Scientific Python

Multivariate Statistics

What will we cover?

- Distances
- Dimension reduction
- Clustering

Artwork by @allison_horst – X account – https://allisonhorst.com/

Equations for the braves

Mean:

$$m_x = \frac{1}{n}\sum_{i=1}^{n} x$$

Variance:

$$\text{var}(x) = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - m)^2$$

Standard-deviation:

$$s_x = \sqrt{\text{var}(x)}$$

Covariance:

$$\text{cov}(x, y) = \frac{1}{n-1}\sum_{i=1}^{n} (x_i - m_x)(y_i - m_y)$$

Correlation:

$$\text{cor}(x, y) = \frac{\text{cov}(x, y)}{s_x s_y}$$

Distances

What is a distance?

Give an adjective or a few words of what a « distance » should be.

Definitions

Distance

$d(x,y) \geq 0$

$d(x,x) = 0$

$d(x,y) = d(y,x)$

$d(x,z) \leq d(x,y) + d(y,z)$

$d(x,y) = 0 \Leftrightarrow x = y$

Similarity

$s(x,y) \geq 0$

$s(x,y) = s(y,x)$

$s(x,x) \geq s(x,y)$

Dissimilarity

Like a distance but without the triangular inequality

Distances

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

$$d(x, y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

$$d(x, y) = ((x_1 - y_1)^q + (x_2 - y_2)^q + \cdots + (x_n - y_n)^q)^{\frac{1}{q}}$$

scikit-learn

- Multivariate statistics and Machine Learning
- API: ‘fit’, ‘predict’, ‘transform’, ‘score’
- Pipelines!
- Programmed with efficiency in mind

- Distances :

from sklearn.metrics import pairwise_distances

Example

- Let’s practice computing distances on a (very fake) dataset
  - 100 rows that I will sometimes call « observations », split evenly into two groups A and B
  - 4 columns that I will very often call variables: group, Gene1, Gene2, and Gene3
- The file with the data is called **day5_example.csv**

- The training Jupyter Notebook is located in the folder « **Courses** » and is called

day5_01_distances.ipynb

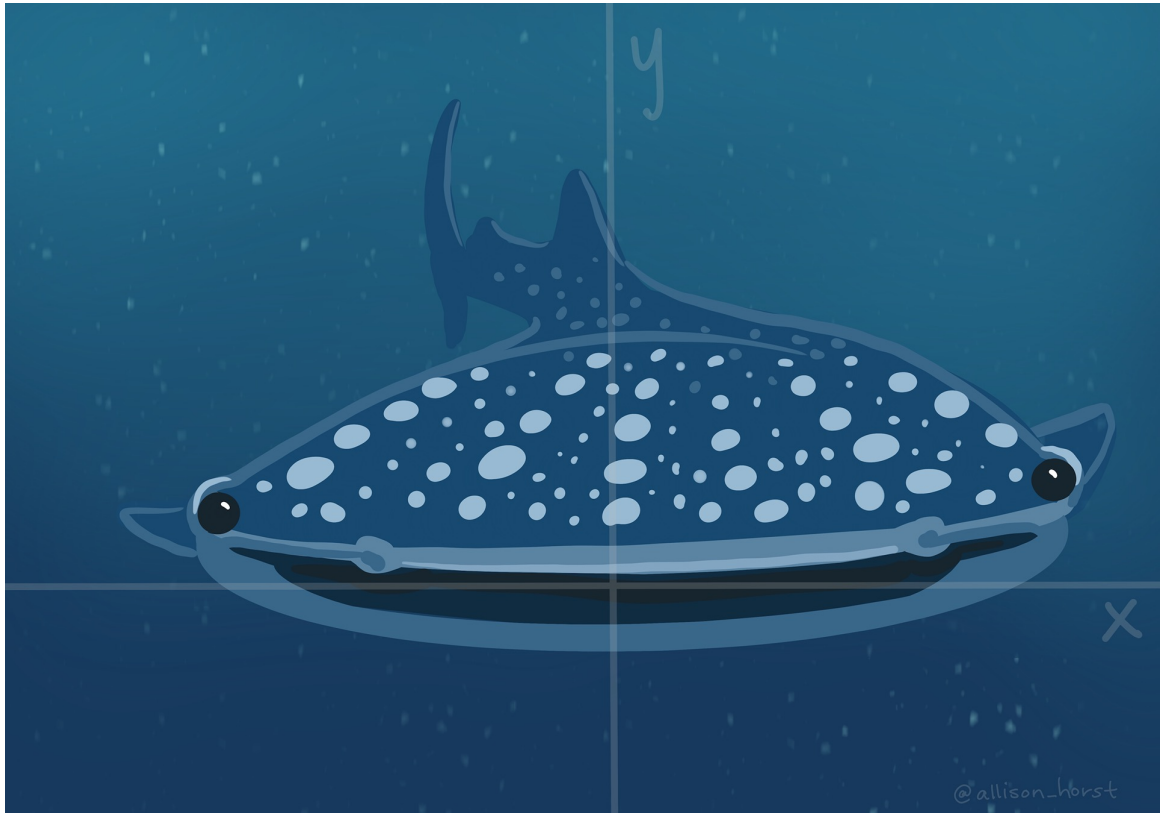
Exercice : day5_01_distance.ipynb

1. Load the dataset « example_day5.csv »
2. Compute the Euclidean distance between the observations
3. Make a heatmap
4. Compute the correlation distance between the variables
5. Make a heatmap

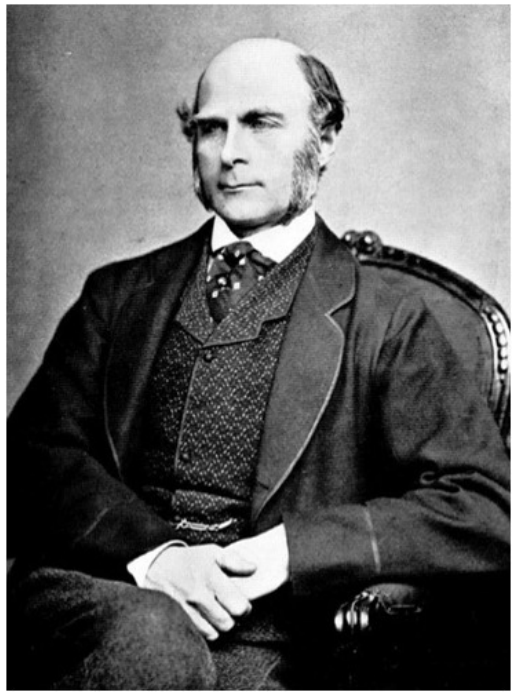
Principal Component Analysis

Whale shark metaphor: the data is krill

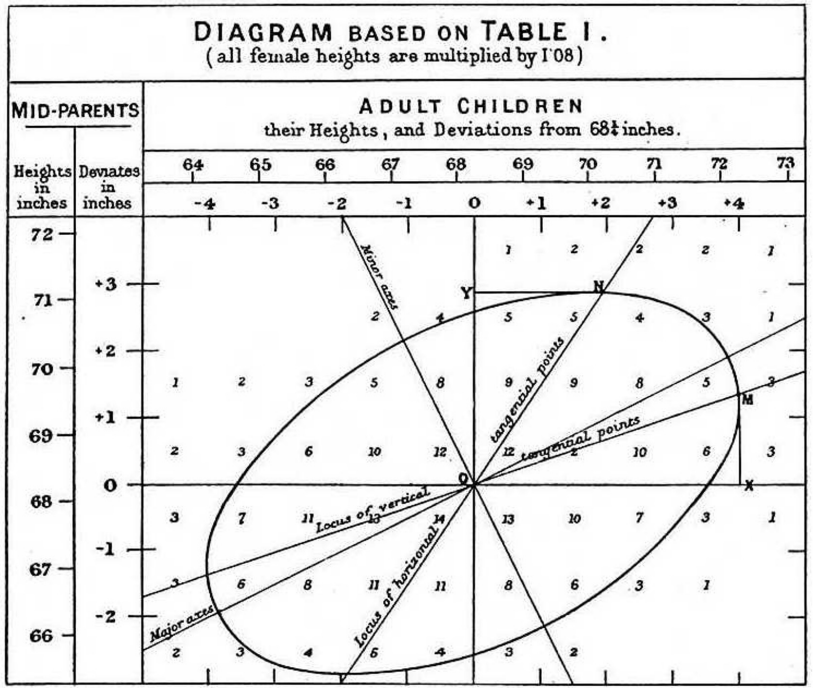
Whale shark metaphor: we are the whale



A little bit of history



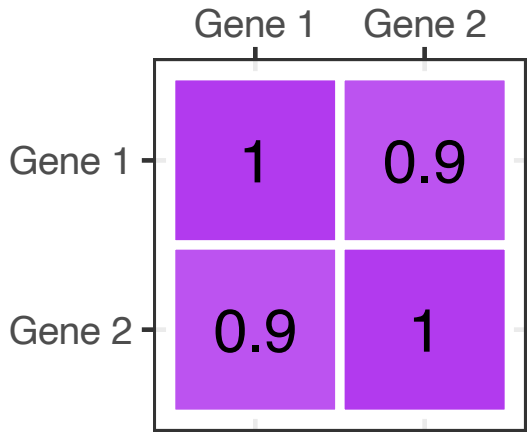
Sir Francis Galton (1822 – 1911)



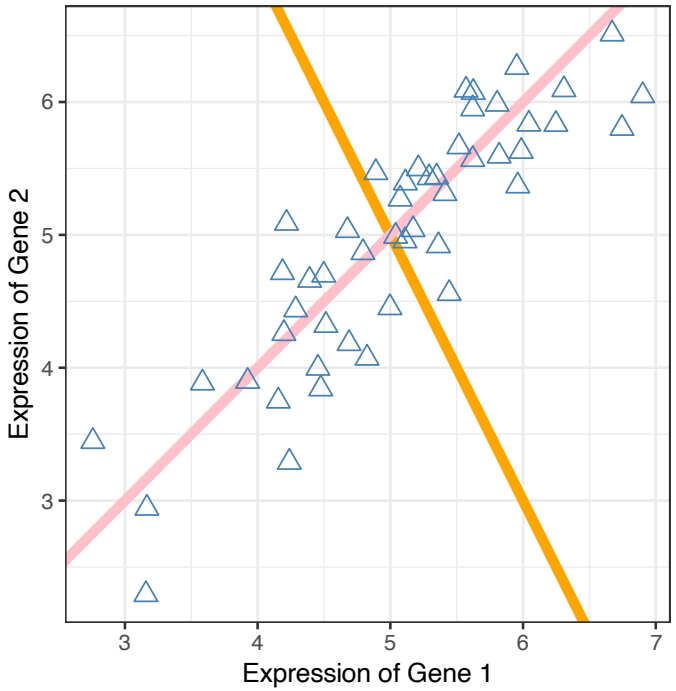
Galton, 1886
(https://doi.org/10.2307/2841583)

OK, but what does it mean (in 2D) ?

