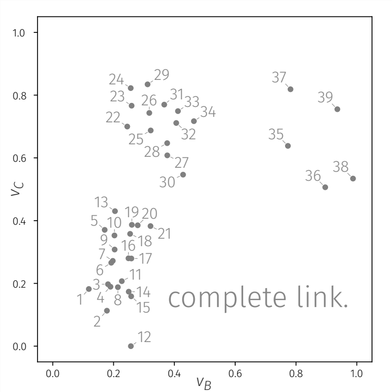


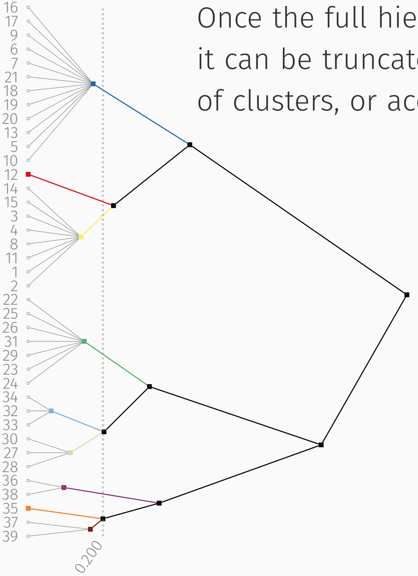
Until a single cluster containing all data points is obtained



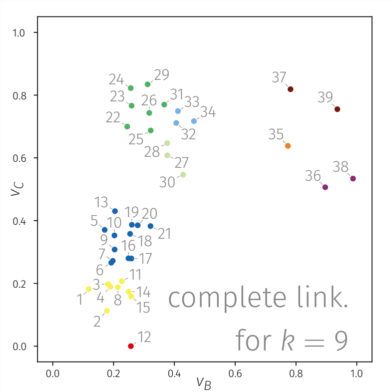


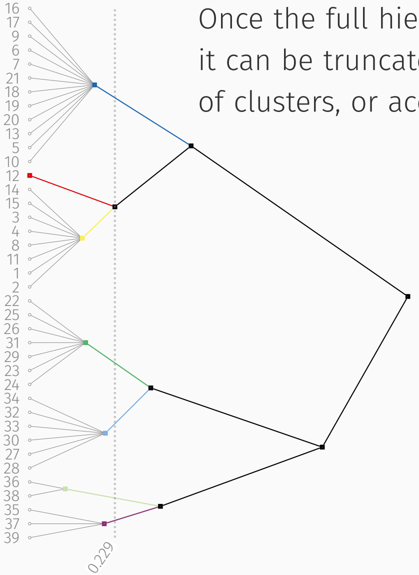
Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances



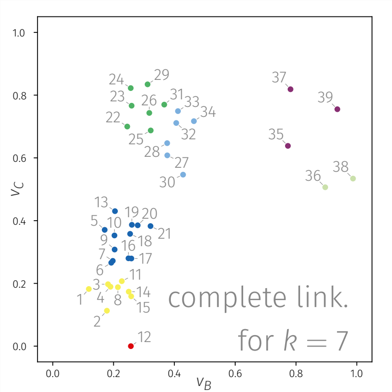


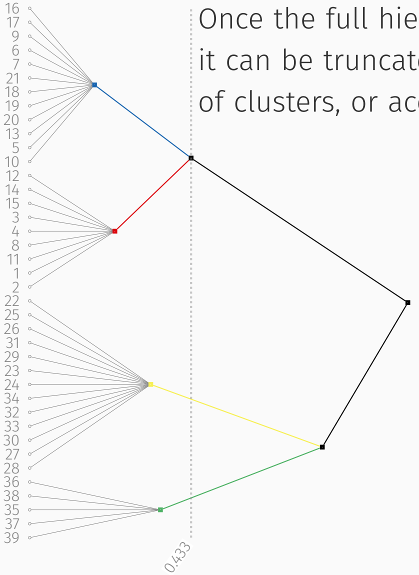
Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances



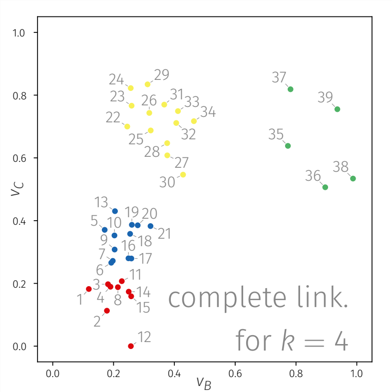


Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances



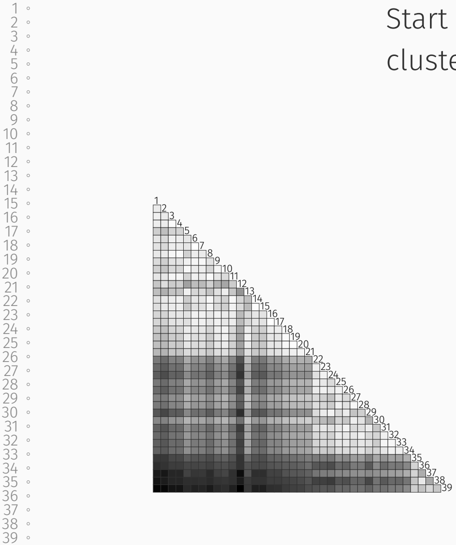


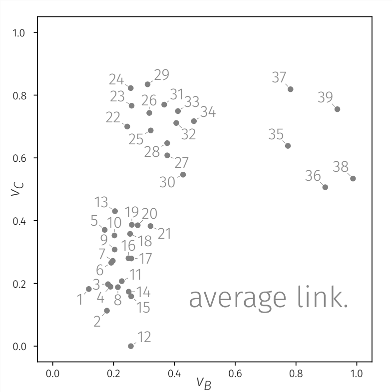
Once the full hierarchy of clusters has been built, it can be truncated to obtain the desired number of clusters, or according to inter-cluster distances



Agglomerative clustering with average linkage

Start with each data point in its own cluster





Agglomerative clustering with average linkage

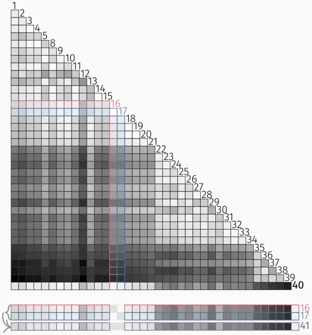
Select the two clusters that are closest and merge them

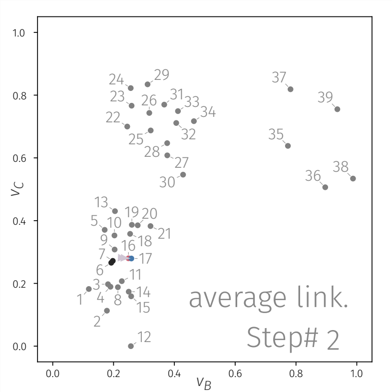
Iterate...

Agglomerative clustering with average linkage

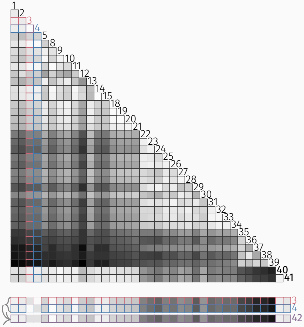
Select the two clusters that are closest and merge them

Iterate...



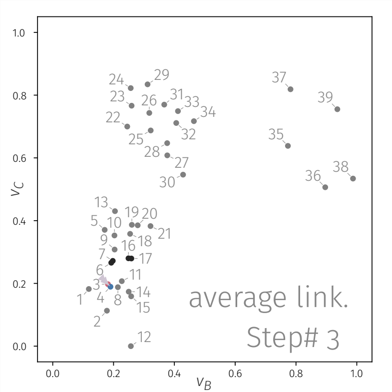


Agglomerative clustering with average linkage



Select the two clusters that are closest and merge them

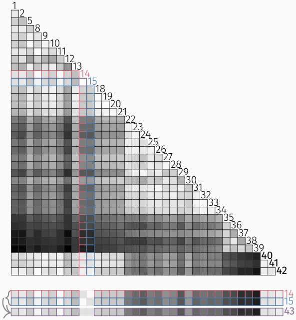
Iterate...

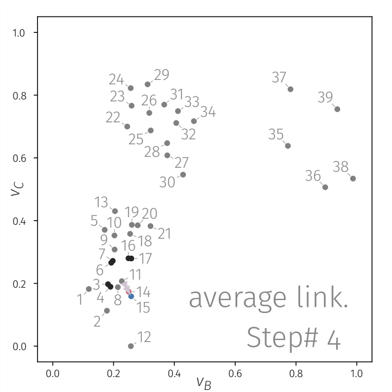


Agglomerative clustering with average linkage

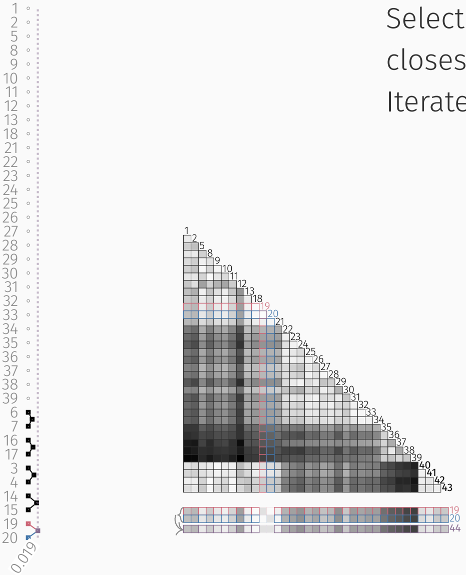
Select the two clusters that are closest and merge them

Iterate…



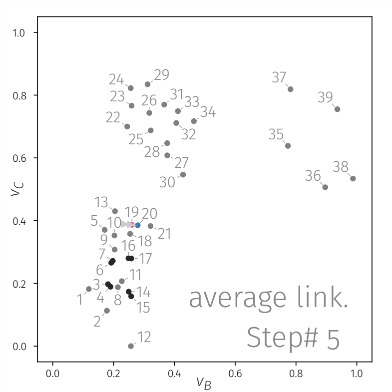


Agglomerative clustering with average linkage



Select the two clusters that are closest and merge them

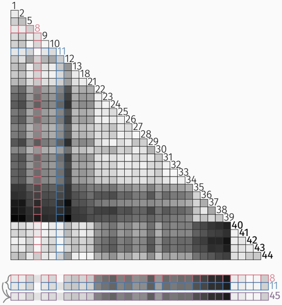
Iterate...

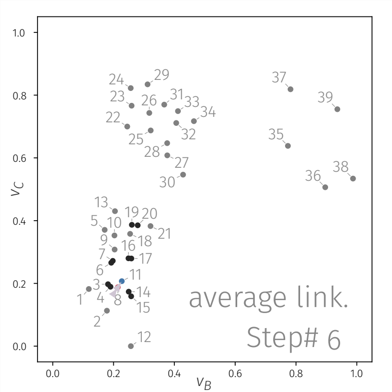


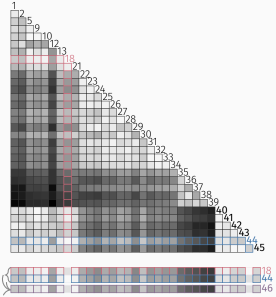
Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

Iterate...

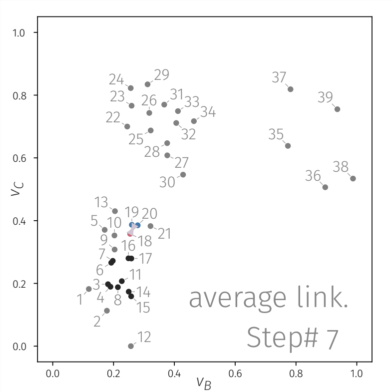






Select the two clusters that are closest and merge them

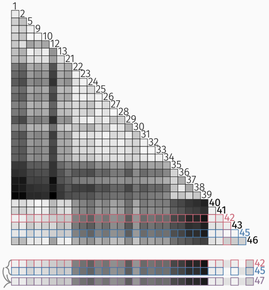
Iterate...

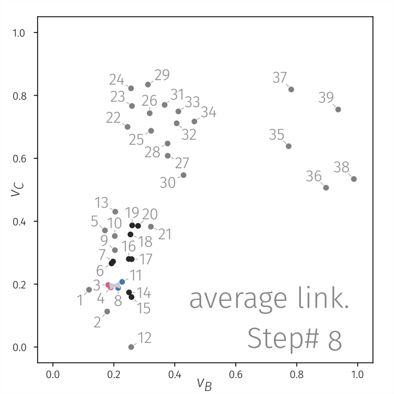


Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

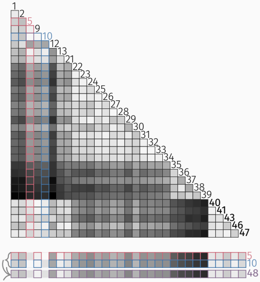
Iterate...

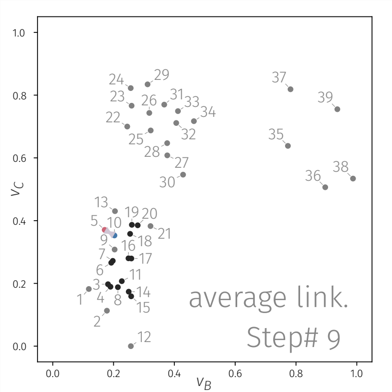




Select the two clusters that are closest and merge them

Iterate...

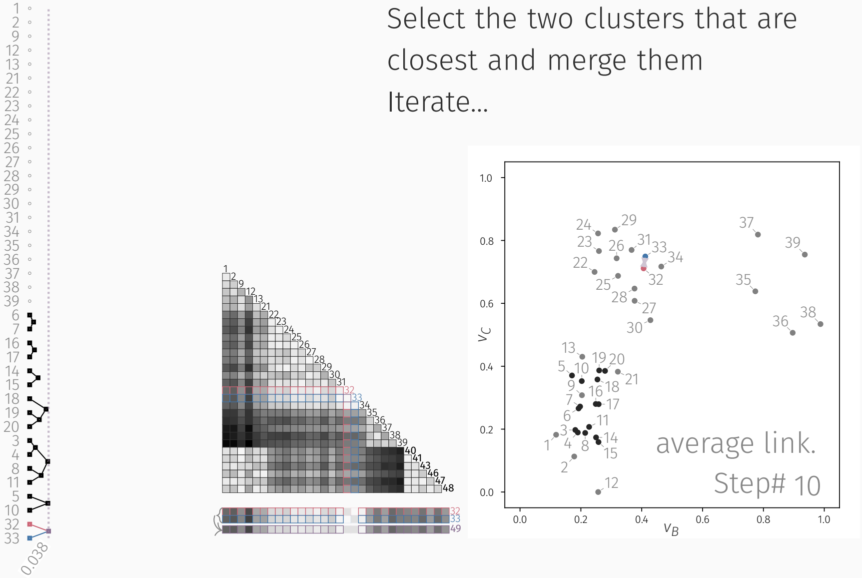




Agglomerative clustering with average linkage

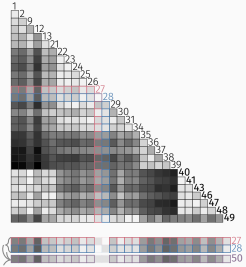
Select the two clusters that are closest and merge them

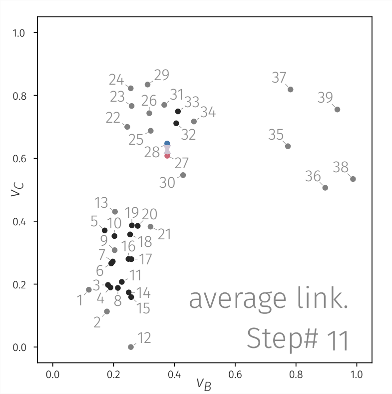
Iterate...