Correlation matrix:

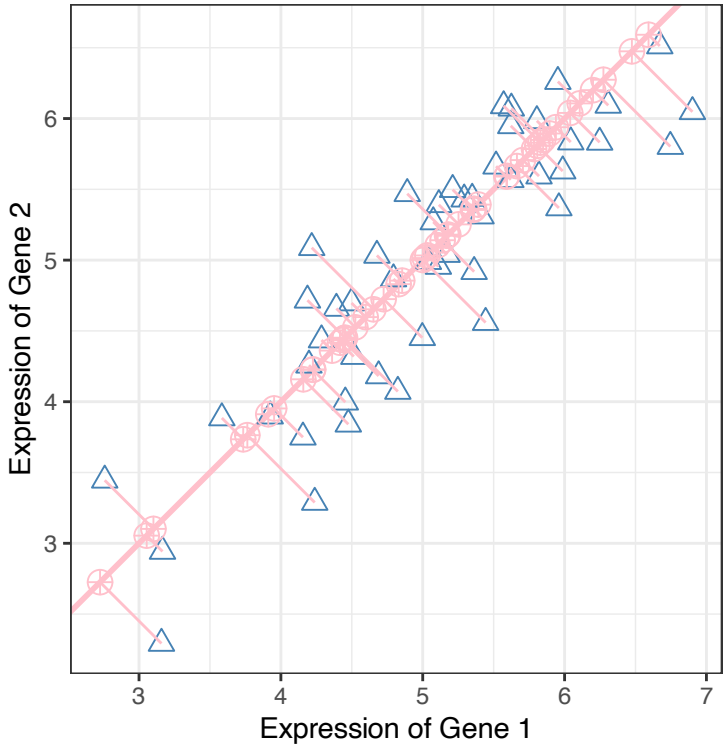


- Orange: not the first PC
- Pink: the first PC



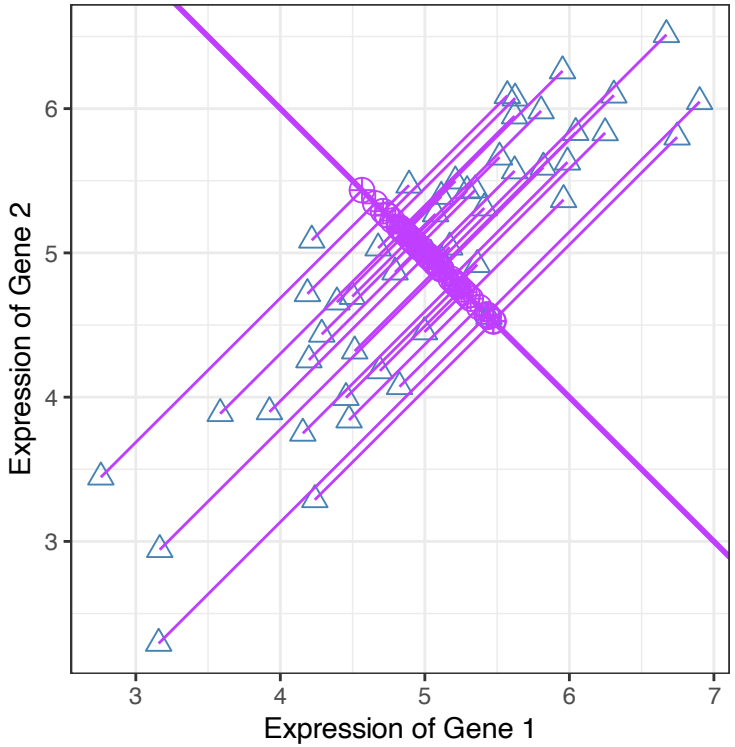
OK, but what does it mean (in 2D) ?

- The line is a mathematical object ($y = ax + b$)
- The observations have coordinates on this line



OK, but what does it mean (in 2D) ?

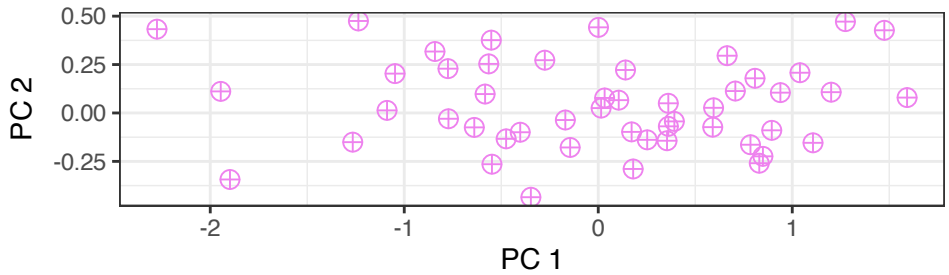
- To get the 2nd principal component, go in the orthogonal direction
- The observations also have coordinates on this line

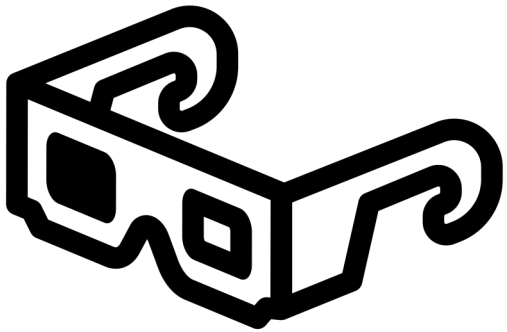


OK, but what does it mean (in 2D) ?

What you want are

- the new coordinates !
- how much each variable (gene) is used to create these coordinates
- the new variance





Can we see it in 3D?

What is the goal of principal component analysis

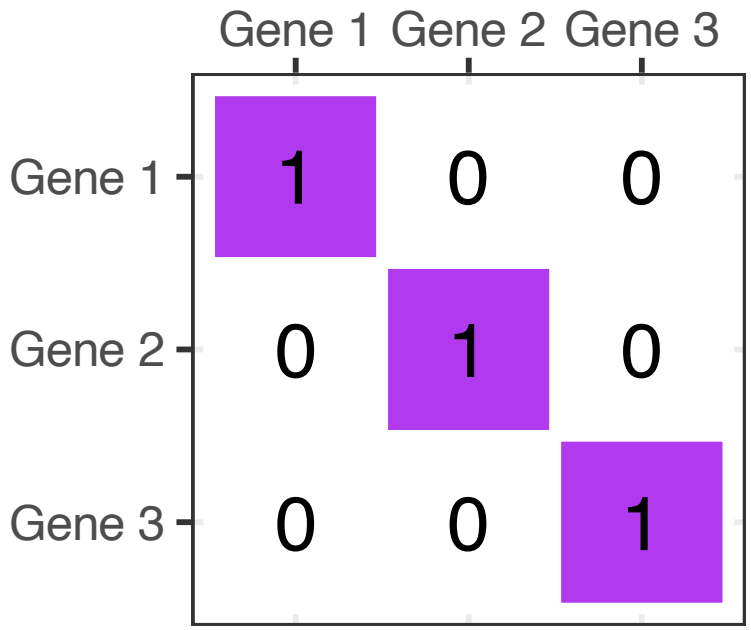
The aim of PCA is to create principal components (PCs):

- a PC is a **linear combination** of the variables (X_i)
- the first PC has the **highest variance**, the second PC has the second highest variance etc.
- the **weights** (w_i) of the linear combination are *constrained*

$$\text{PC} = w_1 X_1 + w_2 X_2 + w_3 X_3 + \cdots$$

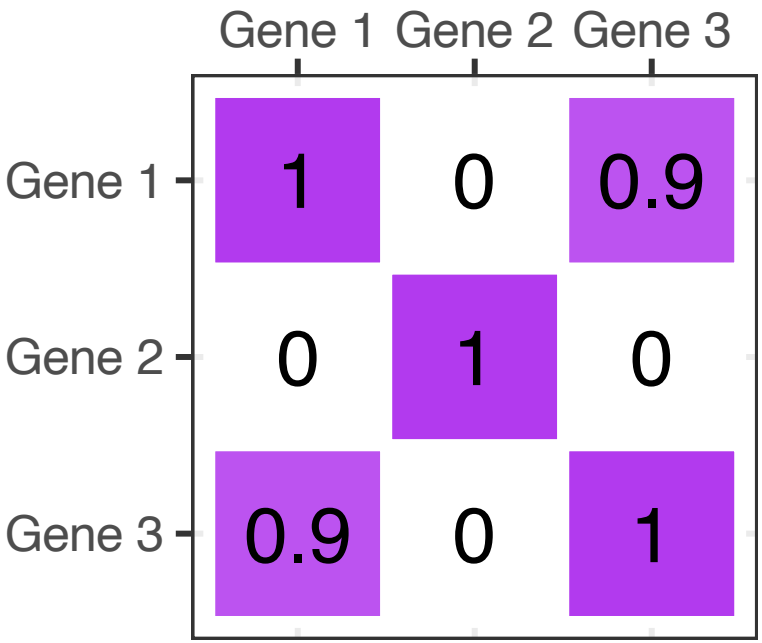
Let's examine three (imaginary) cases.

The variance of PC1 - example 1



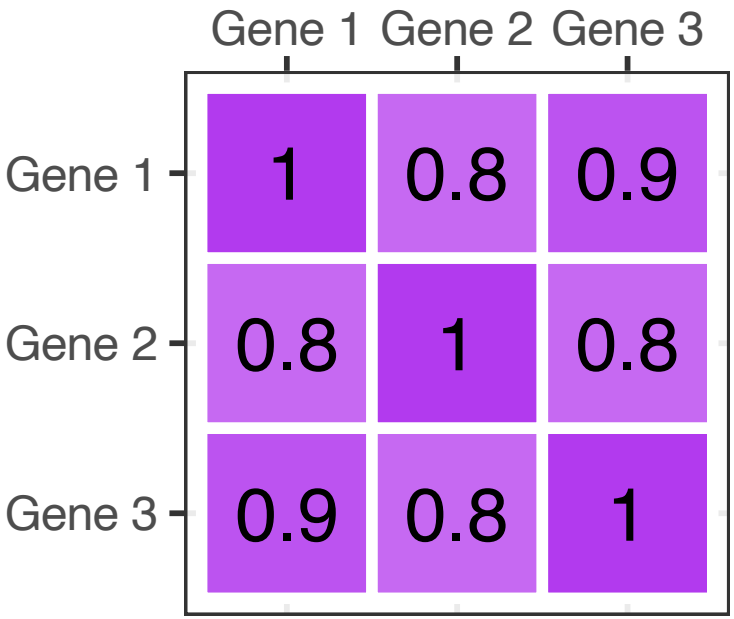
- What is the variance of the first principal component?
- What are the weights?

The variance of PC1 - example 2



- What is the variance of the first principal component?
- What are the weights?

The variance of PC1 - example 3



- What is the variance of the first principal component?
- What are the weights?

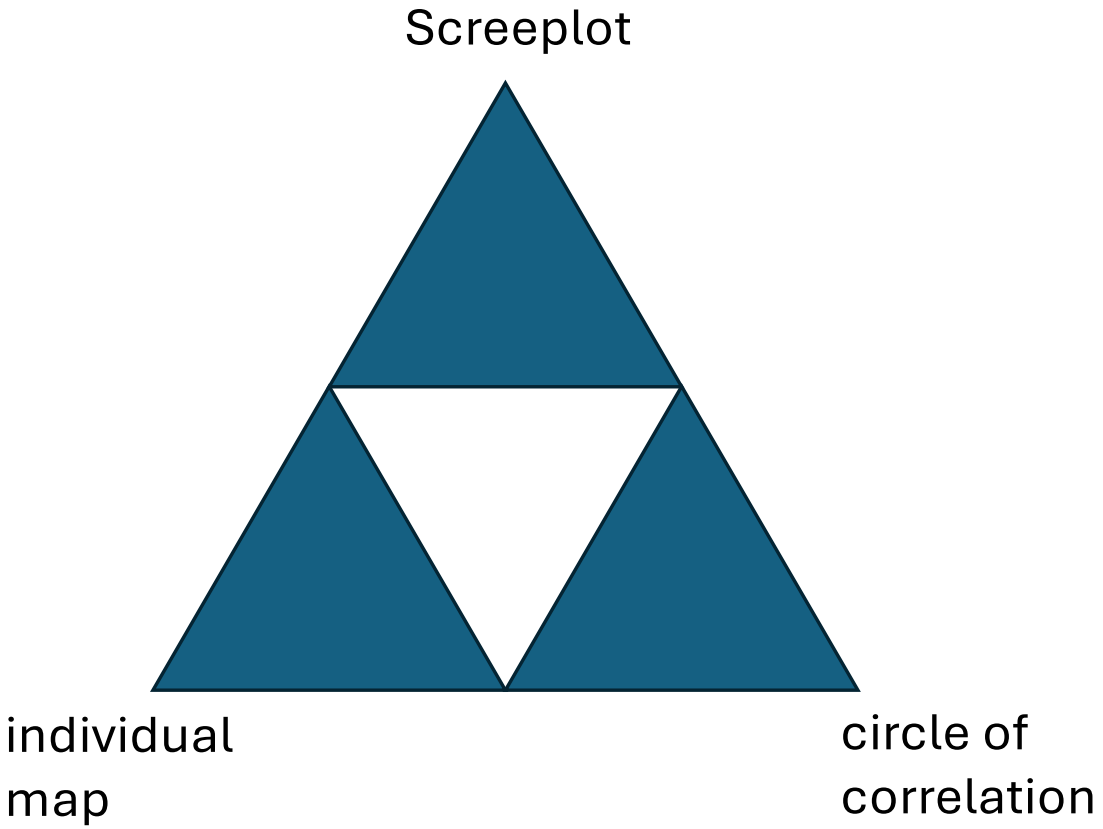
Two (or more) flavors of PCA

- Standardized PCA (in 49.5% of cases): variables are centered and scaled
- (Regular, or centered, or plain) PCA (in 50.4% of cases): variables are only centered.
- Non-centered PCA (in 0.1%): variables remain unchanged.