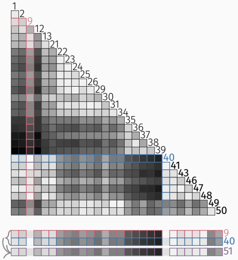


Select the two clusters that are closest and merge them

Iterate...

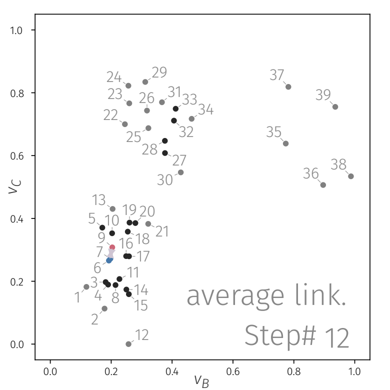






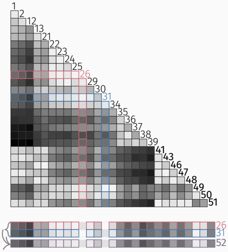
Select the two clusters that are closest and merge them

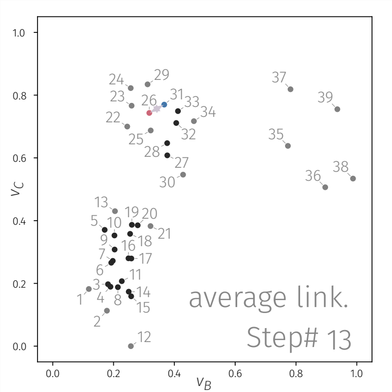
Iterate...

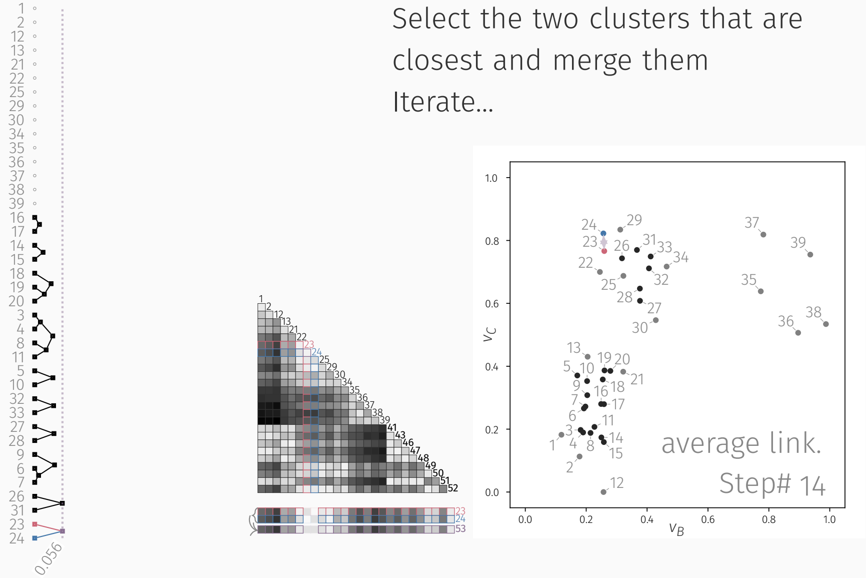


Select the two clusters that are closest and merge them

Iterate...



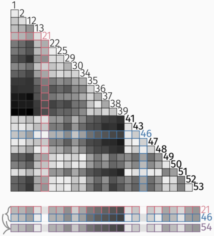




Select the two clusters that are closest and merge them

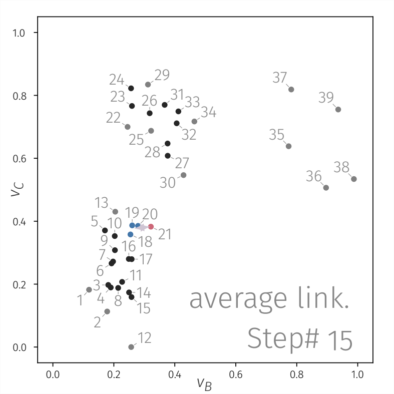
Iterate...

Agglomerative clustering with average linkage

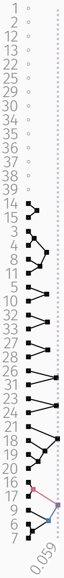


Select the two clusters that are closest and merge them

Iterate...

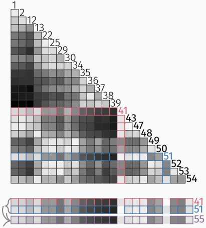


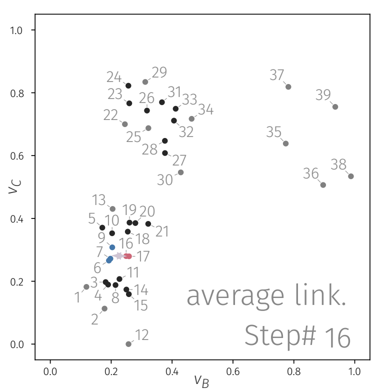
Agglomerative clustering with average linkage



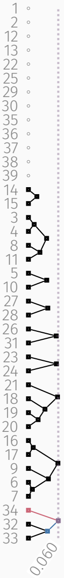
Select the two clusters that are closest and merge them

Iterate...



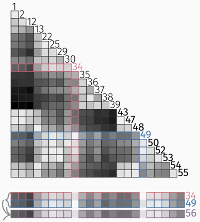


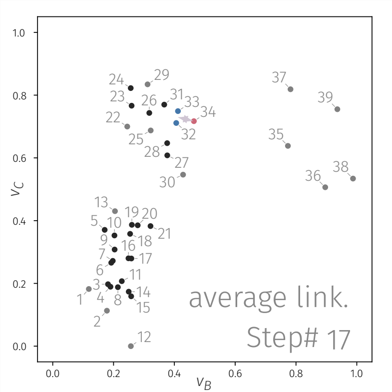
Agglomerative clustering with average linkage



Select the two clusters that are closest and merge them

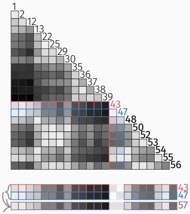
Iterate...

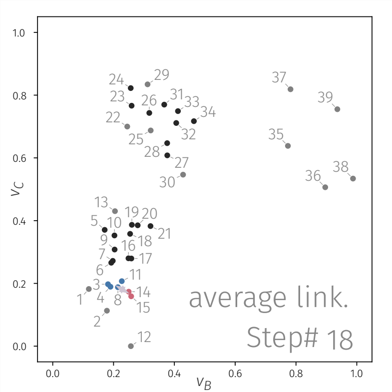




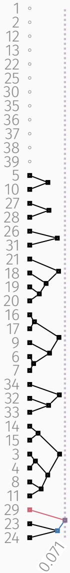
Select the two clusters that are closest and merge them

Iterate...



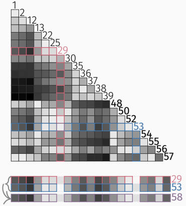


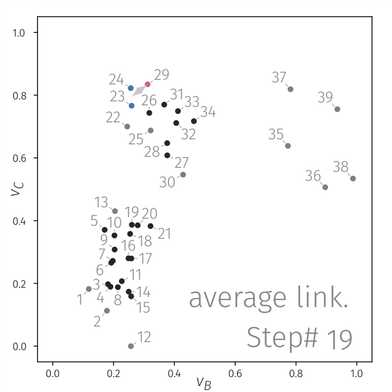
Agglomerative clustering with average linkage



Select the two clusters that are closest and merge them

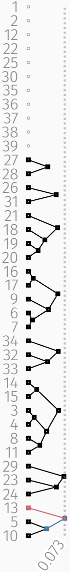
Iterate...

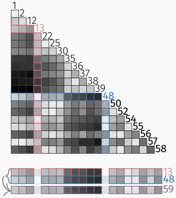


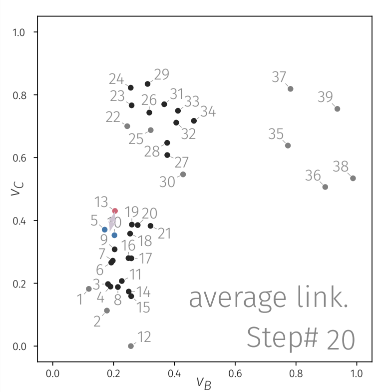


Select the two clusters that are closest and merge them

Iterate…



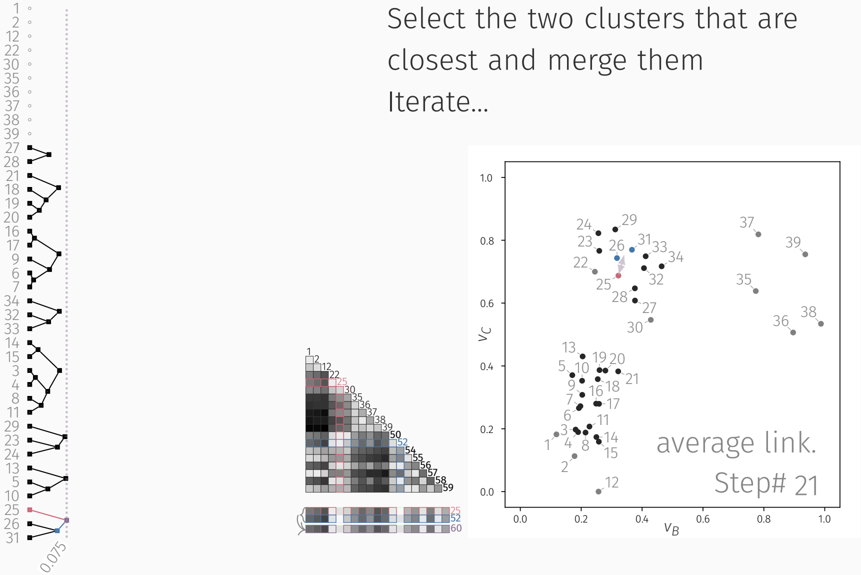




Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

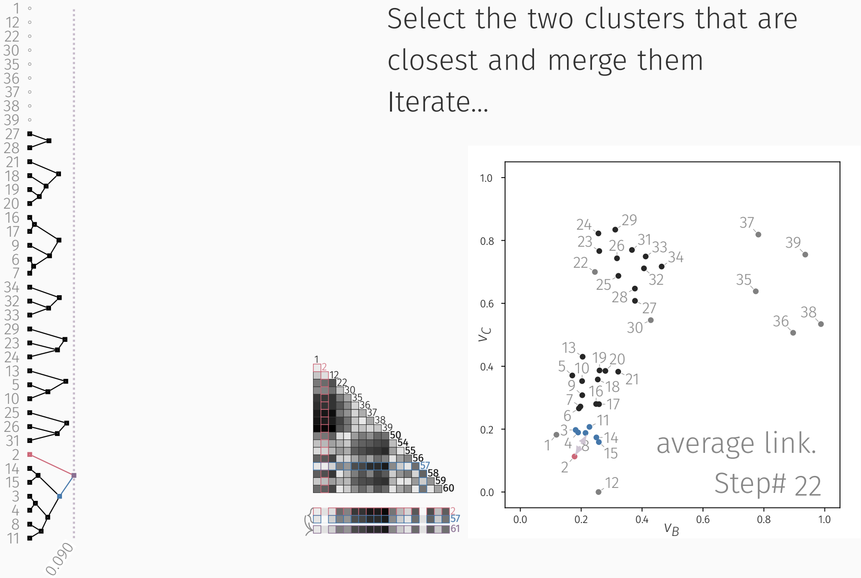
Iterate...

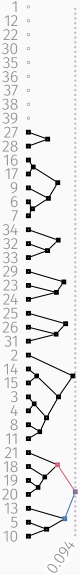


Agglomerative clustering with average linkage

Select the two clusters that are closest and merge them

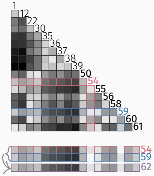
Iterate...

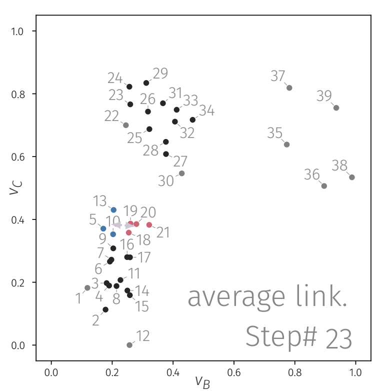


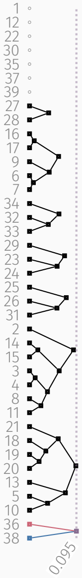


Select the two clusters that are closest and merge them

Iterate...

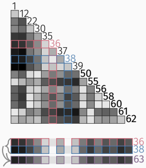


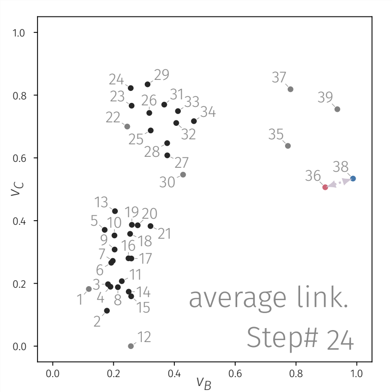


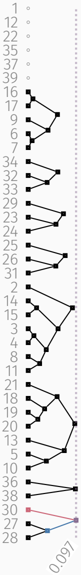


Select the two clusters that are closest and merge them

Iterate…

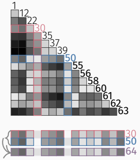


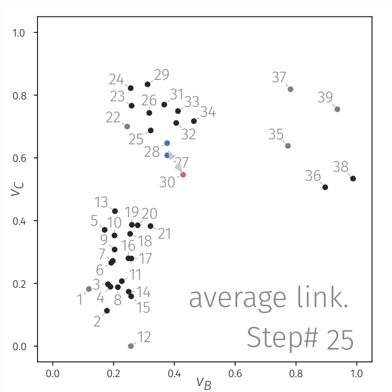




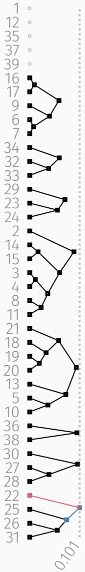
Select the two clusters that are closest and merge them

Iterate...