Remember that correlation is the same as covariance of the scaled data?

- Regular PCA: based on **covariance**
- Standardized PCA: based on **correlation**

The tri-force of PCA



Simulated data

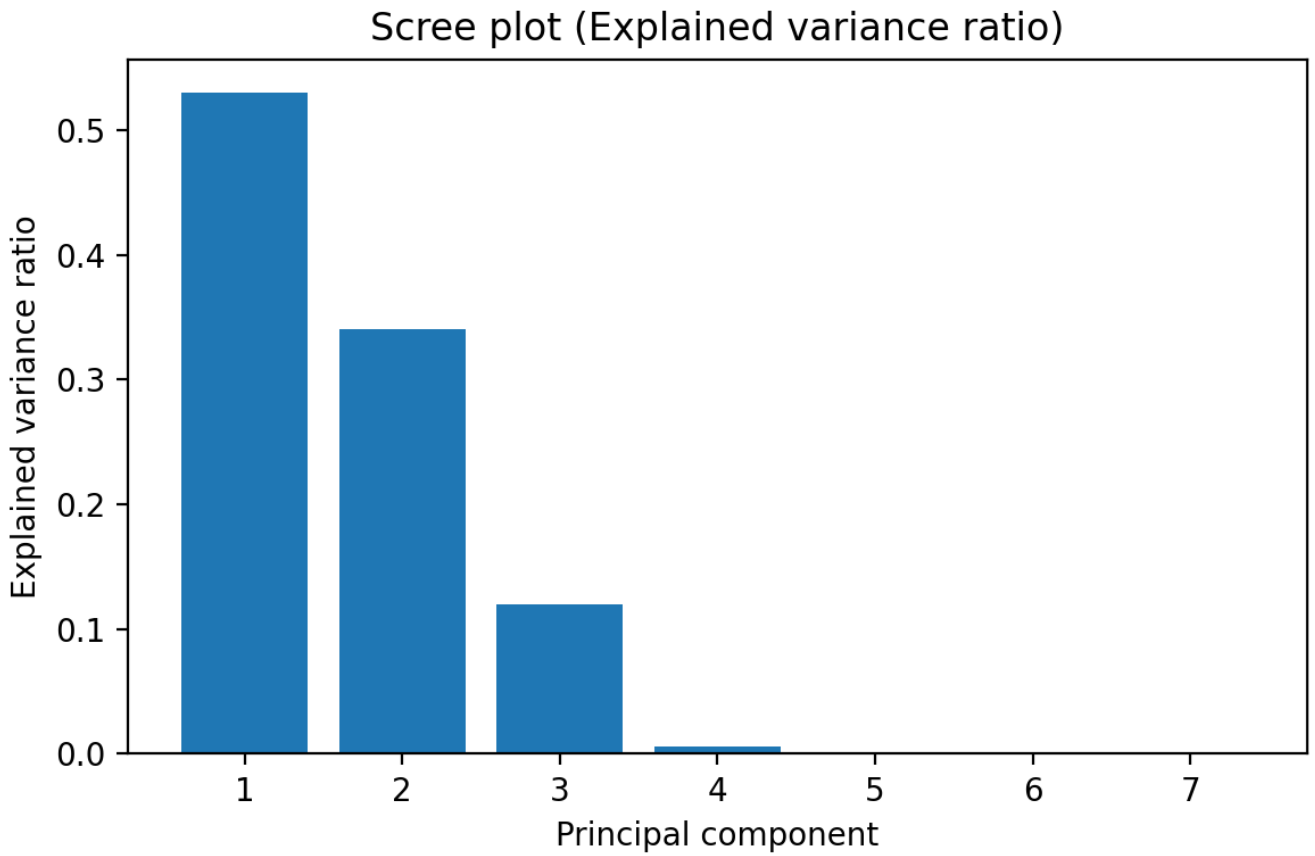
```
import numpy as np
import pandas as pd

n = 15
rng = np.random.default_rng()

X1 = rng.standard_normal(n)
X2 = rng.standard_normal(n)
X7 = 0.1 * rng.standard_normal(n)
```

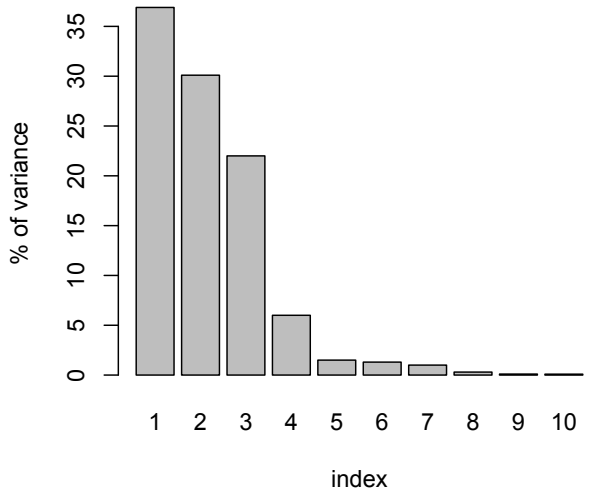
```
dat_ex = pd.DataFrame({
    "X1": X1,
    "X2": X2,
    "X3": -X1,
    "X4": 2 * X2 + 0.25 * rng.standard_normal(n),
    "X5": X1 + X2 + 0.25 * rng.standard_normal(n),
    "X6": X1 - X2 + 0.25 * rng.standard_normal(n),
    "X7": X7
})
```

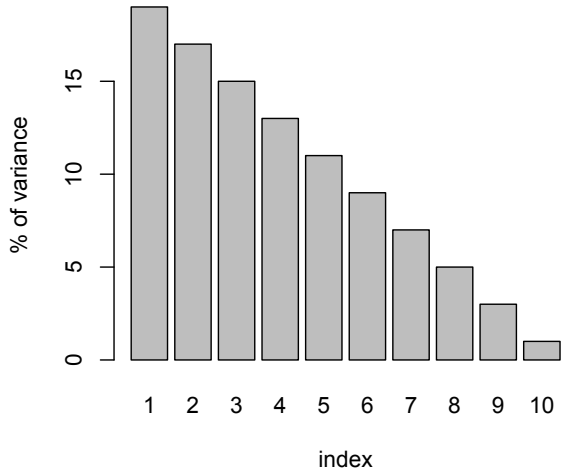
Screeplot


Scree plot (Explained variance ratio)
Explained variance ratio
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
Principal component

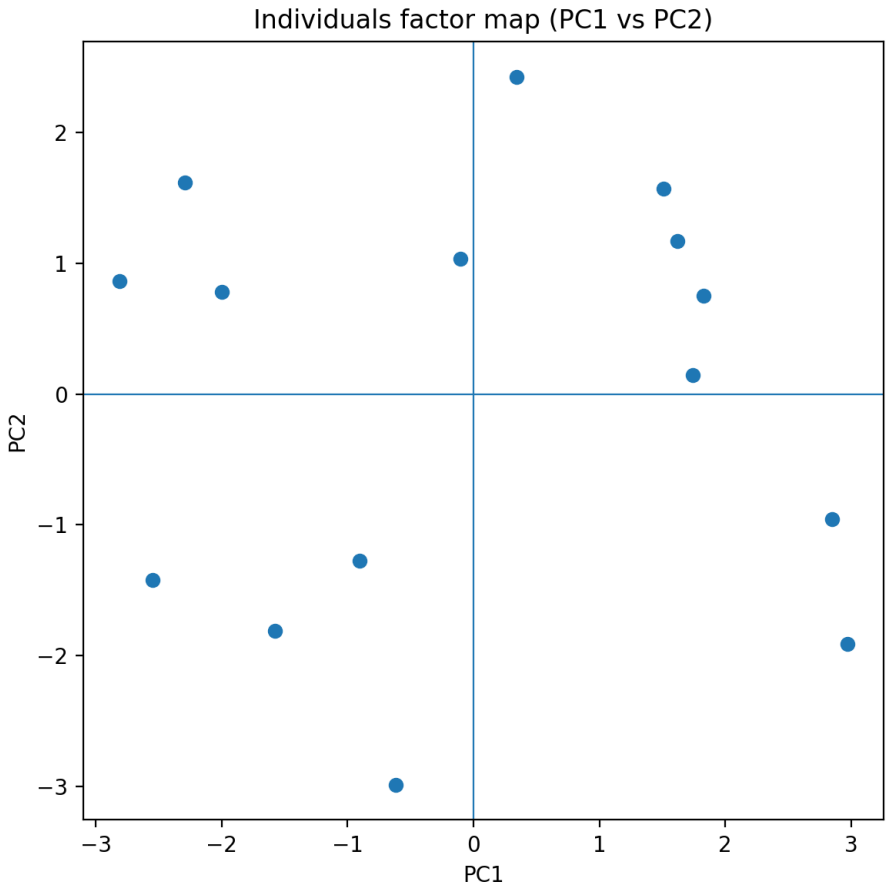
How many PCs?
Examples of methods

- Always useful to have a look at the plot of the fraction of total variance explained by each component.
- *Scree* graph (Cattell, 1966) : Look at the plot of the percentage of variance l_k against k and decide which value of k the defines an *'elbow'* in the graph (subjective). In practice, rarely easy to choose.





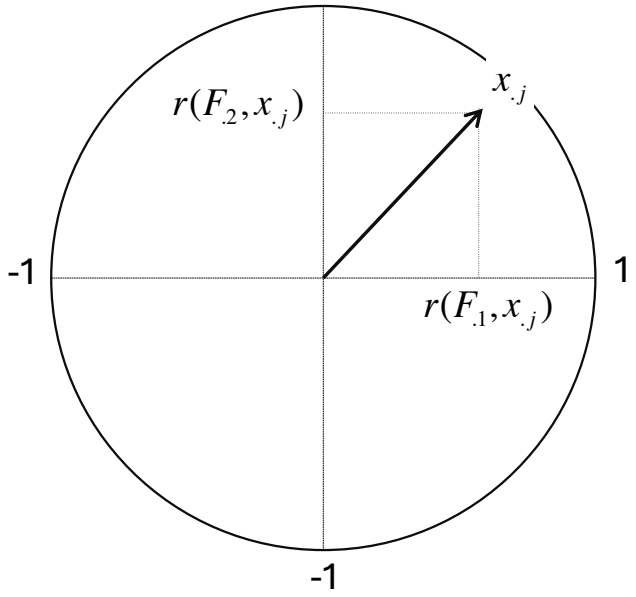
Example individual map



Interpretation of the PCA plot of the individuals using the variables

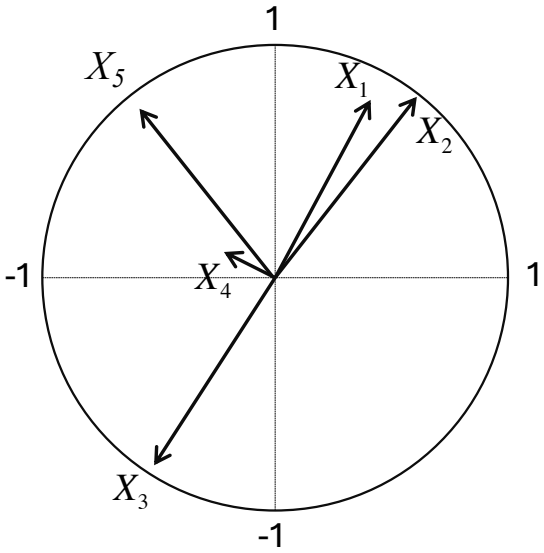
- We can study the correlation between the initial variables and the PCs.
- The correlation between a PC and a variable estimates the information they share.