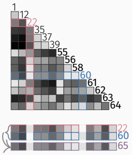


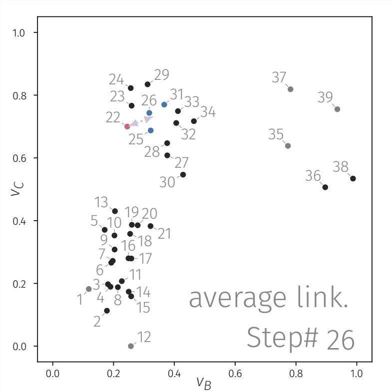
Agglomerative clustering with average linkage



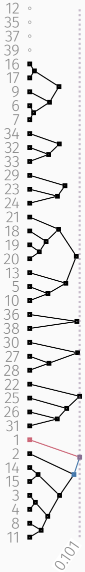
Select the two clusters that are closest and merge them

Iterate...



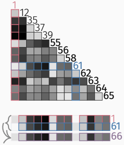


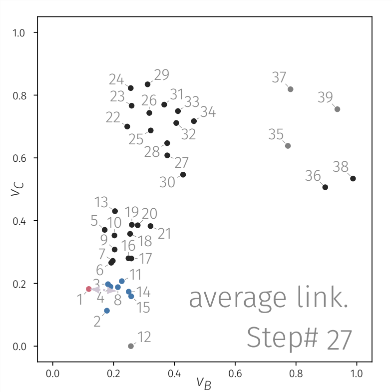
Agglomerative clustering with average linkage



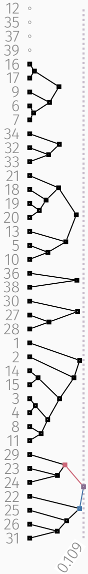
Select the two clusters that are closest and merge them

Iterate...



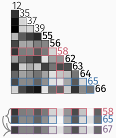


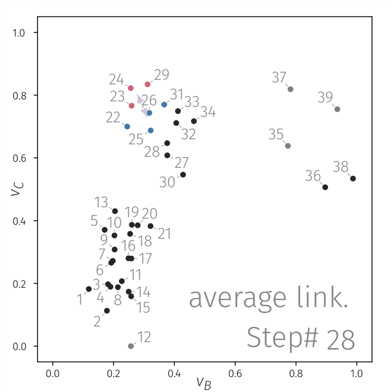
Agglomerative clustering with average linkage

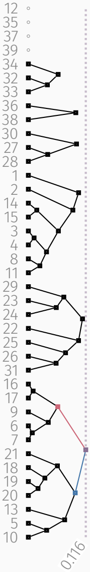


Select the two clusters that are closest and merge them

Iterate...

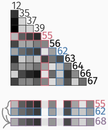


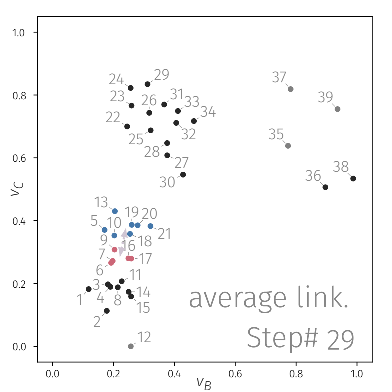




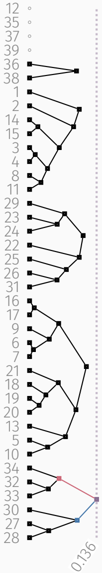
Select the two clusters that are closest and merge them

Iterate...



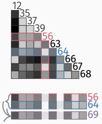


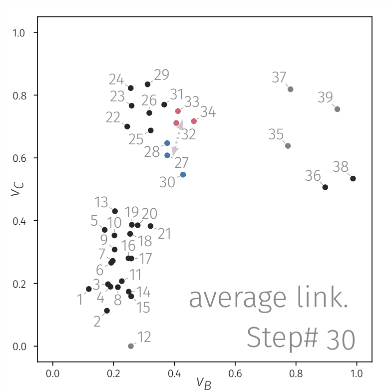
Agglomerative clustering with average linkage



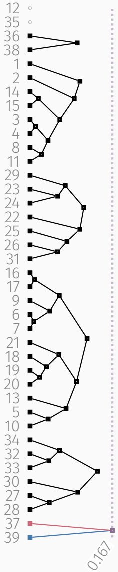
Select the two clusters that are closest and merge them

Iterate...



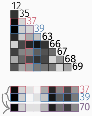


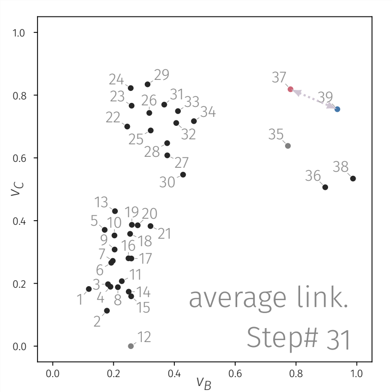
Agglomerative clustering with average linkage

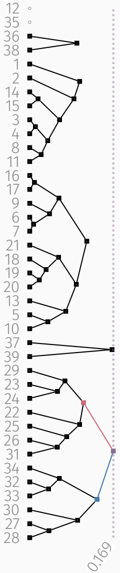


Select the two clusters that are closest and merge them

Iterate...

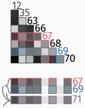


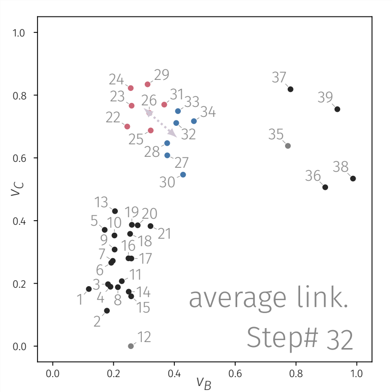




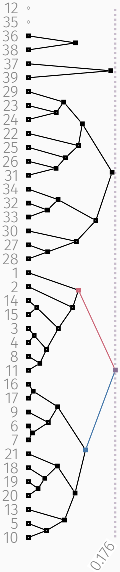
Select the two clusters that are closest and merge them

Iterate...



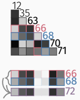


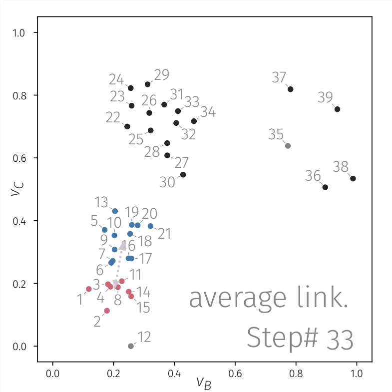
Agglomerative clustering with average linkage



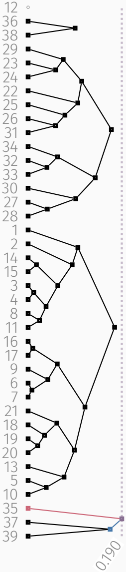
Select the two clusters that are closest and merge them

Iterate...



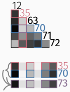


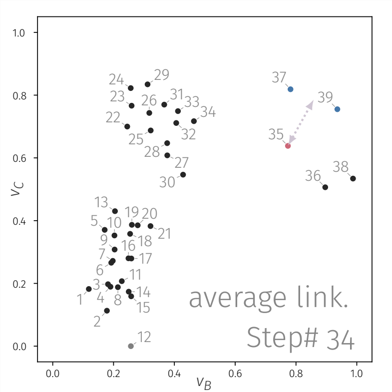
Agglomerative clustering with average linkage



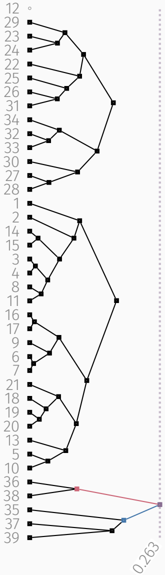
Select the two clusters that are closest and merge them

Iterate...



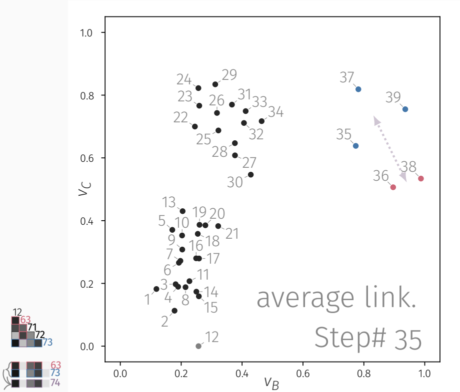


Agglomerative clustering with average linkage

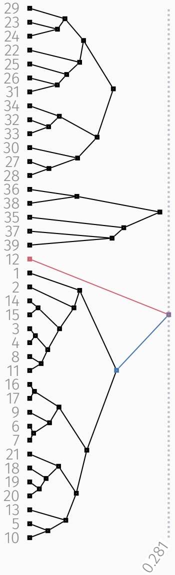


Select the two clusters that are closest and merge them

Iterate...

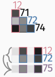


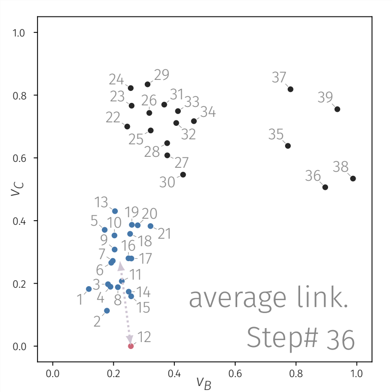
Agglomerative clustering with average linkage



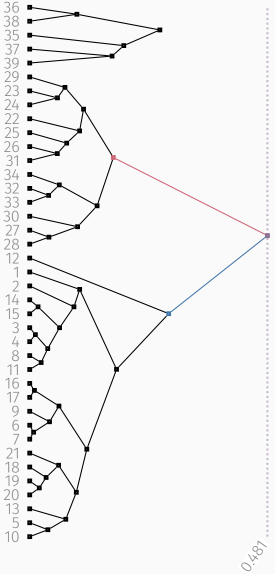
Select the two clusters that are closest and merge them

Iterate...



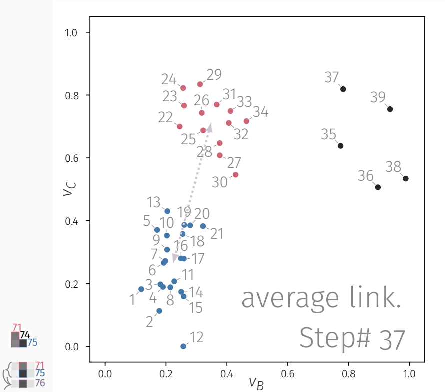


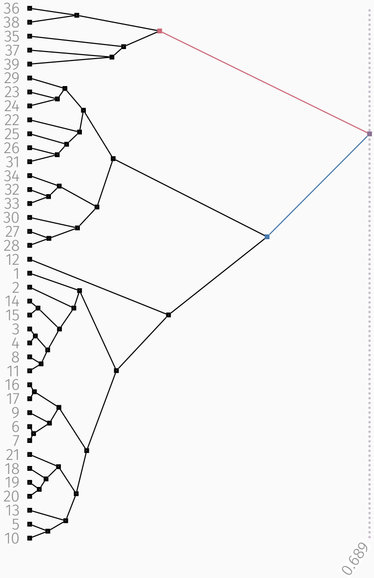
Agglomerative clustering with average linkage



Select the two clusters that are closest and merge them

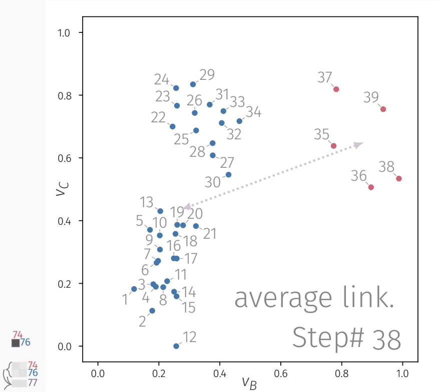
Iterate...

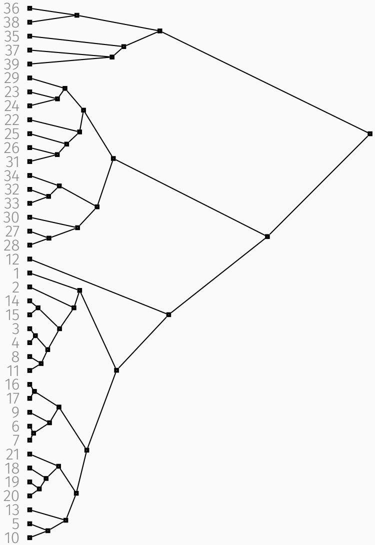




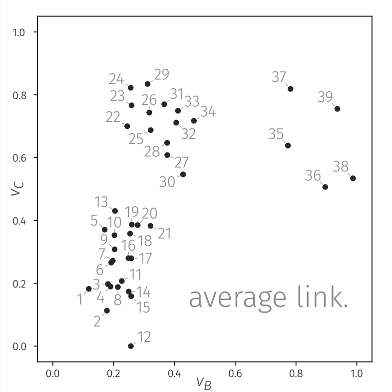
Select the two clusters that are closest and merge them

Iterate...