- **Correlation circle:** The variables can be plotted as points in the component space using their correlations with the coordinates of the individuals on the PCs.

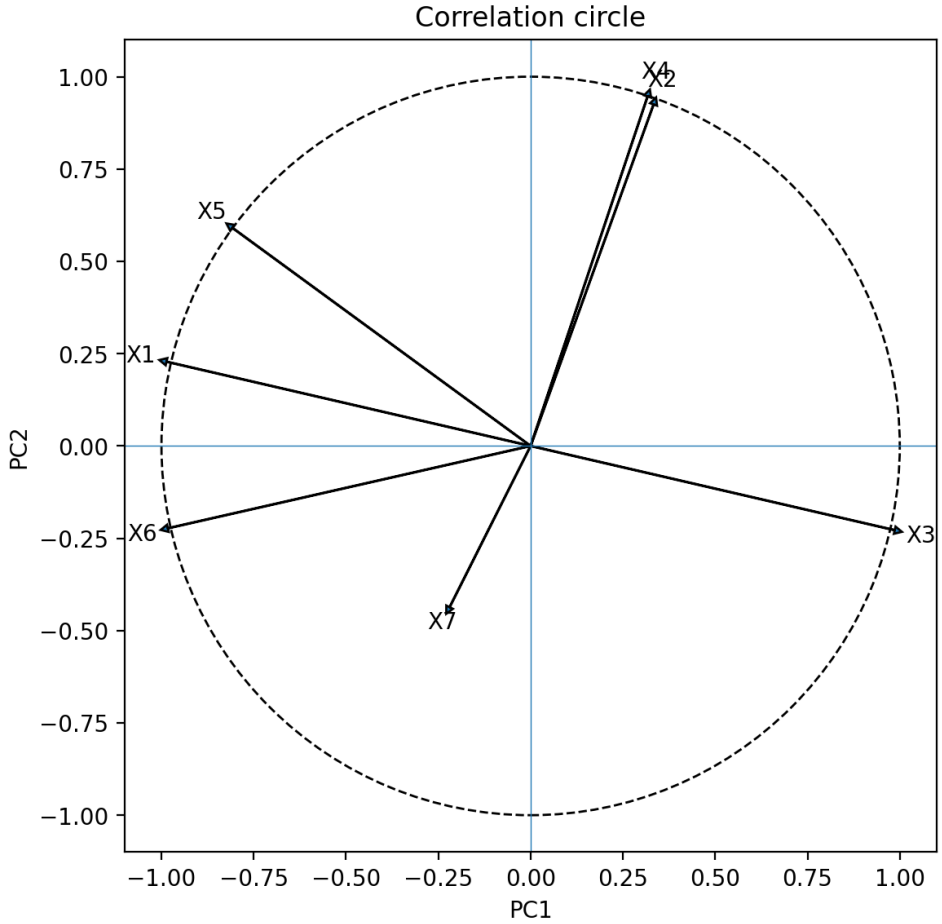


Interpretation of the PCA plot of the individuals using the variables

- An arrow close to the circle indicates a variable that is well characterized by the represented PCs (X_1, X_2, X_3, X_5).
- An arrow close to the center indicates a variable whose properties are not captured/described by the represented PCs (X_4).
- Two arrows close to the circle and to each other indicate a positive correlation between the two variables (X_1, X_2).
- Two arrows close to the circle and in opposite direction indicates a negative correlation between the two variables (X_1, X_3 or X_2, X_3).
- Two arrows close to the circle and at a right angle to one another are not correlated (X_1 with X_5 or X_2 with X_5).



Example circle of correlation



To go further

Explain the results of PCA with

- **Contribution**: how important a variable or an individual is to the construction of a principal component.
- **“Cos2” (squared cosine)**: quality of the representation of a variable or individual on a principal component.

PCA in Python

```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

scaler = StandardScaler()
Xz = scaler.fit_transform(X)

pca = PCA(n_components=3, random_state=0)

X_pca = pca.fit_transform(Xz)
expl_var = pca.explained_variance_ratio_
```

Exercice : day5_02_pca.ipynb

1. Load the dataset « example_day5.csv »
2. Run a standardized PCA on the data
3. Plot the scree: how many dimensions will you keep?
4. Plot the individual map, color the groups
5. Plot the correlation circle
6. Conclude

UMAP

Statistics > Machine Learning

[Submitted on 9 Feb 2018 (v1), last revised 18 Sep 2020 (this version, v3)]

UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction

Leland McInnes, John Healy, James Melville

UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The result is a practical scalable algorithm that applies to real world data. The UMAP algorithm is competitive with t-SNE for visualization quality, and arguably preserves more of the global structure with superior run time performance. Furthermore, UMAP has no computational restrictions on embedding dimension, making it viable as a general purpose dimension reduction technique for machine learning.

Comments: Reference implementation available at this http URL

Subjects: **Machine Learning (stat.ML)**; Computational Geometry (cs.CG); Machine Learning (cs.LG)

Cite as: arXiv:1802.03426 **[stat.ML]**

(or arXiv:1802.03426v3 **[stat.ML]** for this version)

https://doi.org/10.48550/arXiv.1802.03426

The main results

Theorem 1. *The functors* FinReal : **Fin-sFuzz** $\to$ **FinEPMet** *and* FinSing : **FinEPMet** $\to$ **Fin-sFuzz** *form an adjunction with* FinReal *the left adjoint and* FinSing *the right adjoint.*

Algorithm 1 UMAP algorithm

function UMAP(X, n, d, min-dist, n-epochs)

 # Construct the relevant weighted graph
for all $x \in X$ **do**
fs-set$[x] \leftarrow$ LocalFuzzySimplicialSet(X, x, n)
top-rep $\leftarrow \bigcup_{x \in X}$ fs-set$[x]$ *# We recommend the probabilistic t-conorm*

 # Perform optimization of the graph layout
$Y \leftarrow$ SpectralEmbedding(top-rep, d)
$Y \leftarrow$ OptimizeEmbedding(top-rep, Y, min-dist, n-epochs)
return Y

Main parameters

- n = “neighborhood size”
- min-dist = “algorithmic parameter controlling the layout”
- d = “dimension of the target reduced space”
- n-epochs = “amount of optimization work to perform”

Understanding UMAP

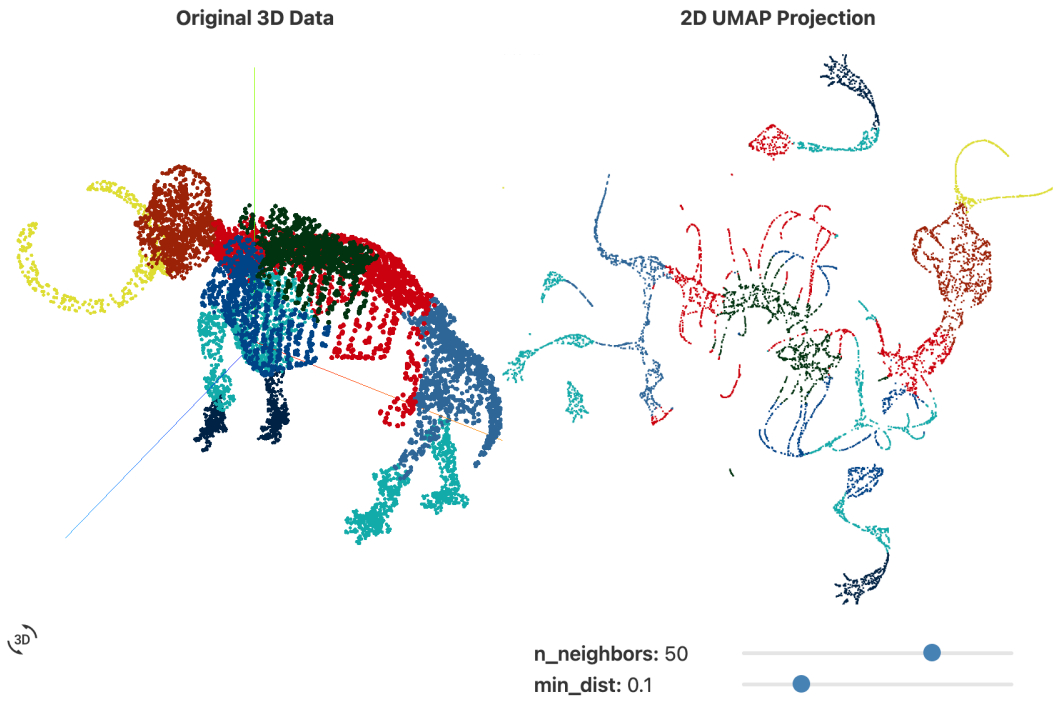


Figure 5: *UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the* `n_neighbors` *and* `min_dist` *parameters.*

https://pair-code.github.io/understanding-umap/

UMAP in Python

```
## Load package
import umap

## Create a Model
umap_model = umap.UMAP(n_neighbors=100, min_dist=10)

## Run the model on X (pandas DataFrame)
embedding = umap_model.fit_transform(X.values)

## Build a DataFrame with the result
dat_umap = pd.DataFrame({
    "UMAP1": embedding[:, 0],
     "UMAP2": embedding[:, 1],
     "group": X.loc[:,'group']
})
```

Exercice : day5_03_umap.ipynb

1. Load the dataset « example_day5.csv »
2. Run UMAP with
    1. min_dist = 1 and n_neighbors = 1
    2. min_dist = 0.1 and n_neighbors = 10
3. Plot the result with seaborn, and color the result (by group)

Hierarchical clustering

hierarchical clustering: single linkage

(Step-by-step: combine clusters with the smallest distance between elements)

elements

DISTANCE MATRIX

	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

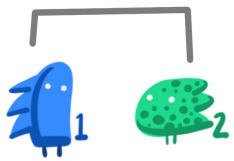
Treat each element as a cluster
- Find smallest distance between elements in 2 clusters
- Those clusters get merged.

elements

DISTANCE MATRIX

	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

build the DENDROGRAM



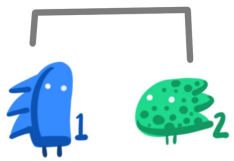
Treat each element as a cluster
- Find smallest distance between elements in 2 clusters
- Those clusters get merged.

elements

DISTANCE MATRIX

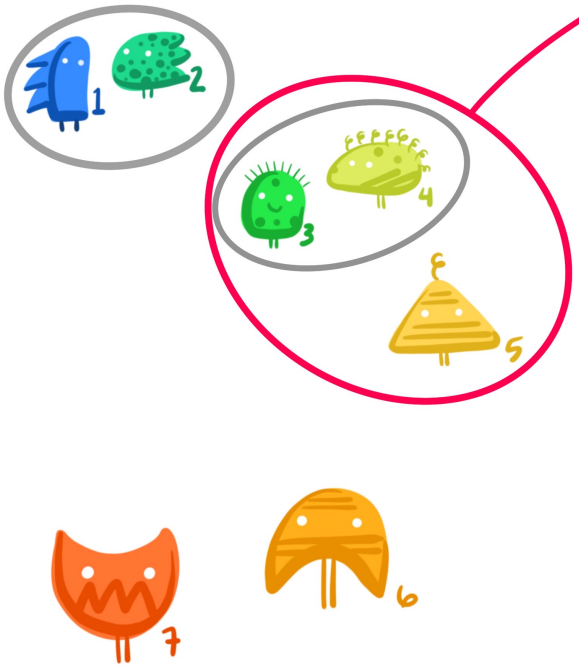
	1	2	3	4	5	6	7
1	0	10	30	40	60	85	82
2	10	0	24	38	55	87	90
3	30	24	0	16	26	50	63
4	40	38	16	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

build the DENDROGRAM



Repeat! Now the 2 clusters with the smallest distance between elements are the (3, 4) and 5 clusters, so we merge them!