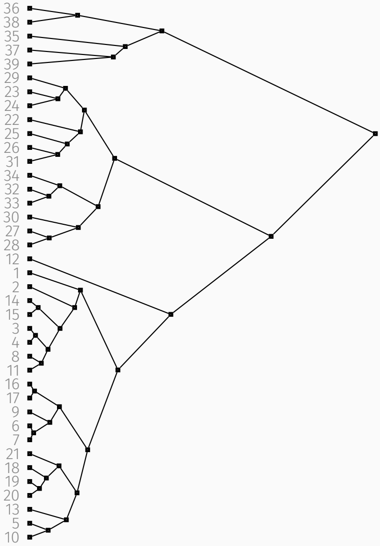




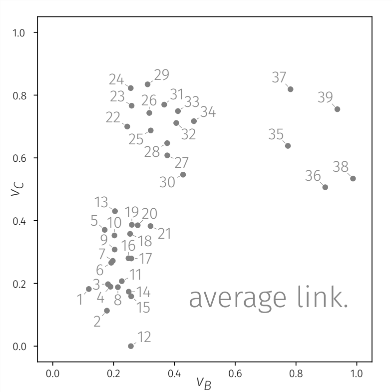
Until a single cluster containing all data points is obtained



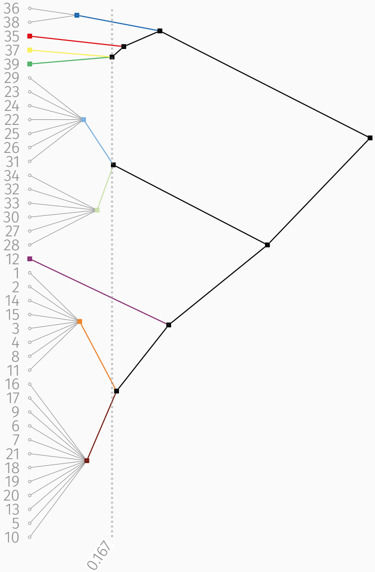
Agglomerative clustering with average linkage



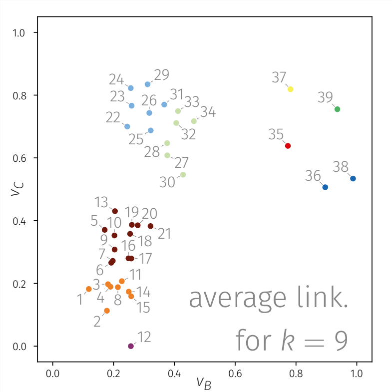
Truncate the hierarchy to obtain the desired number of clusters

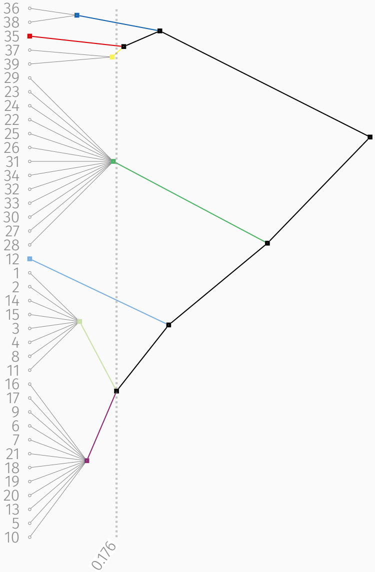


Agglomerative clustering with average linkage

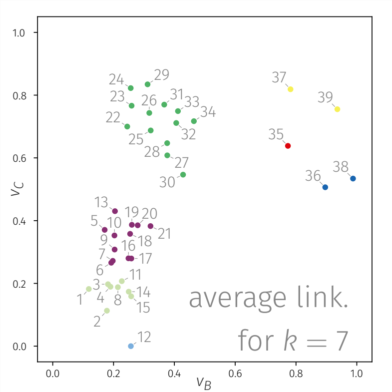


Truncate the hierarchy to obtain the desired number of clusters

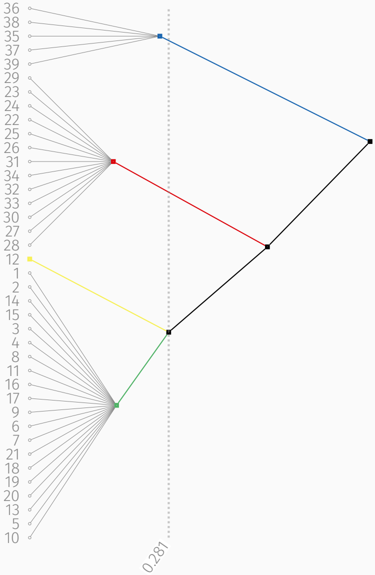




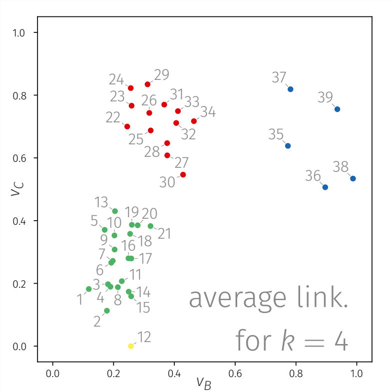
Truncate the hierarchy to obtain the desired number of clusters

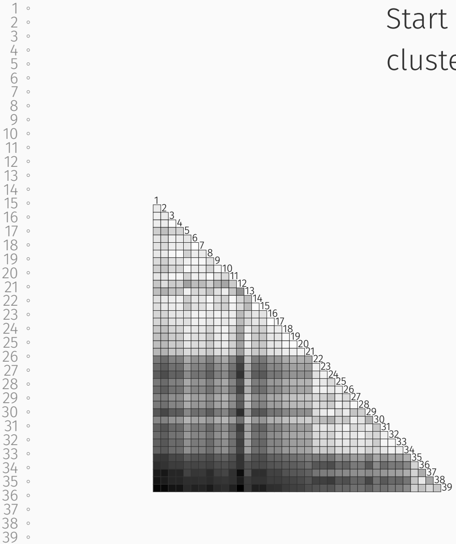


Agglomerative clustering with average linkage

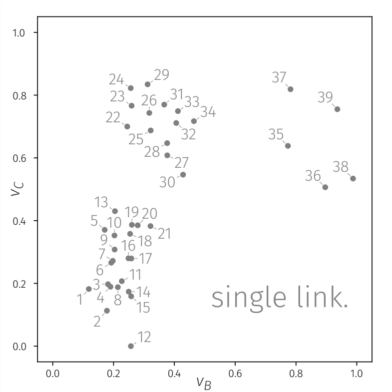


Truncate the hierarchy to obtain the desired number of clusters

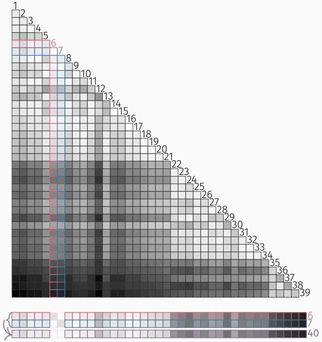




Start with each data point in its own cluster

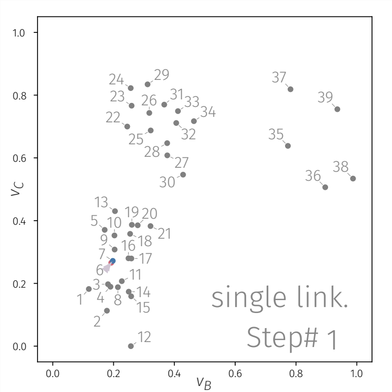


Agglomerative clustering with single linkage



Select the two clusters that are closest and merge them

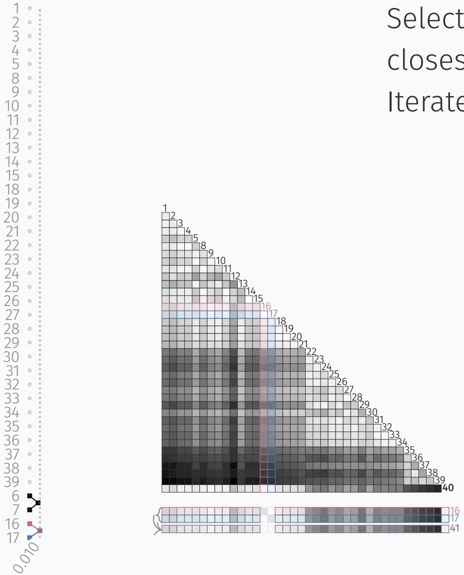
Iterate...

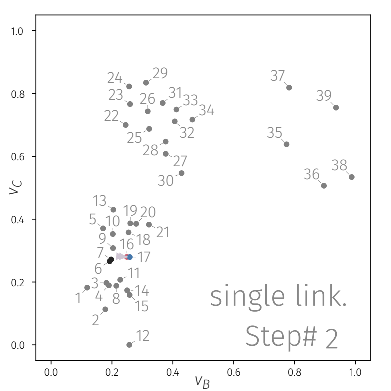


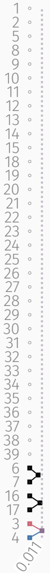
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate...

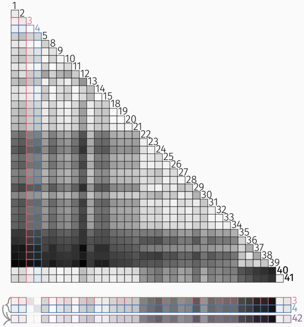


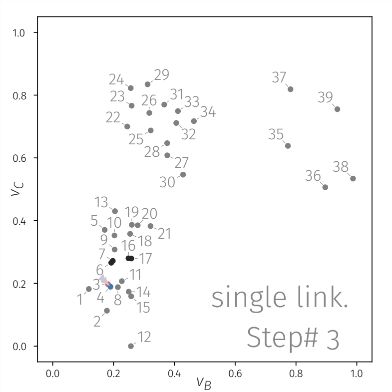




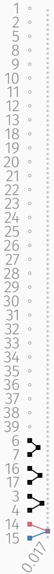
Select the two clusters that are closest and merge them

Iterate...



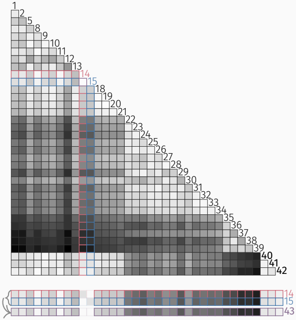


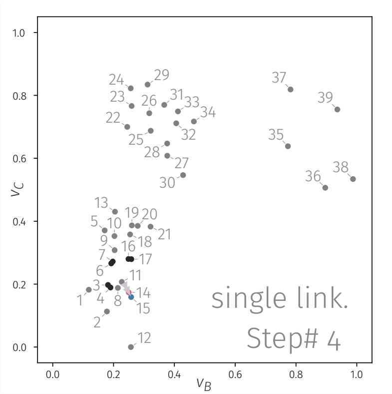
Agglomerative clustering with single linkage



Select the two clusters that are closest and merge them

Iterate...

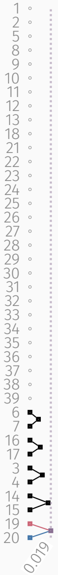


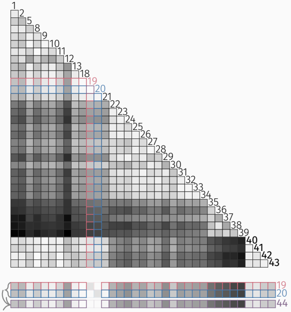


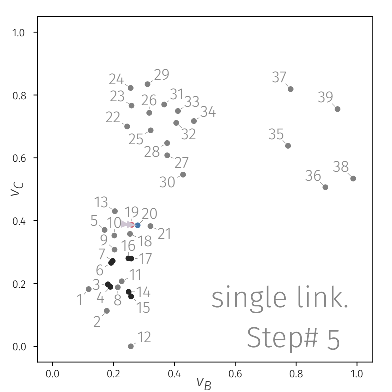
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

Iterate…

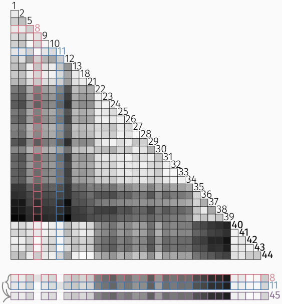


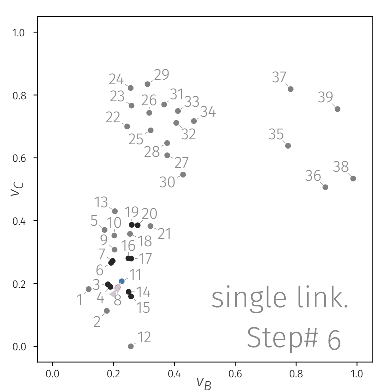




Select the two clusters that are closest and merge them

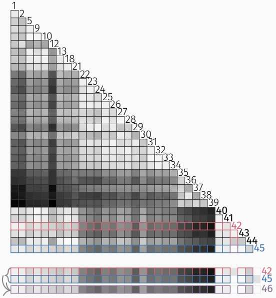
Iterate...

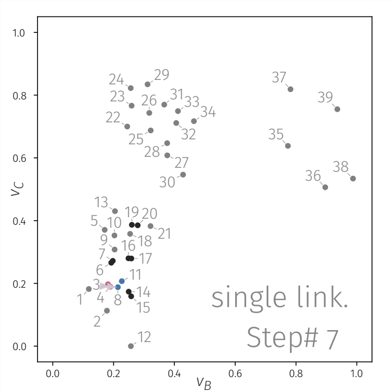




Select the two clusters that are closest and merge them

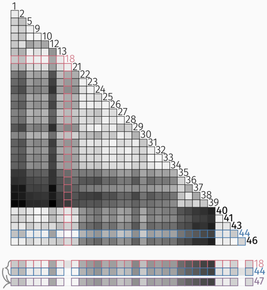
Iterate...

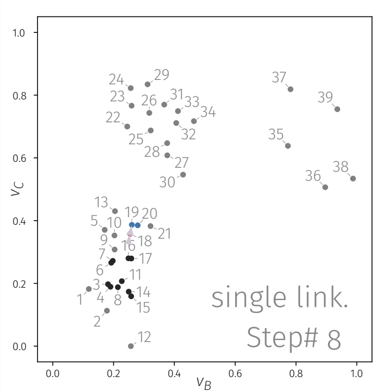




Select the two clusters that are closest and merge them

Iterate...

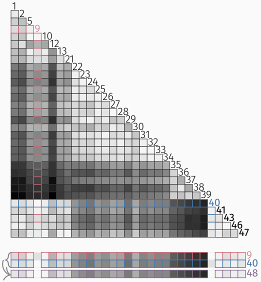


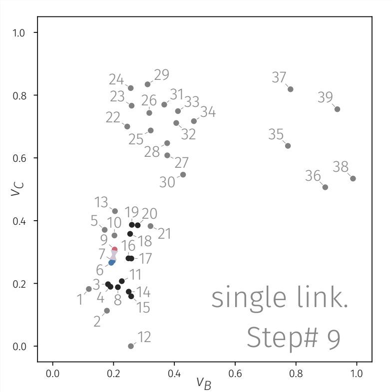


Agglomerative clustering with single linkage

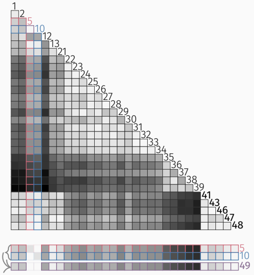
Select the two clusters that are closest and merge them

Iterate...



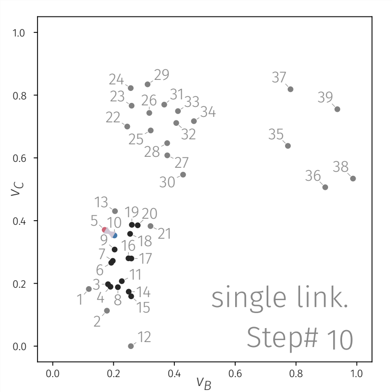


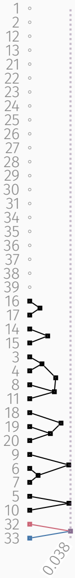
Agglomerative clustering with single linkage

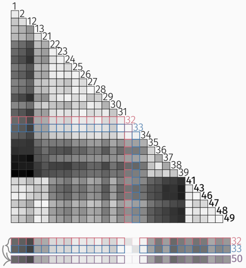


Select the two clusters that are closest and merge them

Iterate...