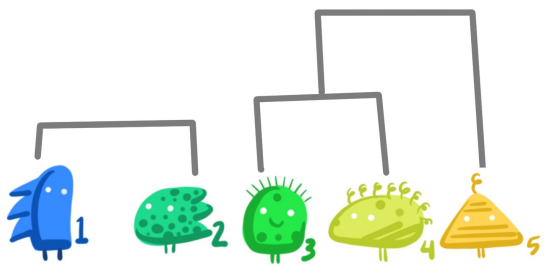
elements



DISTANCE MATRIX

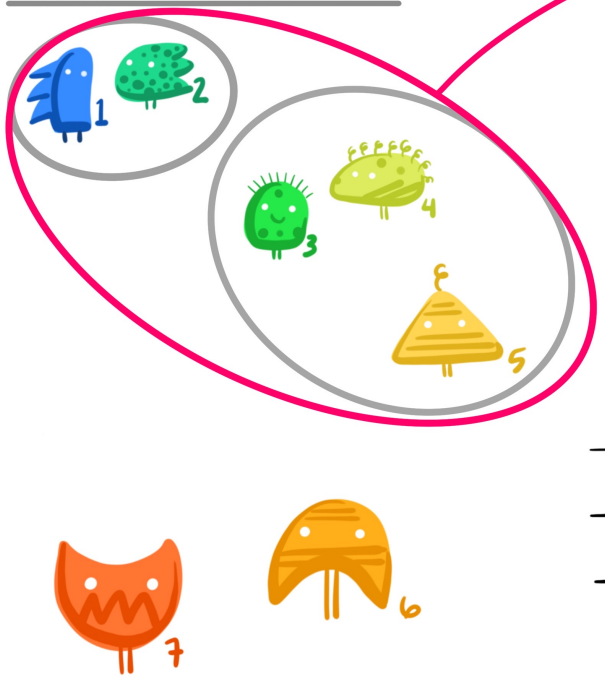
	1	2	3	4	5	6	7
1	0	~~10~~	30	40	60	85	82
2	~~10~~	0	24	38	55	87	90
3	30	24	0	~~16~~	26	50	63
4	40	38	~~16~~	0	21	52	67
5	60	55	26	21	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

build the DENDROGRAM



Yep, do it again! Now, the smallest distance between elements in two clusters is between 2 & 3, so we merge the clusters they're in!

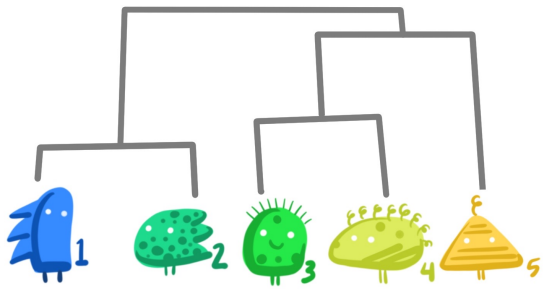
elements



DISTANCE MATRIX

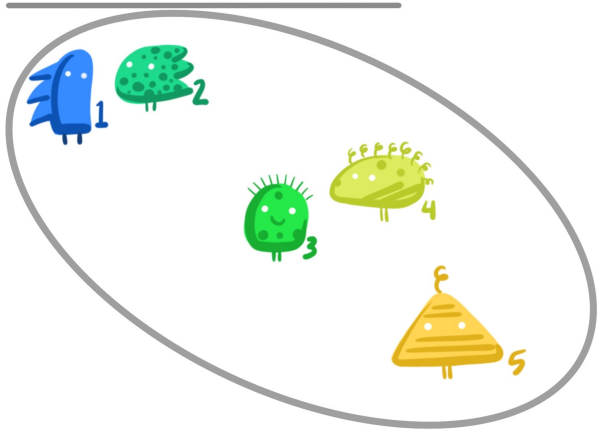
	1	2	3	4	5	6	7
1	0	~~10~~	30	40	60	85	82
2	~~10~~	0	24	38	55	87	90
3	30	24	0	~~16~~	26	50	63
4	40	38	~~16~~	0	~~21~~	52	67
5	60	55	26	~~21~~	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

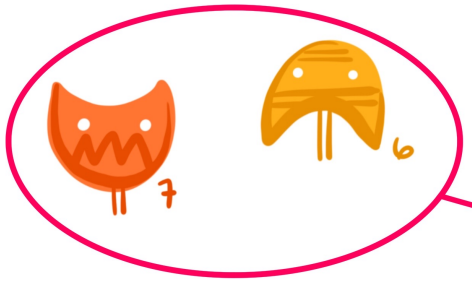
build the DENDROGRAM



The next smallest distance between elements in separate clusters is between 6 & 7, so we merge them...

elements

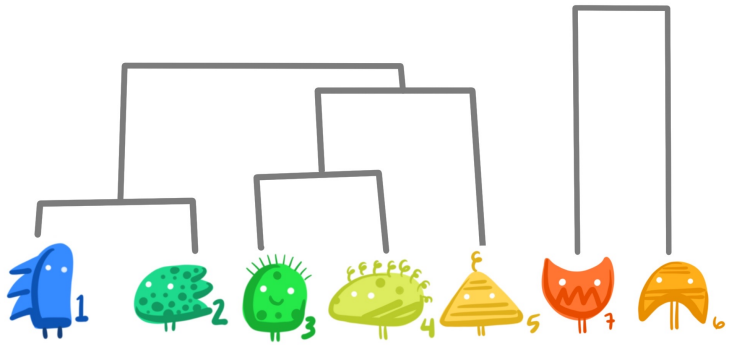




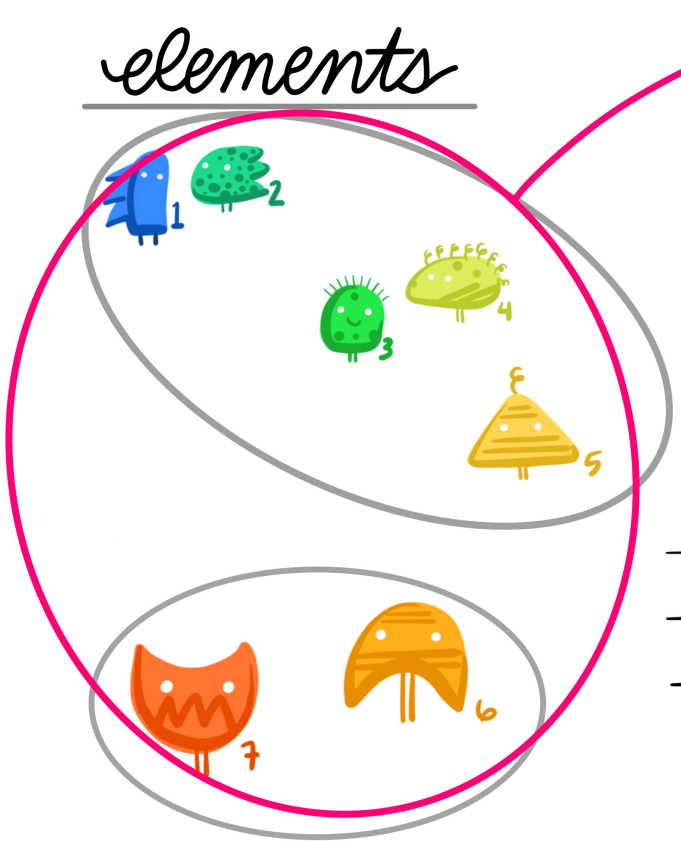
DISTANCE MATRIX

	1	2	3	4	5	6	7
1	0	~~10~~	~~30~~	~~40~~	~~60~~	85	82
2	~~10~~	0	~~24~~	~~38~~	~~55~~	87	90
3	~~30~~	~~24~~	0	~~16~~	~~26~~	50	63
4	~~40~~	~~38~~	~~16~~	0	~~21~~	52	67
5	~~60~~	~~55~~	~~26~~	~~21~~	0	41	58
6	85	87	50	52	41	0	32
7	82	90	63	67	58	32	0

build the DENDROGRAM

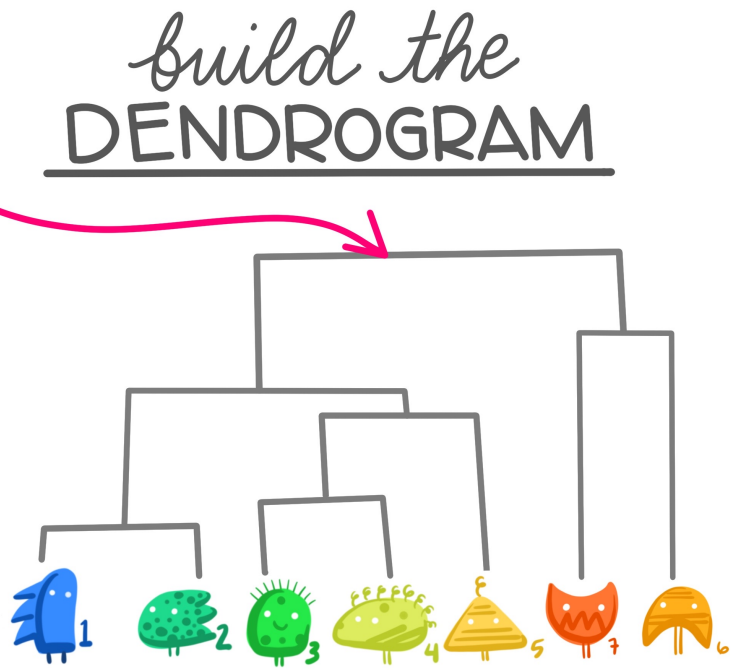


Now we only have two clusters, so they get merged!



DISTANCE MATRIX

	1	2	3	4	5	6	7
1	0	~~10~~	~~30~~	~~40~~	~~60~~	85	82
2	~~10~~	0	~~24~~	~~38~~	~~55~~	87	90
3	~~30~~	~~24~~	0	~~16~~	~~26~~	50	63
4	~~40~~	~~38~~	~~16~~	0	~~21~~	52	67
5	~~60~~	~~55~~	~~26~~	~~21~~	0	~~41~~	58
6	85	87	50	52	41	0	~~32~~
7	82	90	63	67	58	~~32~~	0



tada.