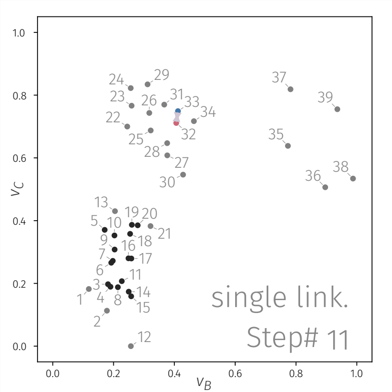




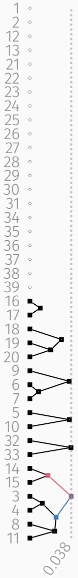


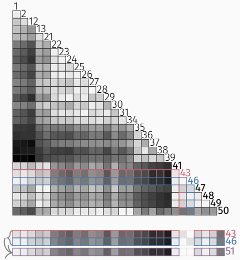
Select the two clusters that are closest and merge them

Iterate...



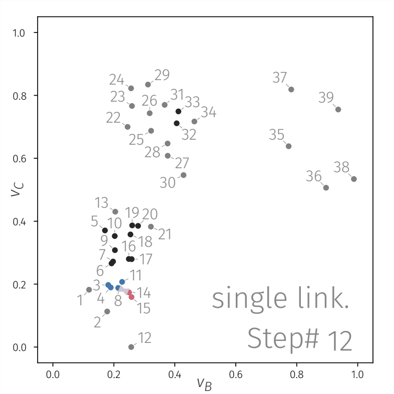
Agglomerative clustering with single linkage



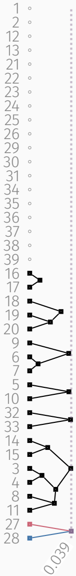


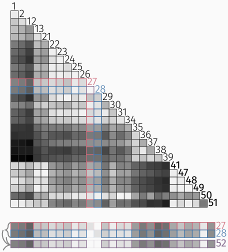
Select the two clusters that are closest and merge them

Iterate...



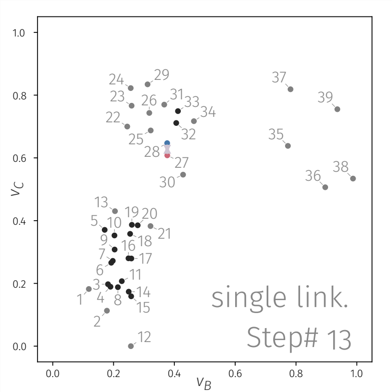
Agglomerative clustering with single linkage



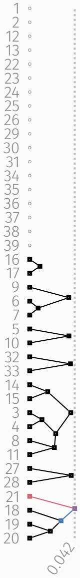


Select the two clusters that are closest and merge them

Iterate...

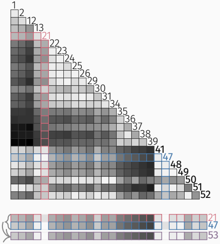


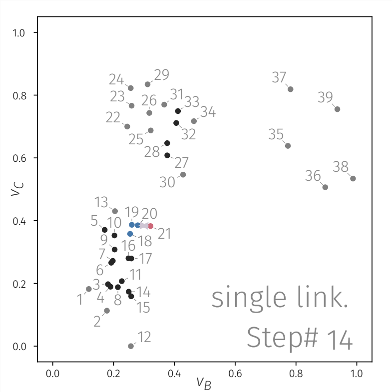
Agglomerative clustering with single linkage



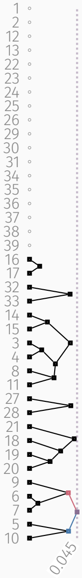
Select the two clusters that are closest and merge them

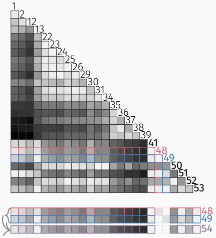
Iterate...





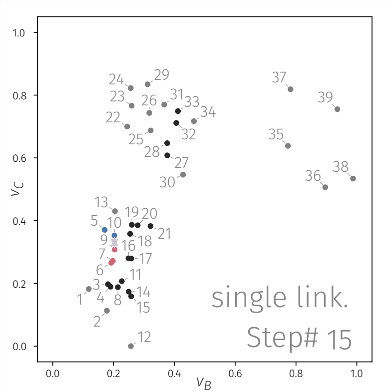
Agglomerative clustering with single linkage



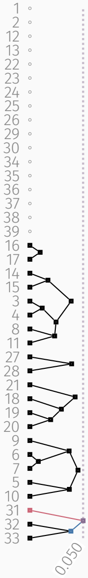


Select the two clusters that are closest and merge them

Iterate...

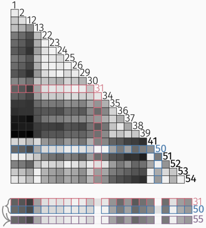


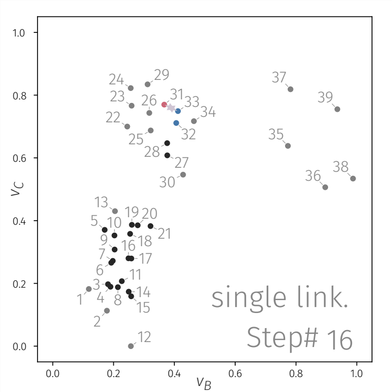
Agglomerative clustering with single linkage



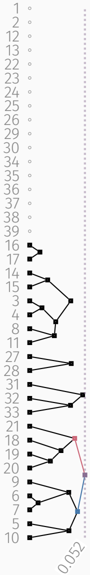
Select the two clusters that are closest and merge them

Iterate…



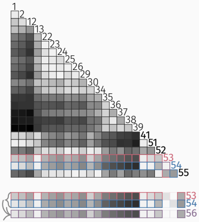


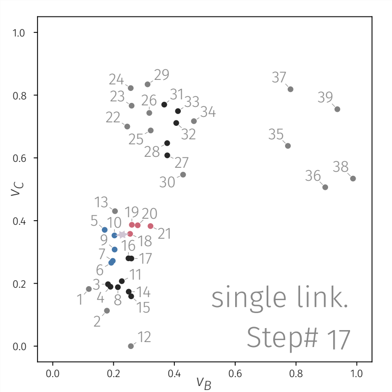
Agglomerative clustering with single linkage



Select the two clusters that are closest and merge them

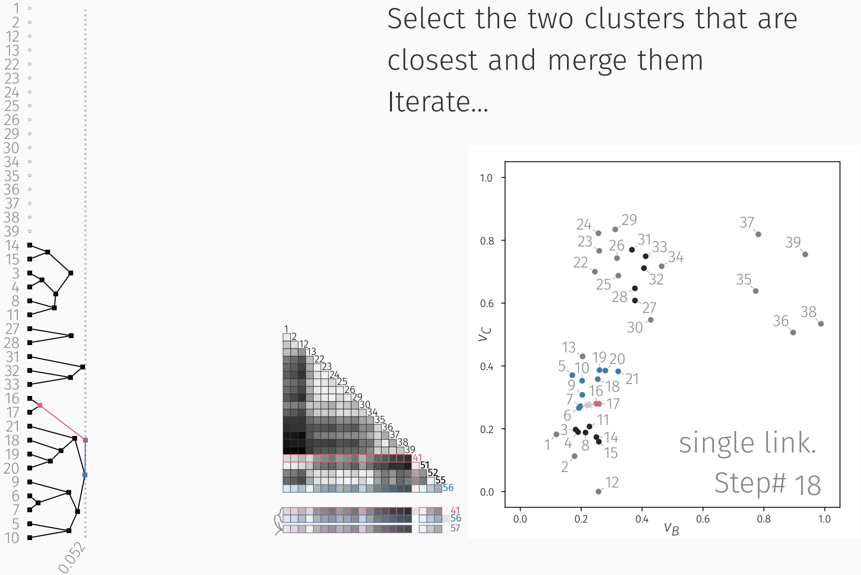
Iterate...

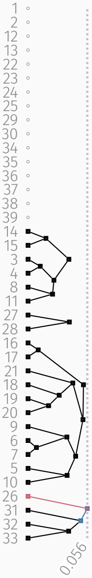




Select the two clusters that are closest and merge them

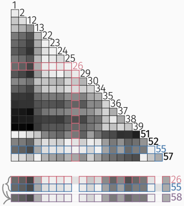
Iterate...

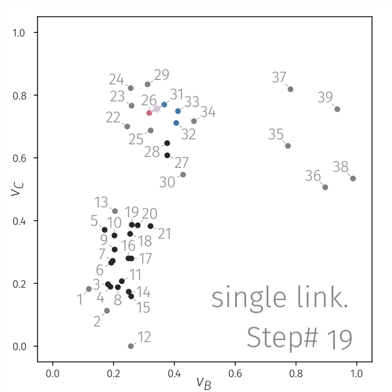




Select the two clusters that are closest and merge them

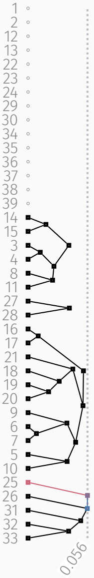
Iterate...

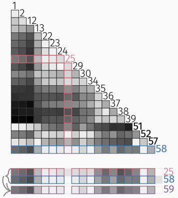


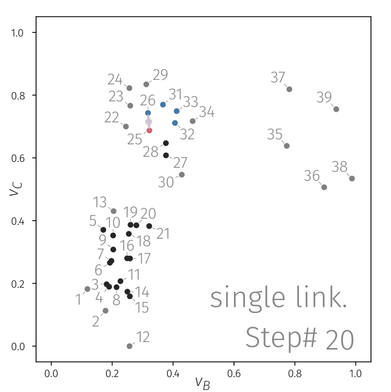


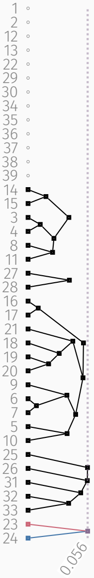
Select the two clusters that are closest and merge them

Iterate...



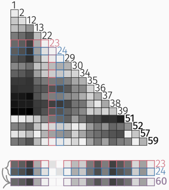


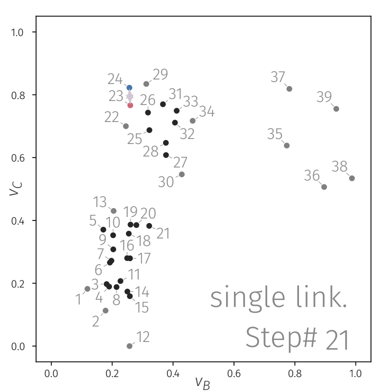




Select the two clusters that are closest and merge them

Iterate...

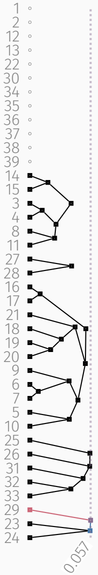


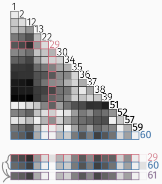


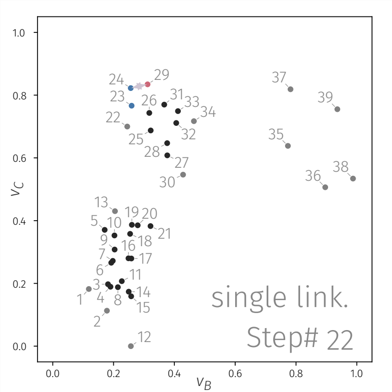
Agglomerative clustering with single linkage

Select the two clusters that are closest and merge them

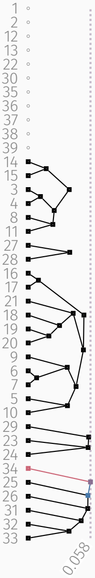
Iterate...





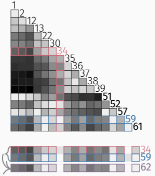


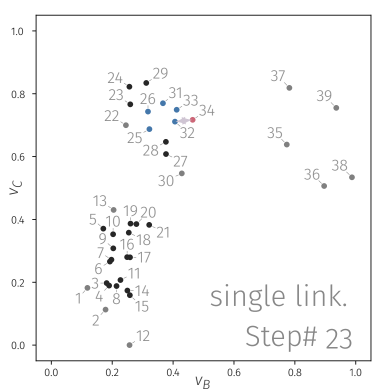
Agglomerative clustering with single linkage



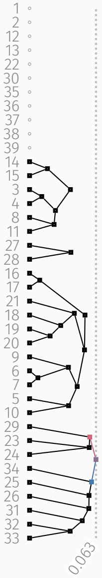
Select the two clusters that are closest and merge them

Iterate...



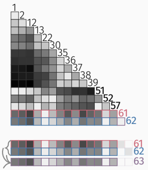


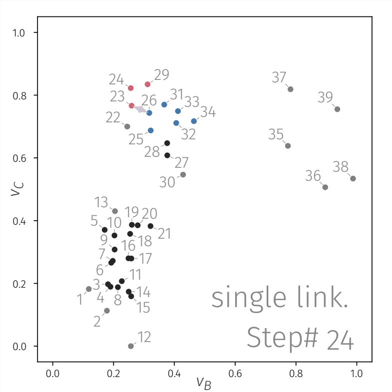
Agglomerative clustering with single linkage

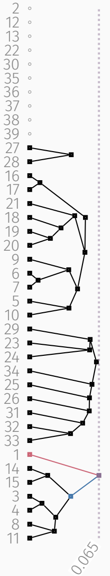


Select the two clusters that are closest and merge them

Iterate…

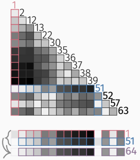


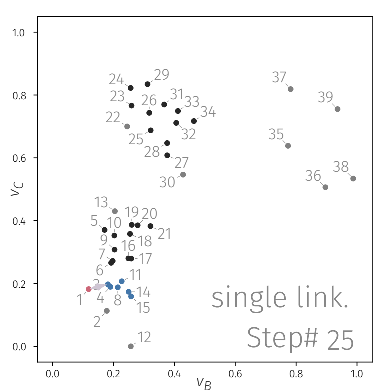




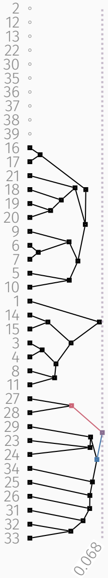
Select the two clusters that are closest and merge them

Iterate...



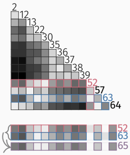


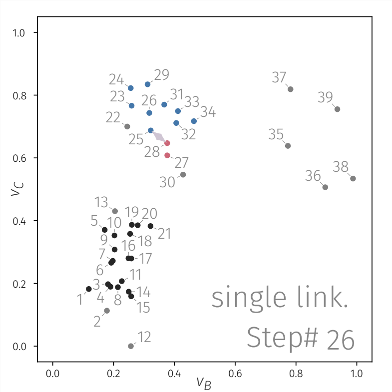
Agglomerative clustering with single linkage

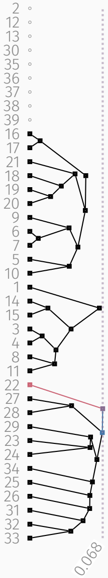


Select the two clusters that are closest and merge them

Iterate...