7/7

Hierarchical clustering in Python

```
from sklearn.cluster import AgglomerativeClustering

model = AgglomerativeClustering(n_clusters=k, linkage='ward')
labels = model.fit_predict(X)
```

Maxi Bonus 1: Silhouette for the quality of clustering

Given a partition P and an observation i:

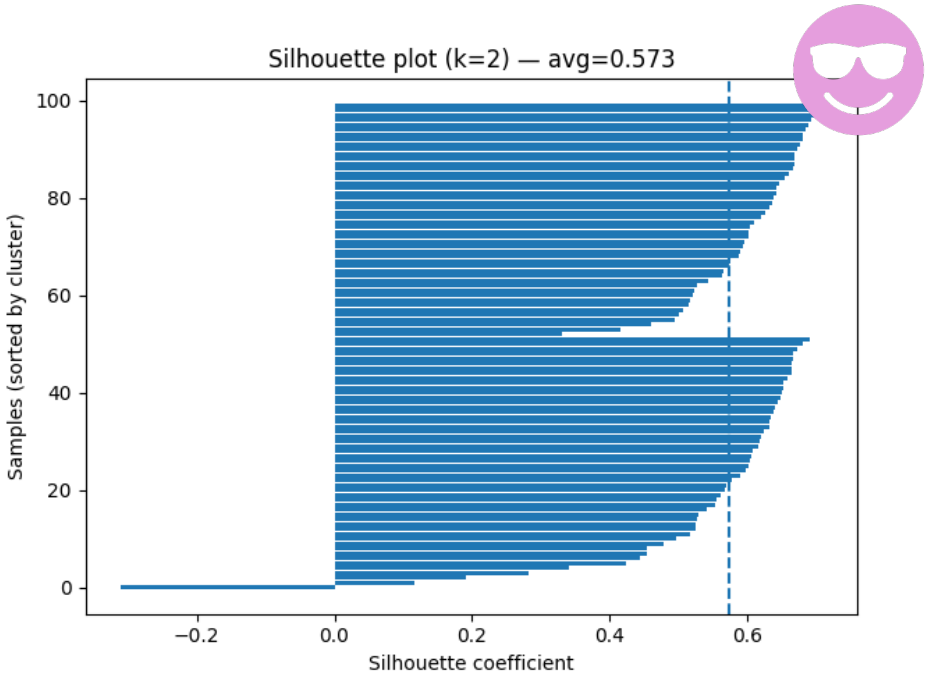
- $a(i) =$ mean distance of an object i to its own cluster
- $b(i) =$ mean of the distances to the other clusters,

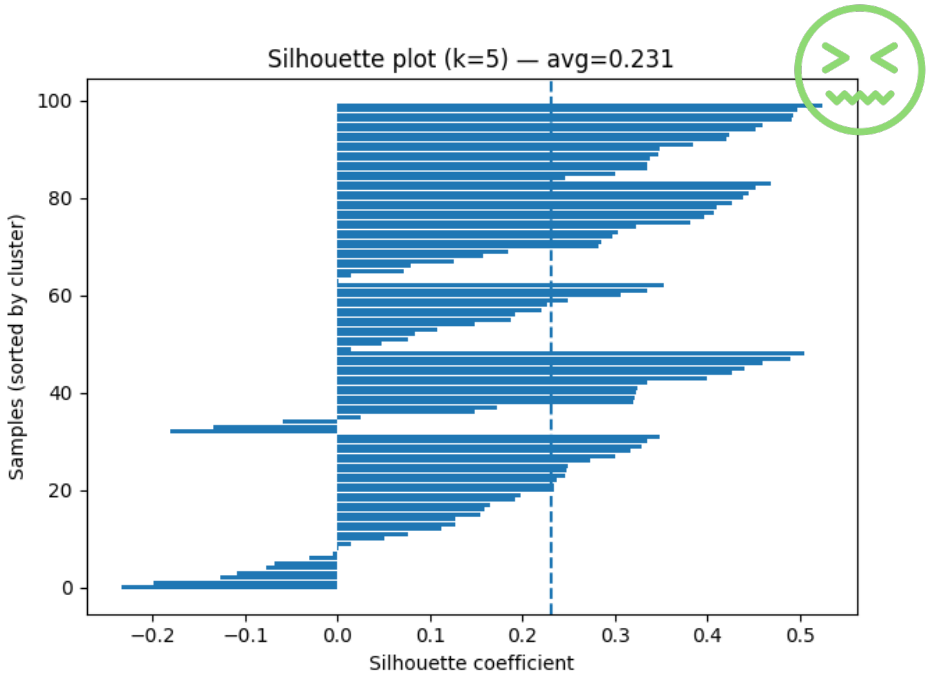
$$\text{sil}(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}.$$

Take the average across all i to get a global index.... or plot all the coefficients!

```
from sklearn.metrics import silhouette_samples, silhouette_score
sil = silhouette_samples(X, labels, metric='euclidean')
sil_avgs[k] = silhouette_score(X, labels, metric='euclidean')
```

Silhouette plot





Exercice : day5_04_hclust.ipynb

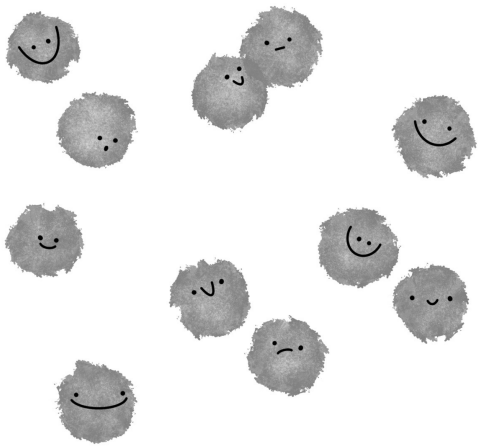
1. Load the dataset « example_day5.csv »
2. Plot the data (heatmap + dendrograms) with seaborn
3. Compute the silhouette plot for k = 2, 3, 4 and 5, and select the best value
4. Compute Rand index between a partition with the optimal number of clusters and the column « group »

k-means

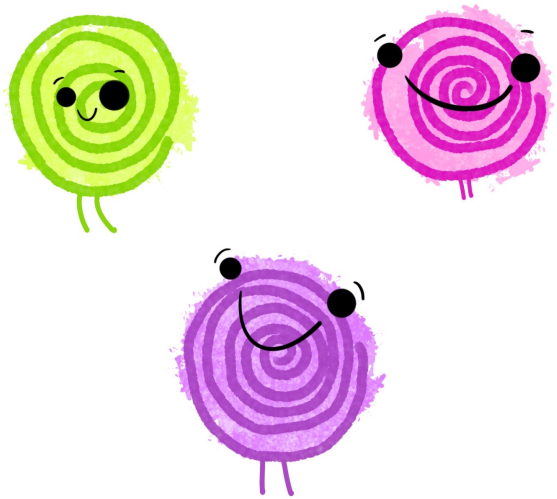
k-means clustering

: assign each observation to one of k clusters based on the nearest cluster centroid.