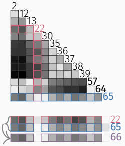


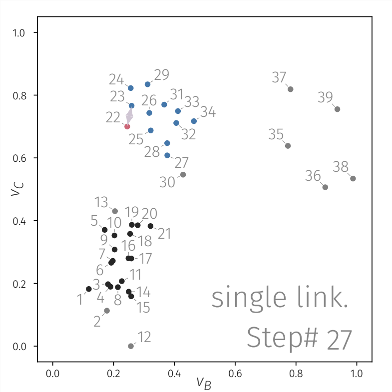




Select the two clusters that are closest and merge them

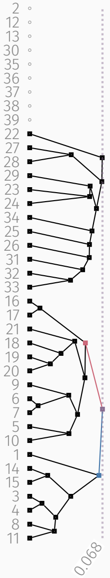
Iterate...

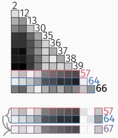


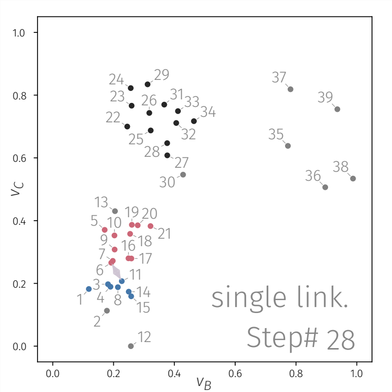


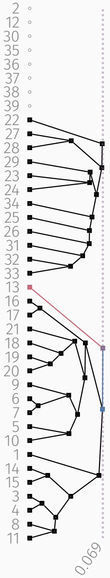
Select the two clusters that are closest and merge them

Iterate...



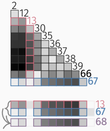


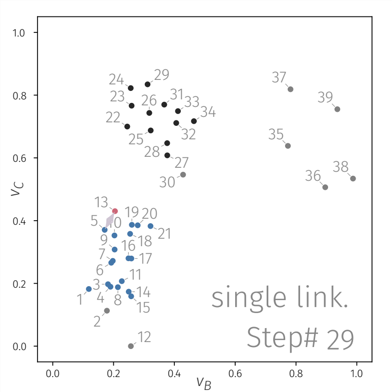


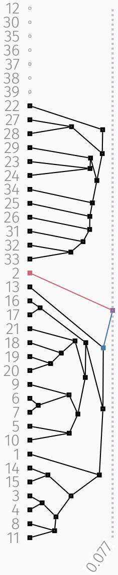


Select the two clusters that are closest and merge them

Iterate...

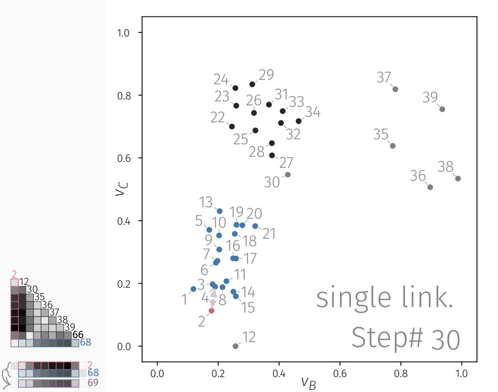




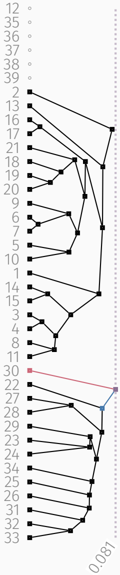


Select the two clusters that are closest and merge them

Iterate...

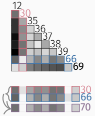


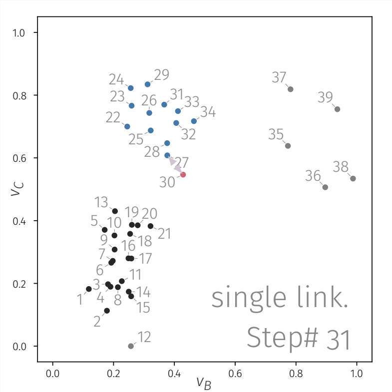
Agglomerative clustering with single linkage

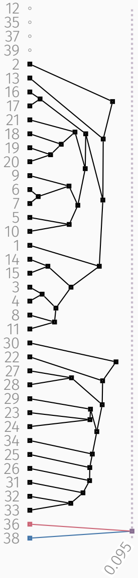


Select the two clusters that are closest and merge them

Iterate...

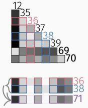


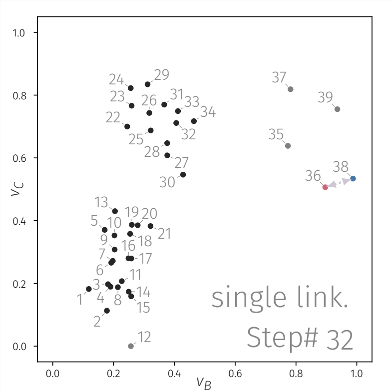


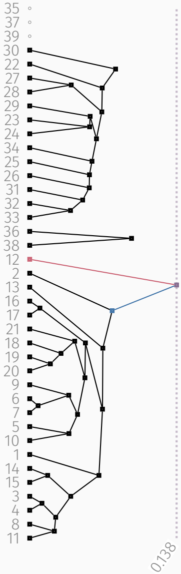


Select the two clusters that are closest and merge them

Iterate...

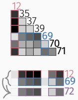


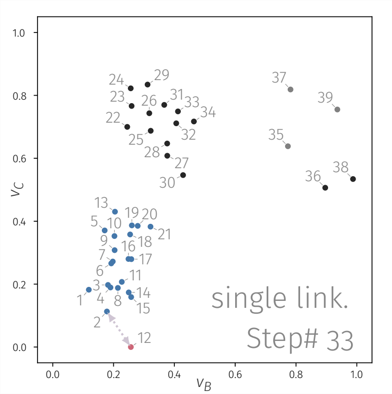


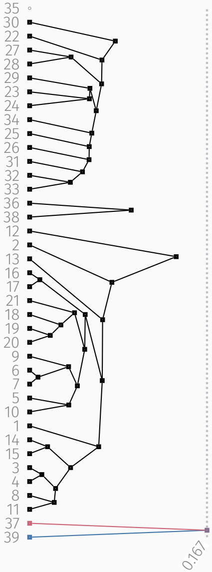


Select the two clusters that are closest and merge them

Iterate...

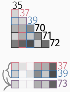


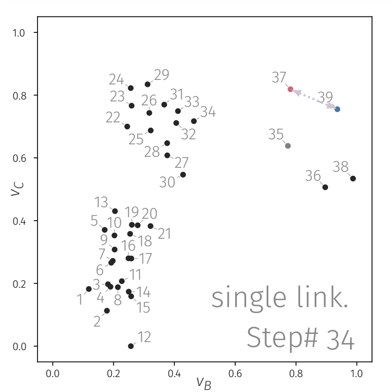




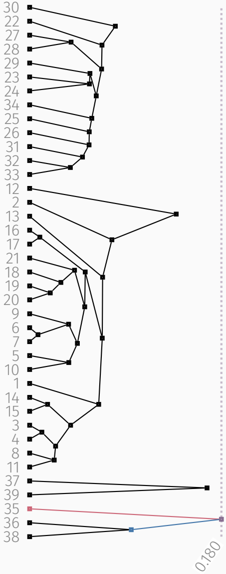
Select the two clusters that are closest and merge them

Iterate...



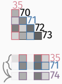


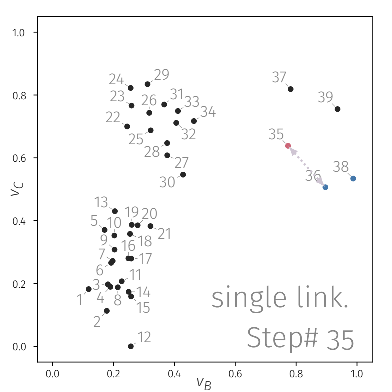
Agglomerative clustering with single linkage

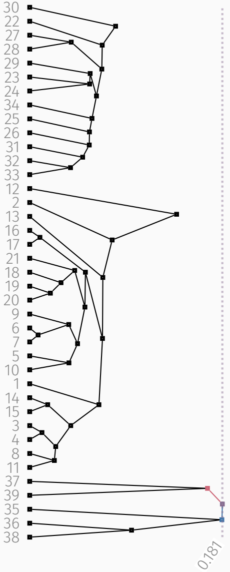


Select the two clusters that are closest and merge them

Iterate…

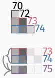


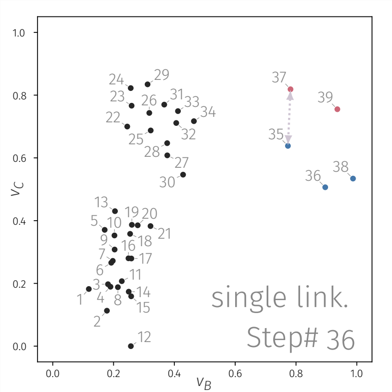




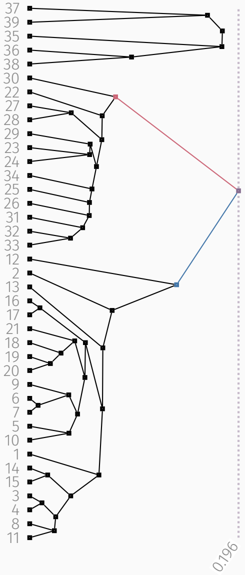
Select the two clusters that are closest and merge them

Iterate...



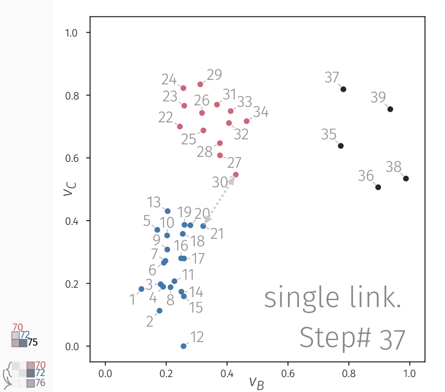


Agglomerative clustering with single linkage

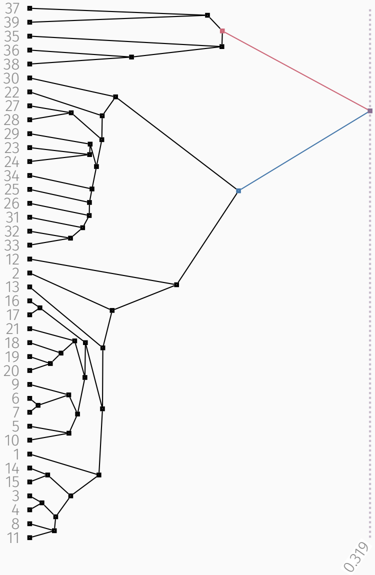


Select the two clusters that are closest and merge them

Iterate…

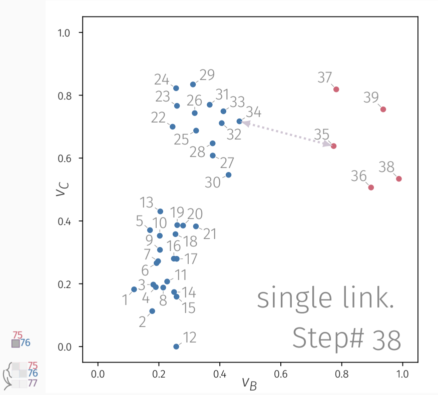


Agglomerative clustering with single linkage

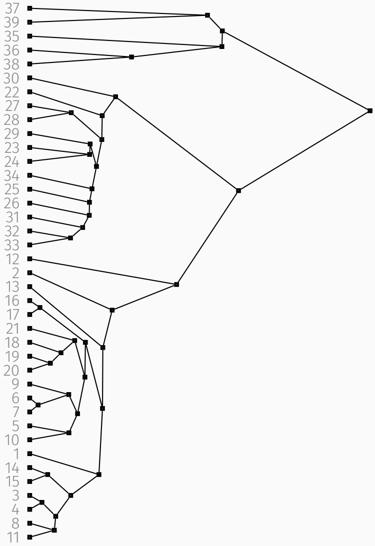


Select the two clusters that are closest and merge them

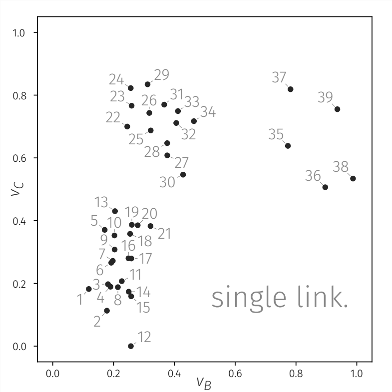
Iterate...



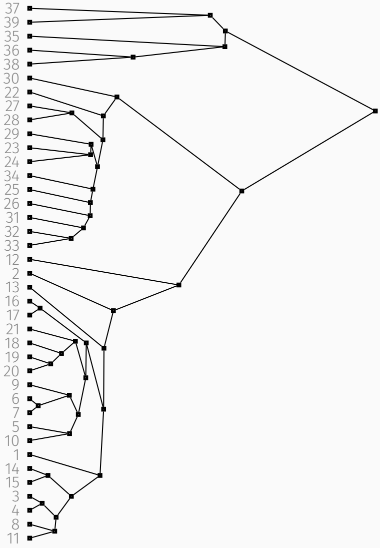
Agglomerative clustering with single linkage



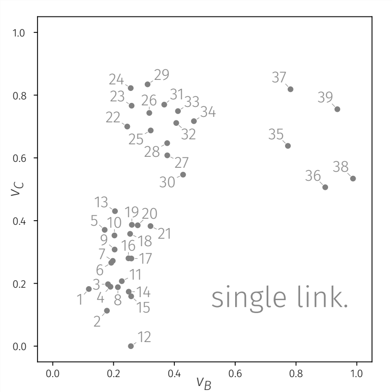
Until a single cluster containing all data points is obtained