OBSERVATIONS

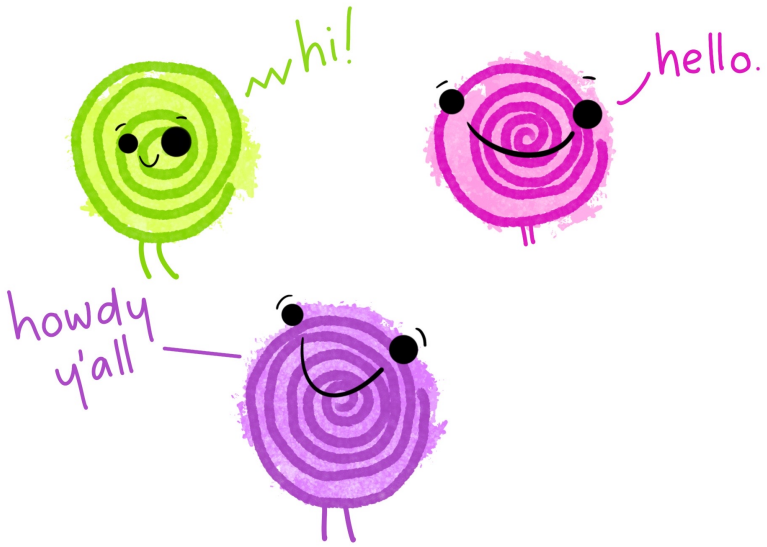


cluster CENTROIDS

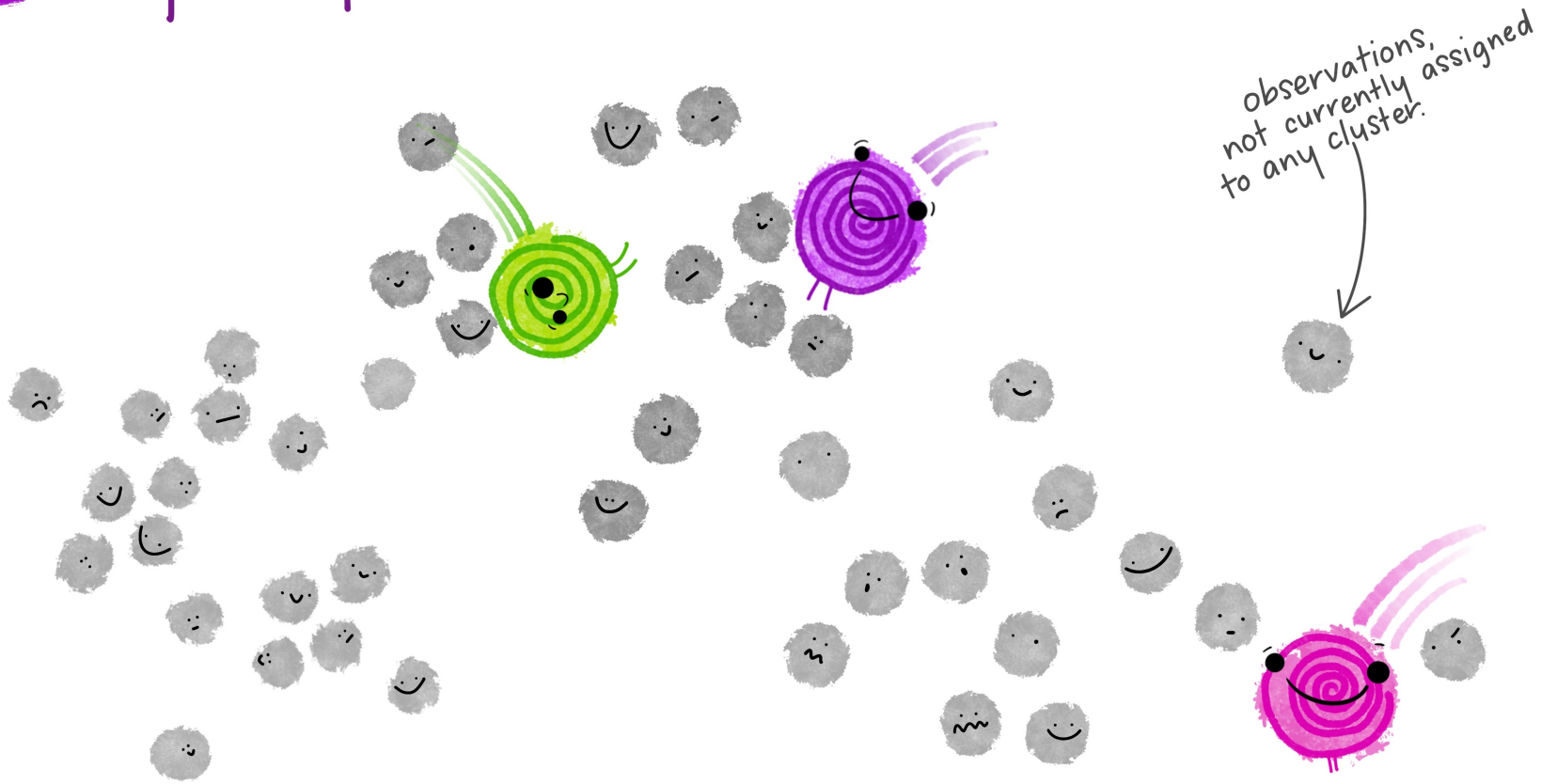


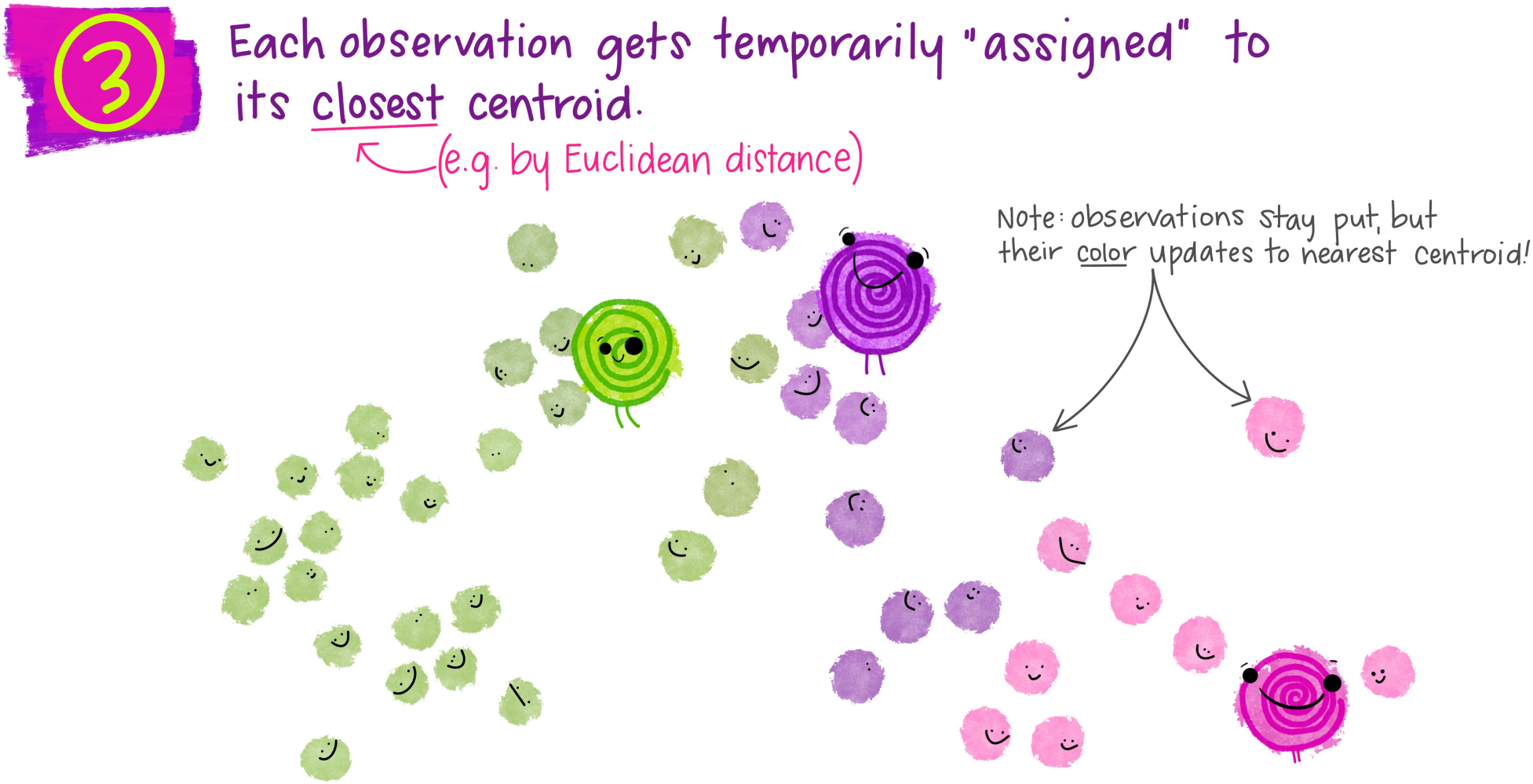
1 Specify the number of clusters (in this example, k = 3).

Then imagine k cluster centroids are created.



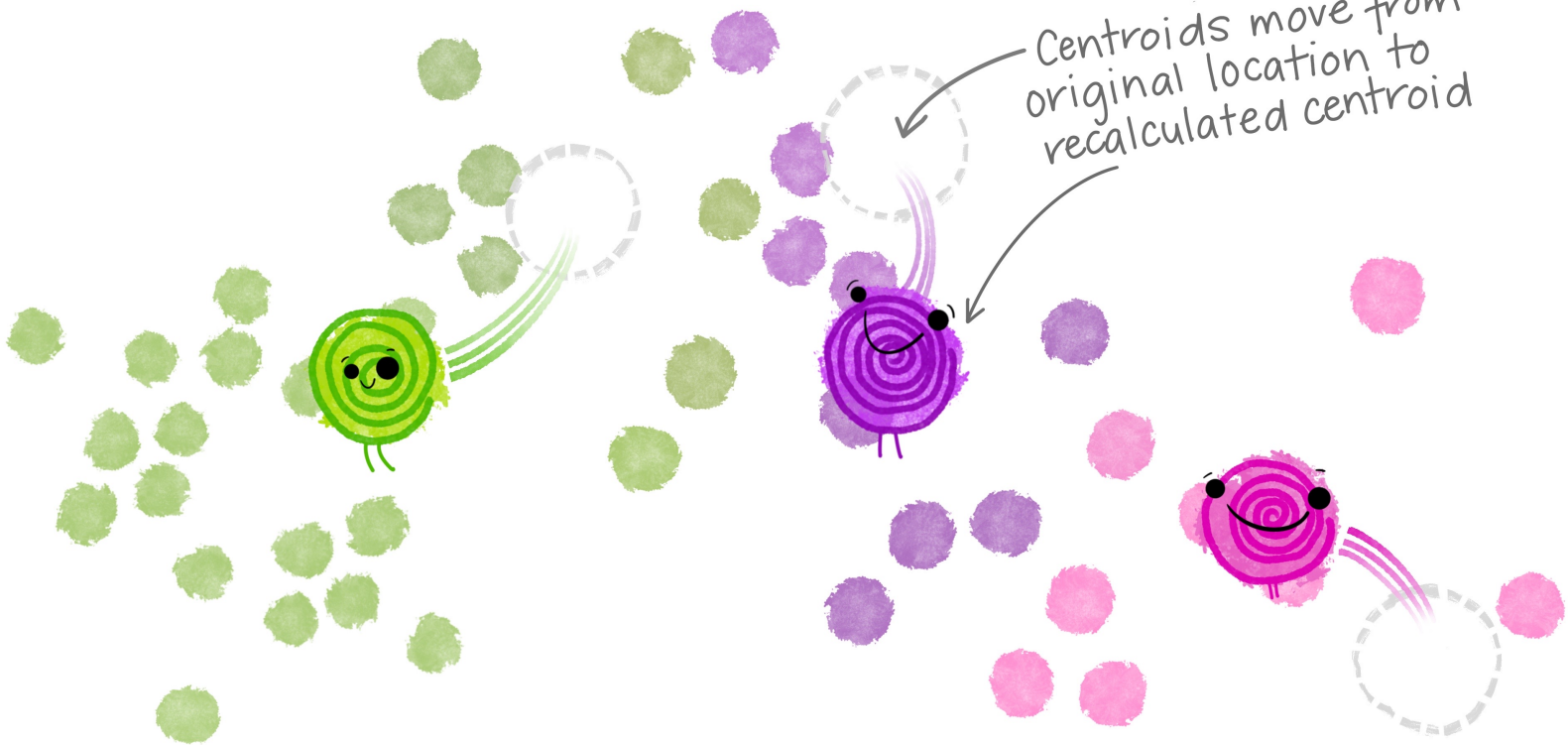
2 Those k centroids get randomly placed in your space.

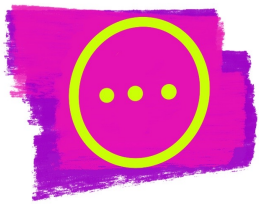



3
Each observation gets temporarily "assigned" to its closest centroid.
(e.g. by Euclidean distance)
Note: observations stay put, but their color updates to nearest centroid!
@allison_horst

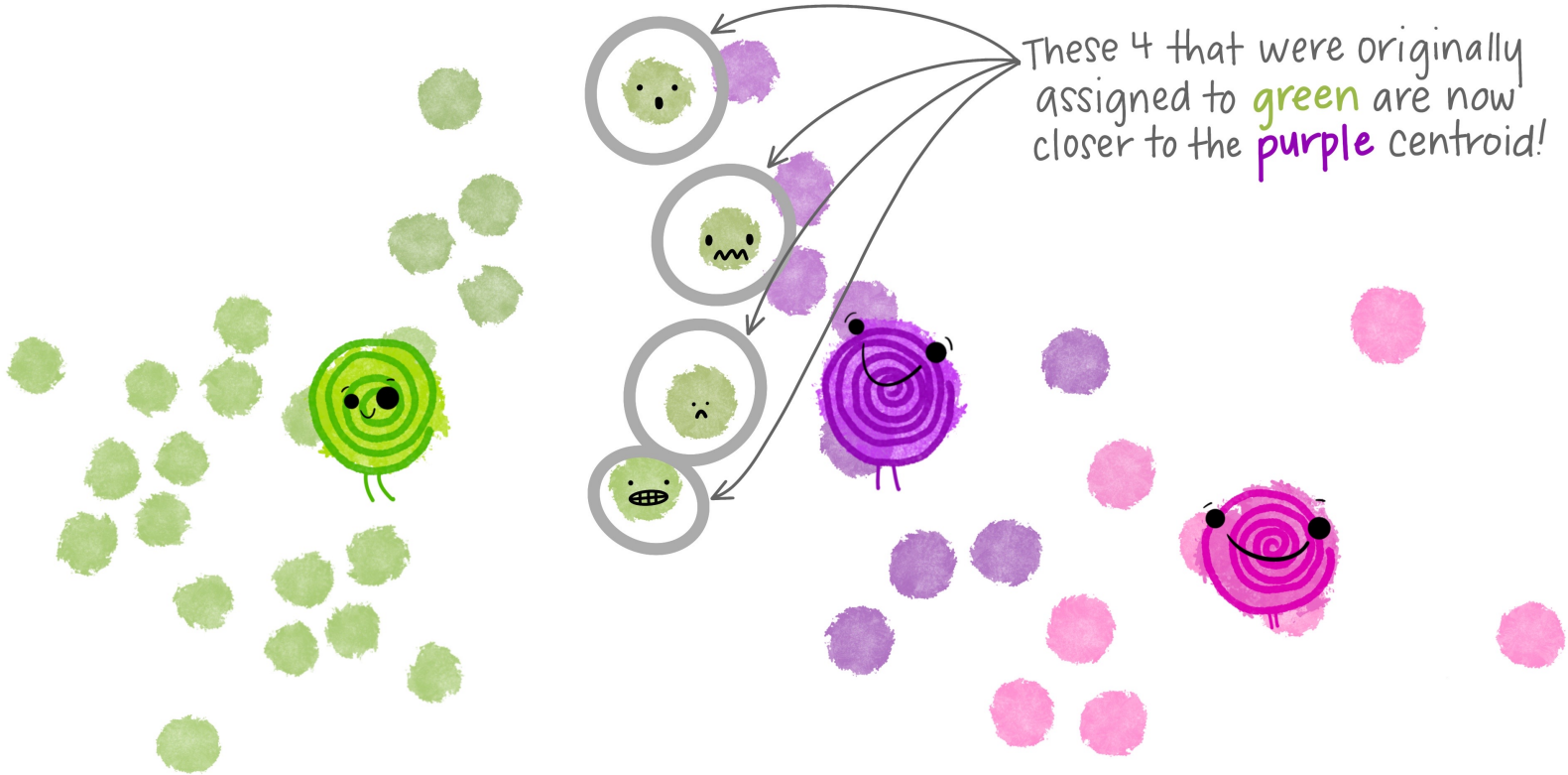
4

Then the centroid of each cluster is calculated based on all observations assigned to that cluster...