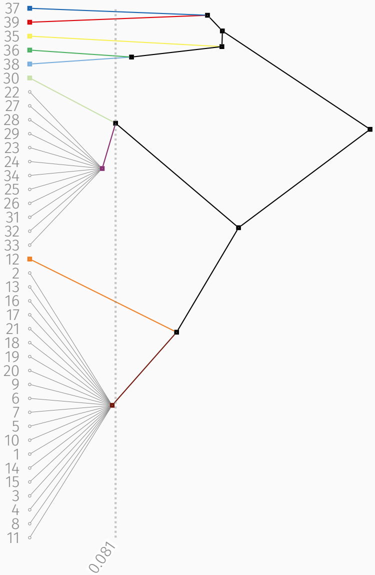


Agglomerative clustering with single linkage

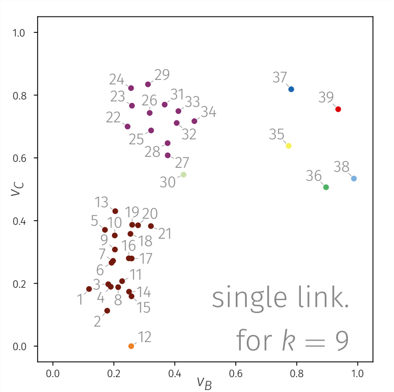


Truncate the hierarchy to obtain the desired number of clusters

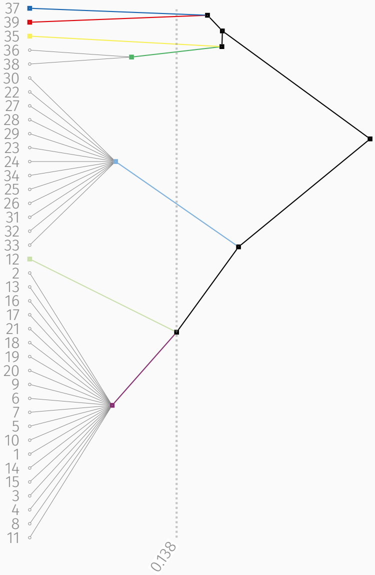




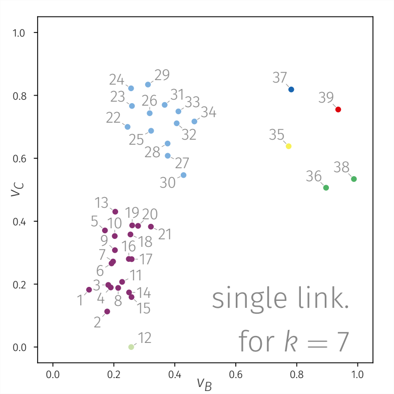
Truncate the hierarchy to obtain the desired number of clusters



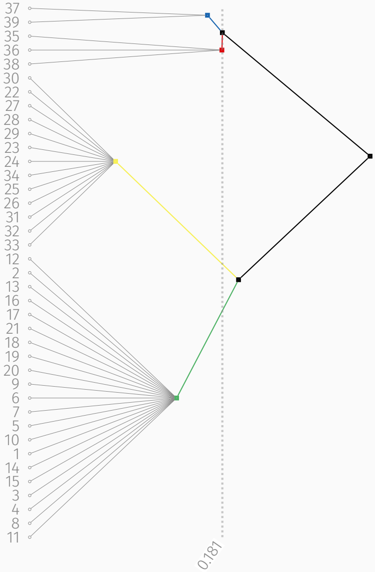
Agglomerative clustering with single linkage



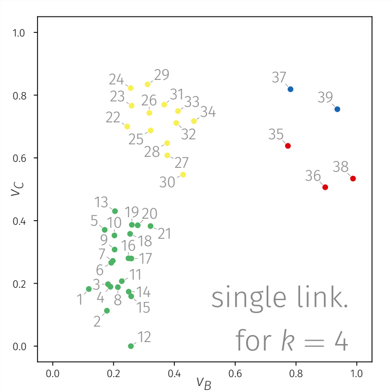
Truncate the hierarchy to obtain the desired number of clusters



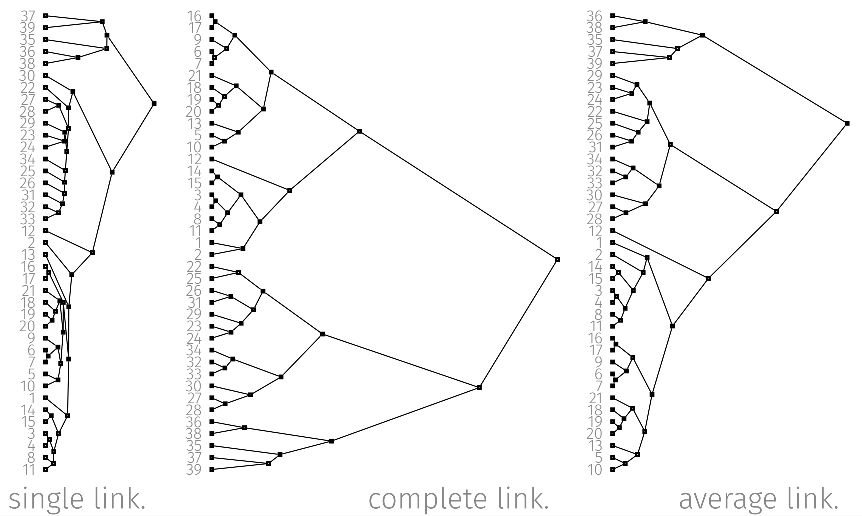
Agglomerative clustering with single linkage



Truncate the hierarchy to obtain the desired number of clusters



Different linkage functions produce different cluster hierarchies



Density-based algorithms

Density-based algorithms aim to identify *connected dense areas* of the data as the clusters

Data points that lie in sparse areas of the data might not be assigned to any cluster

Density-based algorithms

Density-based algorithms aim to identify *connected dense areas* of the data as the clusters

Data points that lie in sparse areas of the data might not be assigned to any cluster

DBSCAN is a popular example of a density-based algorithm

The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into three categories, depending on their neighborhood

For chosen parameters ϵ and τ

core points have at least τ points within a radius ϵ

border points have less than τ points within a radius ϵ, but at least one is a core point

noise points have less than τ points within a radius ϵ, and none is a core point

The DBSCAN algorithm

For chosen parameters ϵ and τ

Let $N(\mathbf{x}, \epsilon)$ denote the set of points within a radius ϵ of point $\mathbf{x}$ (including the point itself), that is

$$N(\mathbf{x}, \epsilon) = \{\mathbf{x}' \in \mathcal{D}, \mathrm{d}(\mathbf{x}, \mathbf{x}') \leq \epsilon\}$$

Let D_c, D_b and D_n denote the sets of **core**, **border** and **noise** points respectively, then

$$D_c = \{\mathbf{x} \in \mathcal{D}, |N(\mathbf{x}, \epsilon)| \geq \tau\}$$
$$D_b = \{\mathbf{x} \in \mathcal{D}, |N(\mathbf{x}, \epsilon)| < \tau \text{ and } N(\mathbf{x}, \epsilon) \cap D_c \neq \emptyset\}$$
$$D_n = \{\mathbf{x} \in \mathcal{D}, |N(\mathbf{x}, \epsilon)| < \tau \text{ and } N(\mathbf{x}, \epsilon) \cap D_c = \emptyset\}$$

The DBSCAN algorithm

The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into **core**, **border** and **noise** points

(ii) construct a graph with core points as the vertices and with an edge between two vertices if the corresponding points are within a radius ϵ of each other, find connected components from this graph

The DBSCAN algorithm

The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into **core**, **border** and **noise** points

(ii) find connected components from the graph of **core** points

(iii) assign **border** points to the component they are most strongly connected to

The DBSCAN algorithm

The DBSCAN algorithm proceeds in three main steps

(i) divide the data points into **core**, **border** and **noise** points

(ii) find connected components from the graph of **core** points

(iii) assign **border** points to the most relevant component

→ The connected components are returned as the clusters
Noise points are not assigned to any cluster

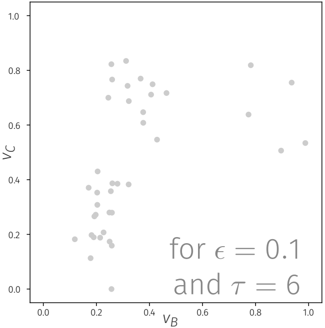
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



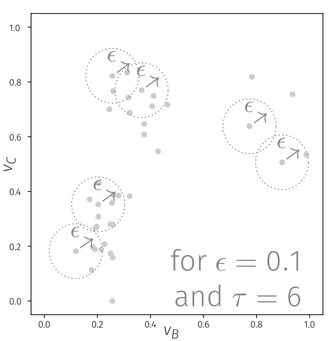
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



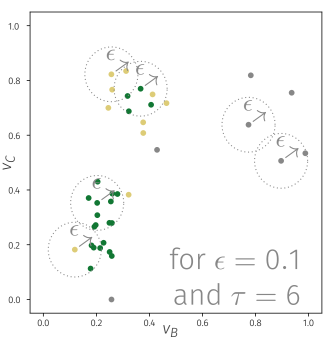
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



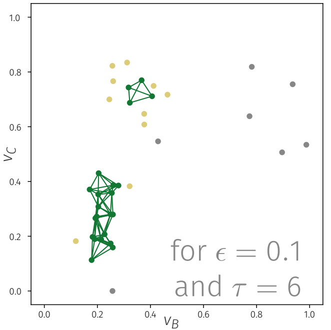
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



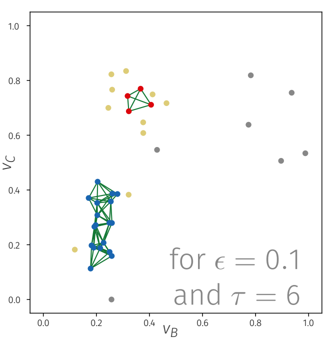
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



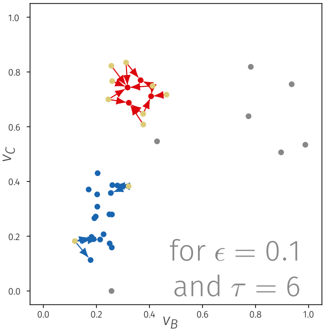
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

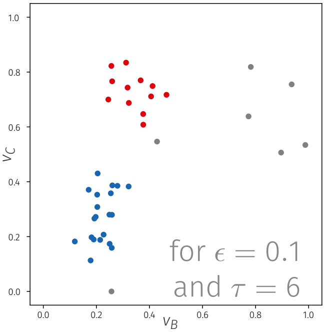
Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n
Construct a graph with D_c as vertices,
and find the connected components
Assign each point in D_b to the most relevant component



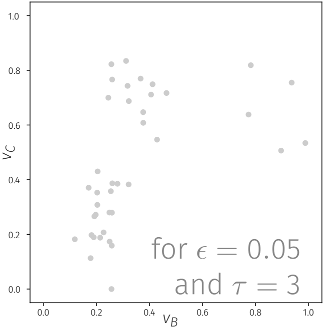
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



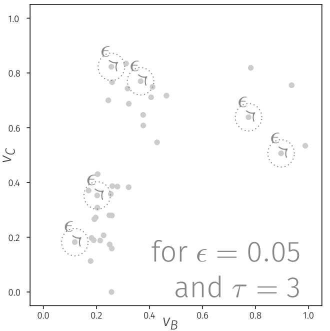
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component

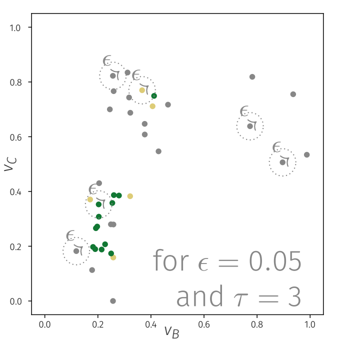


Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

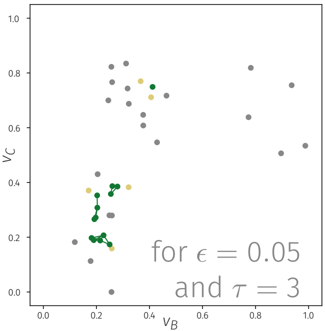
Assign each point in D_b to the most relevant component



Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,
and find the connected components

Assign each point in D_b to the most relevant component

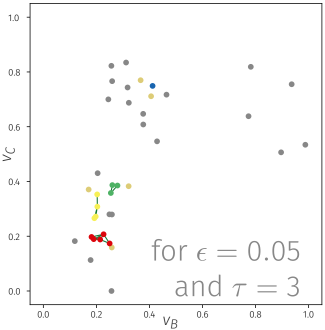


Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



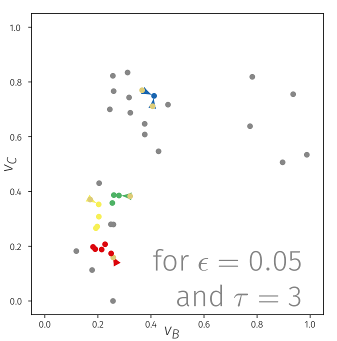
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

Assign each point in D_b to the most relevant component



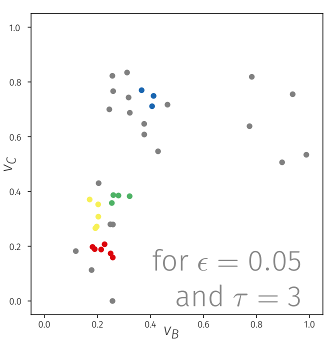
The DBSCAN algorithm

Given ϵ and τ, partition $\mathcal{D}$ into D_c, D_b and D_n

Construct a graph with D_c as vertices,

and find the connected components

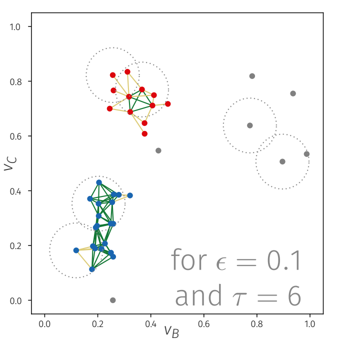
Assign each point in D_b to the most relevant component

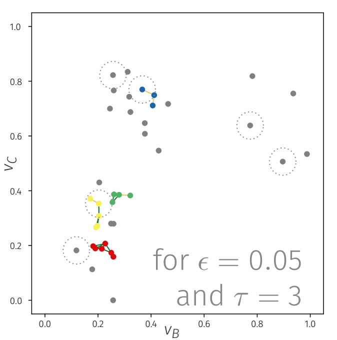


The DBSCAN algorithm

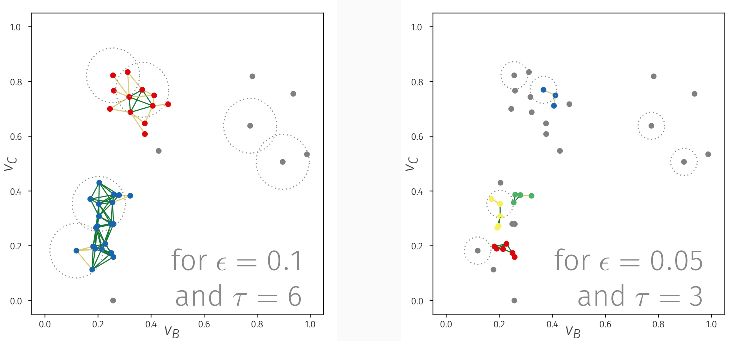
Unlike for instance *k*-means, DBSCAN is not limited to *spherical clusters* but can detect clusters of arbitrary shapes

On the other hand, it is limited to detecting clusters of *similar densities*





DBSCAN does not require to provide the *number of clusters* as input parameter, it is set implicitly based on the connectivity of the graph



The DBSCAN algorithm

DBSCAN does not require to provide the *number of clusters* as input parameter, it is set implicitly based on the connectivity of the graph

On the other hand, DBSCAN requires to set parameters ϵ and τ
While their meaning is relatively intuitive, that is, a smaller radius ϵ and a greater number of neighbors τ increase the density needed for an area to be considered a cluster,
they might be difficult to adjust for a specific data set

Evaluation

Clustering evaluation

Given a dataset, we can obtain various clusterings, by applying different methods and using different parameter settings

We need to quantify the quality of clusterings, in order to

- measure the effectiveness and tune the parameters of a particular algorithm
- compare and select clusterings

Clustering evaluation

Clustering is defined as an *unsupervised* task, and often there is *no ground truth clustering* to compare against

Hence we often need to rely on *internal* validation criteria

Internal validation criteria

We often need to rely on *internal* validation criteria, such as

sum of square distances to centroids determine a centroid for each cluster (or use the representative, for representative-based methods) and compute the sum of square distances from every point to the associated centroid

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, let $\boldsymbol{r}^{(u)}$ denote the centroid of cluster C_u, then

$$\mathrm{SSDC}(\mathcal{C}) = \sum_{C_u \in \mathcal{C}} \sum_{\boldsymbol{x} \in C_u} \mathrm{d}(\boldsymbol{x}, \boldsymbol{r}^{(u)})^2$$

Smaller values indicate more cohesive clusters

Internal validation criteria

We often need to rely on *internal* validation criteria, such as

Intra-cluster vs. inter-cluster distance ratio compare the distances between pairs of points in the same vs. in different clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, let

$$D(C_u) = \sum_{\substack{(x,x')\in C_u \times C_u \\ x \neq x'}} \mathrm{d}(x, x') \qquad D_{\text{intra}} = \sum_{C_u \in \mathcal{C}} D(C_u) \qquad P_{\text{intra}} = \sum_{C_u \in \mathcal{C}} |C_u| \cdot (|C_u| - 1)$$

$$D(C_u, C_v) = \sum_{(x,x')\in C_u \times C_v} \mathrm{d}(x, x') \qquad D_{\text{inter}} = \sum_{\substack{(C_u,C_v)\in \mathcal{C} \times \mathcal{C} \\ C_u \neq C_v}} D(C_u, C_v) \qquad P_{\text{inter}} = \sum_{\substack{(C_u,C_v)\in \mathcal{C} \times \mathcal{C} \\ C_u \neq C_v}} |C_u| \cdot |C_v|$$

then, $\mathrm{DR}(\mathcal{C}) = \dfrac{D_{\text{intra}}/P_{\text{intra}}}{D_{\text{inter}}/P_{\text{inter}}}$

Internal validation criteria

We often need to rely on *internal* validation criteria, such as

Intra-cluster vs. inter-cluster distance ratio compare the distances between pairs of points in the same vs. in different clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$,

$$\mathrm{DR}(\mathcal{C}) = \frac{D_{\text{intra}}/P_{\text{intra}}}{D_{\text{inter}}/P_{\text{inter}}}$$

D_{intra} and D_{inter} can be computed for P_{intra} and P_{inter} pairs of points sampled at random rather than from all pairs, especially for large datasets

Smaller values indicate more cohesive clusters

Internal validation criteria

We often need to rely on *internal* validation criteria, such as

Silhouette coefficient compare for each point the average distance to other points within the same cluster and average distance to points in other clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ and data point $\boldsymbol{x} \in C_j$, let

$$D_S(x) = \sum_{\substack{x' \in C_j \\ x' \neq x}} \frac{d(x, x')}{|C_j| - 1} \quad \text{and} \quad D_O(x) = \min_{\substack{C \in \mathcal{C} \\ C \neq C_j}} \sum_{x' \in C} \frac{d(x, x')}{|C|}$$

then, the silhouette coefficient for point $\boldsymbol{x}$ is

$$S(x) = \frac{D_O(x) - D_S(x)}{\max(D_O(x), D_S(x))}$$

We often need to rely on *internal* validation criteria, such as

Silhouette coefficient compare for each point the average distance to other points within the same cluster and average distance to points in other clusters

For a clustering $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, the overall silhouette coefficient is the average of point-specific coefficients

$$S(\mathcal{C}) = \sum_{x \in \mathcal{D}} \frac{S(x)}{|\mathcal{D}|}$$

The silhouette coefficient takes values in $[-1, 1]$, with large positive values indicating more clearly separated clusters whereas negative values indicate more blending

Internal validation criteria

We often need to rely on *internal* validation criteria, such as

- sum of square distances to centroids
- Intra-cluster vs. inter-cluster distance ratio
- Silhouette coefficient

These measures are biased towards algorithms that optimize a similar criterion

Internal validation criteria

We often need to rely on *internal* validation criteria, such as

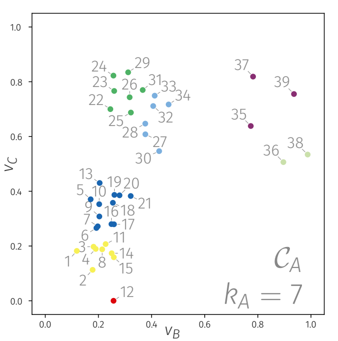
- sum of square distances to centroids
- Intra-cluster vs. inter-cluster distance ratio
- Silhouette coefficient

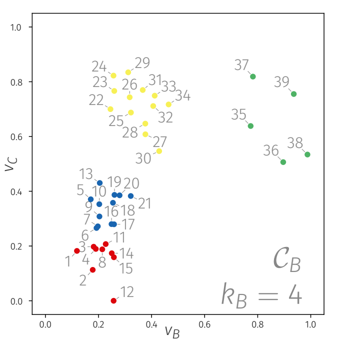
These measures can be used to select values for the parameters, such as the number of clusters k

One might look at how the value of the validation measure evolves when varying the value of a parameter and look for an inflection point

! Caution is required due to the inherent flaws of the measures

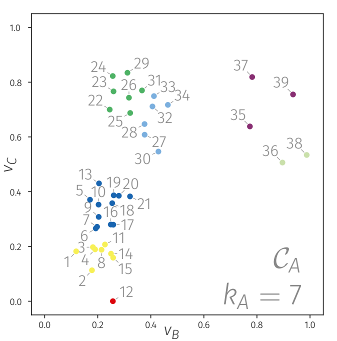
We might want to compare the specific assignments of data points corresponding to two different clusterings, to evaluate how much they agree

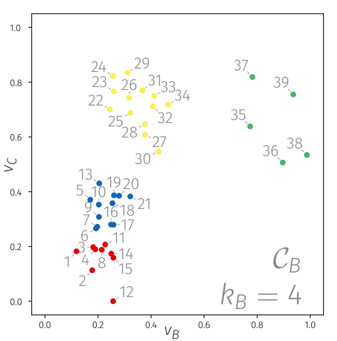




Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters, respectively





! Note that k_A and k_B might be different, and that no mapping between clusters of $\mathcal{C}_A$ and of $\mathcal{C}_B$ is assumed a priori

Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters

Given a data point, let a (respectively b) be the index of the cluster to which it is assigned in clustering $\mathcal{C}_A$ (respectively $\mathcal{C}_B$)

A data point might be assigned to

the first cluster of $\mathcal{C}_A$ *and the first cluster of* $\mathcal{C}_B$*, i.e.* $(a = 1, b = 1)$
the first cluster of $\mathcal{C}_A$ *and the second cluster of* $\mathcal{C}_B$*, i.e.* $(a = 1, b = 2)$

$\vdots$

the first cluster of $\mathcal{C}_A$ *and the last cluster of* $\mathcal{C}_B$*, i.e.* $(a = 1, b = k_B)$

$\vdots$

the last cluster of $\mathcal{C}_A$ *and the last cluster of* $\mathcal{C}_B$*, i.e.* $(a = k_A, b = k_B)$

There are $k_A \cdot k_B$ distinct possible outcomes, i.e. pairs (a, b)

Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters

Given a data point, let a (respectively b) be the index of the cluster to which it is assigned in clustering $\mathcal{C}_A$ (respectively $\mathcal{C}_B$)

There are $k_A \cdot k_B$ distinct possible outcomes, i.e. pairs (a, b)

Let $\#(a = i, b = j)$ denote the number of data points that belong to the i^{th} cluster of $\mathcal{C}_A$ and the j^{th} cluster of $\mathcal{C}_B$

Similarly, let $\#(a = i)$ and $\#(b = j)$ respectively denote the total number of data points in the i^{th} cluster of $\mathcal{C}_A$ and in the j^{th} cluster of $\mathcal{C}_B$

We assume the clusterings are partitions of the data set, that is

$$n = \sum_{i \in [1..k_A]} \#(a = i) = \sum_{j \in [1..k_B]} \#(b = j)$$

Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters

Let $\#(a = i, b = j)$ denote the number of data points that belong to the i^{th} cluster of $\mathcal{C}_A$ and the j^{th} cluster of $\mathcal{C}_B$

The assignment outcome of the pair of clusterings for the data set can be summarized in a $k_A \times k_B$ *contingency matrix*

		Clustering $\mathcal{C}_B$				
		$b = 1$	$\dots$	$b = j$	$\dots$	$b = k_B$
Clustering $\mathcal{C}_A$	$a = 1$	$\#(a = 1, b = 1)$		$\#(a = 1, b = j)$		$\#(a = 1, b = k_B)$
	$\vdots$					
	$a = i$	$\#(a = i, b = 1)$		$\#(a = i, b = j)$		$\#(a = i, b = k_B)$
	$\vdots$					
	$a = k_A$	$\#(a = k_A, b = 1)$		$\#(a = k_A, b = j)$		$\#(a = k_A, b = k_B)$

Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters

		Clustering $\mathcal{C}_B$				
		$b = 1$	$\dots$	$b = j$	$\dots$	$b = k_B$
Clustering $\mathcal{C}_A$	$a = 1$	$\#(a = 1, b = 1)$		$\#(a = 1, b = j)$		$\#(a = 1, b = k_B)$
	$\vdots$					
	$a = i$	$\#(a = i, b = 1)$		$\#(a = i, b = j)$		$\#(a = i, b = k_B)$
	$\vdots$					
	$a = k_A$	$\#(a = k_A, b = 1)$		$\#(a = k_A, b = j)$		$\#(a = k_A, b = k_B)$

For equal $k_A = k_B$, perfect agreement means that the clusterings are identical up to relabeling of the clusters, i.e. that the rows and columns of the contingency matrix can be reordered so that non-zero values appear only on the diagonal

Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters

		Clustering $\mathcal{C}_B$				
		$b = 1$	$\ldots$	$b = j$	$\ldots$	$b = k_B$
Clustering $\mathcal{C}_A$	$a = 1$	$\#(a = 1, b = 1)$		$\#(a = 1, b = j)$		$\#(a = 1, b = k_B)$
	$\vdots$					
	$a = i$	$\#(a = i, b = 1)$		$\#(a = i, b = j)$		$\#(a = i, b = k_B)$
	$\vdots$					
	$a = k_A$	$\#(a = k_A, b = 1)$		$\#(a = k_A, b = j)$		$\#(a = k_A, b = k_B)$

More in general, large values concentrated in few cells of the contingency matrix indicate high agreement, whereas a more uniform distribution of values across the matrix indicate poor agreement

Comparing clusterings

Consider two clusterings $\mathcal{C}_A$ and $\mathcal{C}_B$ with k_A and k_B clusters

		Clustering $\mathcal{C}_B$		
		$b = 1$ \ $\dots$	$b = j$ \ $\dots$	$b = k_B$
Clustering $\mathcal{C}_A$	$a = 1$	$\#(a = 1, b = 1)$	$\#(a = 1, b = j)$	$\#(a = 1, b = k_B)$
	$\vdots$			
	$a = i$	$\#(a = i, b = 1)$	$\#(a = i, b = j)$	$\#(a = i, b = k_B)$
	$\vdots$			
	$a = k_A$	$\#(a = k_A, b = 1)$	$\#(a = k_A, b = j)$	$\#(a = k_A, b = k_B)$

Hence, the agreement between the clusterings can be intuitively assessed by just looking at the distribution of values across the contingency matrix

Measures can be computed from this matrix in order to quantify the degree of agreement

Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings include *cluster purity*

$$\text{Purity}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}$$

Values close to 1 are desirable, indicating that the clusters of $\mathcal{C}_A$ are very pure with respect to those of $\mathcal{C}_B$, i.e. a given cluster of $\mathcal{C}_A$ mainly contains points from the same cluster of $\mathcal{C}_B$
Cluster purity only takes into account the majority assignment

Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings include *cluster purity*, the *Gini index*

$$\text{Purity}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}$$

$$\text{Gini}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \cdot (1 - \sum_{j \in [1..k_B]} (\frac{\#(a = i, b = j)}{\#(a = i)})^2)$$

Values close to 0 are desirable
Larger values indicate that points from the same cluster of $\mathcal{C}_A$ are scattered across several clusters of $\mathcal{C}_B$

Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings include *cluster purity*, the *Gini index* and the *entropy*

$$\text{Purity}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}$$

$$\text{Gini}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \cdot \left(1 - \sum_{j \in [1..k_B]} \left(\frac{\#(a = i, b = j)}{\#(a = i)}\right)^2\right)$$

$$\text{Entropy}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \sum_{j \in [1..k_B]} -\frac{\#(a = i, b = j)}{\#(a = i)} \log_2 \left(\frac{\#(a = i, b = j)}{\#(a = i)}\right)$$

Values close to 0 are desirable

The *entropy* captures similar properties as the *Gini index*

Comparing clusterings

$$\text{Purity}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \max_{j \in [1..k_B]} \frac{\#(a = i, b = j)}{n}$$

$$\text{Gini}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \cdot (1 - \sum_{j \in [1..k_B]} (\frac{\#(a = i, b = j)}{\#(a = i)})^2)$$

$$\text{Entropy}(\mathcal{C}_A, \mathcal{C}_B) = \sum_{i \in [1..k_A]} \frac{\#(a = i)}{n} \sum_{j \in [1..k_B]} -\frac{\#(a = i, b = j)}{\#(a = i)} \log_2 (\frac{\#(a = i, b = j)}{\#(a = i)})$$

! Note that these measures are not symmetric

Computing Purity($\mathcal{C}_A, \mathcal{C}_B$) corresponds to taking $\mathcal{C}_B$ as reference, evaluating the purity of the clusters of $\mathcal{C}_A$ with respect to $\mathcal{C}_B$
Computing Purity($\mathcal{C}_B, \mathcal{C}_A$), i.e. taking $\mathcal{C}_A$ as reference instead, will not yield the same value in general
One might compute the measure in both directions and take the average

Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings, to evaluate one clustering against another clustering, include *cluster purity*, the *Gini index* and the *entropy*

These measures constitute *external* validation criteria, since they rely on an external reference to evaluate a clustering

Comparing clusterings

Measures computed from the contingency matrix in order to quantify the degree of agreement between two clusterings, to evaluate one clustering against another clustering, include *cluster purity*, the *Gini index* and the *entropy*

These measures constitute *external* validation criteria, since they rely on an external reference to evaluate a clustering

If some ground truth is available, for instance in the case of synthetically generated data, it can be used as reference

In our example, for instance, we might consider that grouping beans by species provides a useful reference for evaluating the obtained clusters