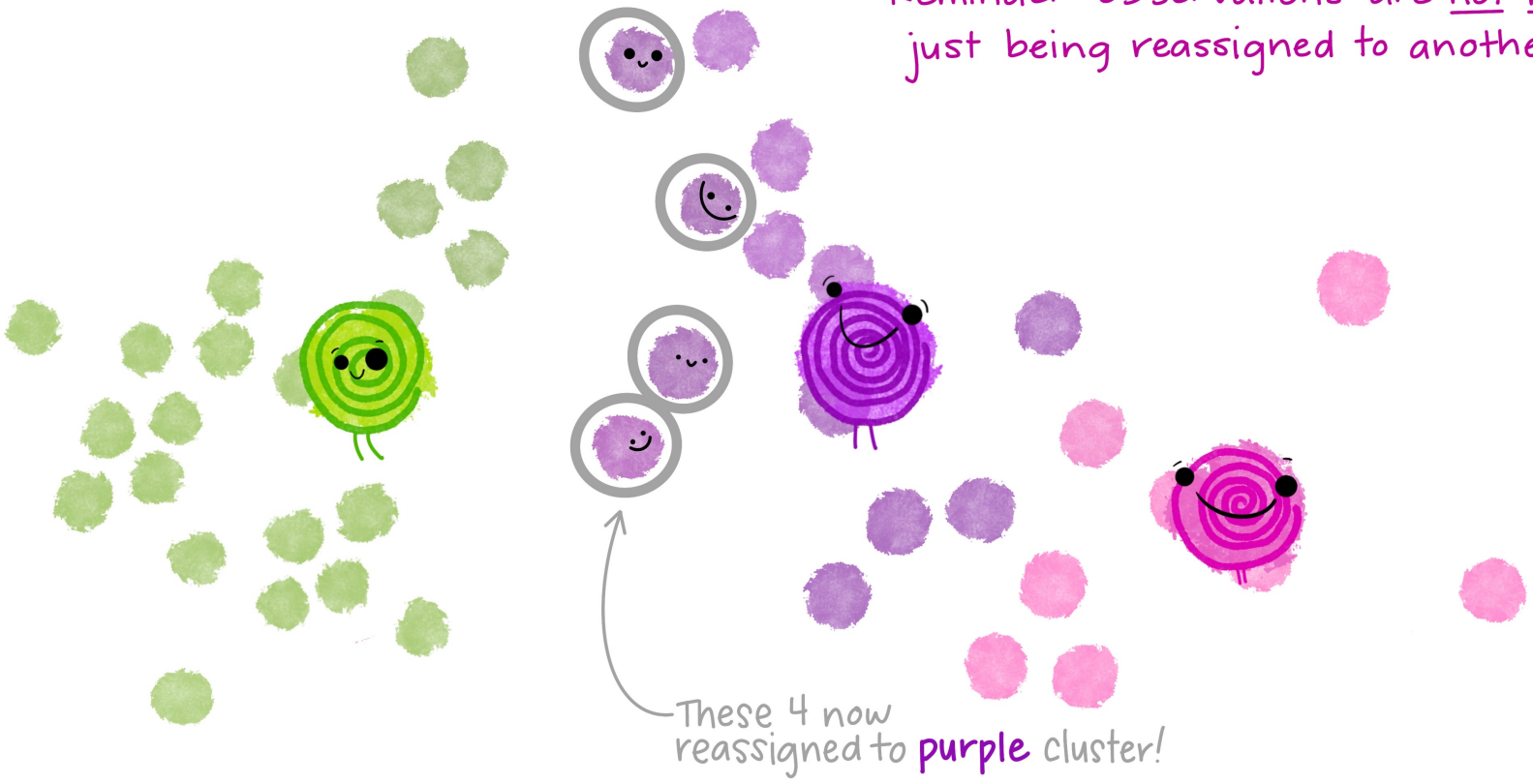
UH OH. Now that the cluster centroids have moved, some of the observations are now closer to a different centroid!

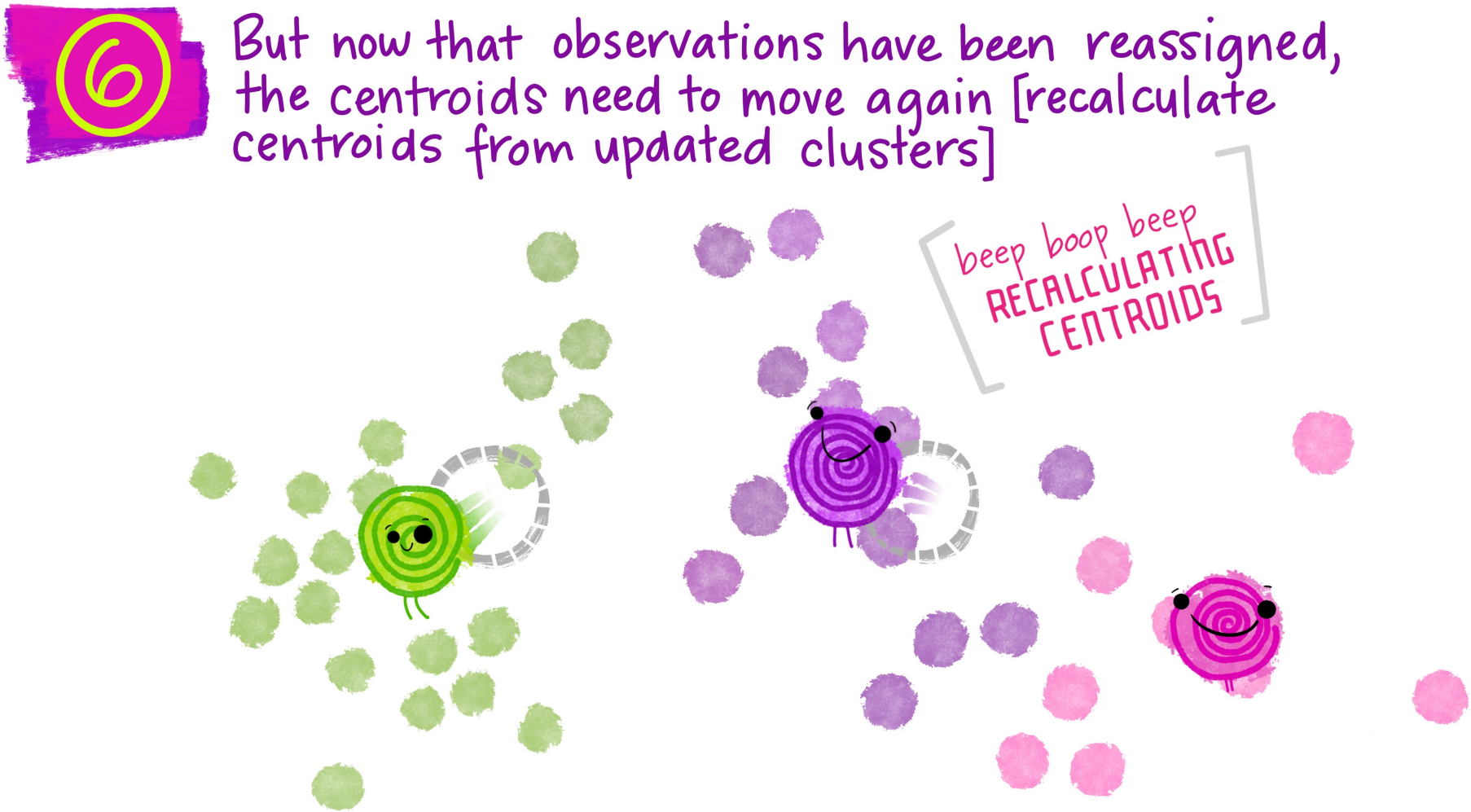


5

NO PROBLEM!
Observations get reassigned* to a different cluster based on the recalculated centroid.

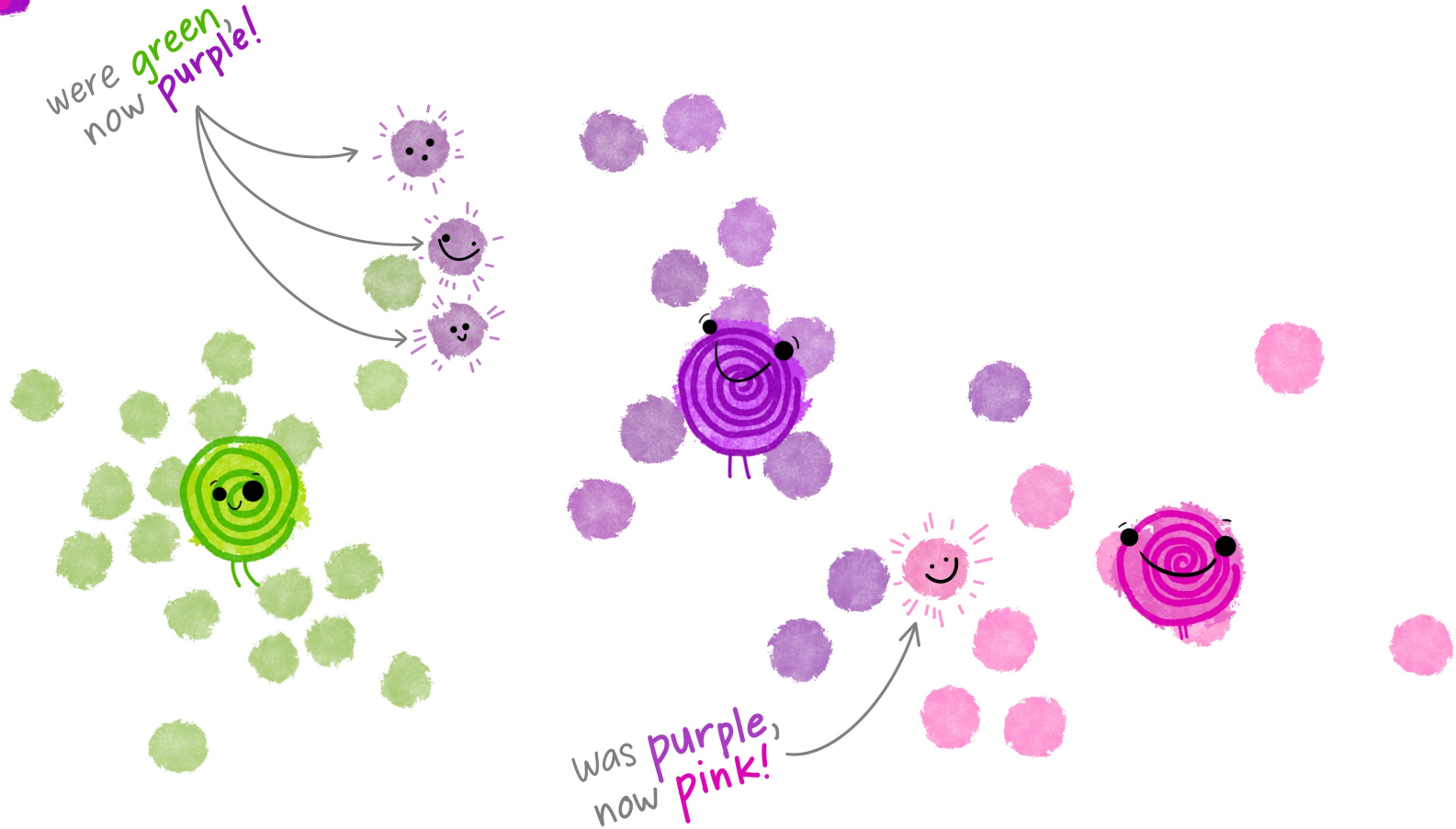
*Reminder: observations are not moving, just being reassigned to another cluster.

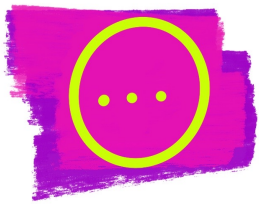



6
But now that observations have been reassigned, the centroids need to move again [recalculate centroids from updated clusters]
beep boop beep
RECALCULATING CENTROIDS
@allison_horst

7

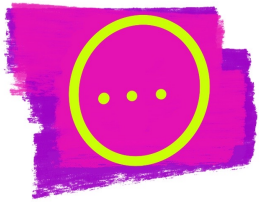
Again, now observations are reassigned as needed to the closest centroid.





Then the centroid for each cluster is recalculated...

...which means observations will be reassigned...



That iterative process of

Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↳ Recalculate cluster centroids

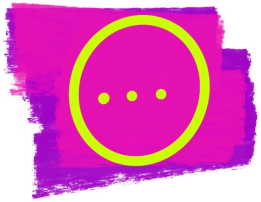
↳ Reassign observations to nearest centroid

↳ Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↓

Continues until nothing is moving or being reassigned anymore!



That iterative process of

Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↳ Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↳ Recalculate cluster centroids

↳ Reassign observations to nearest centroid

↓

Continues until nothing is moving or being reassigned anymore!

fin

Which means the iteration is done and each observation is assigned to its final cluster.

K-means in Python

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0, n_init=10)
labels = kmeans.fit_predict(X)
labels
```

Maxi Bonus 2: Rand index to compare partitions

Given two partitions P_1 and P_2:

- $a =$ **pairs** of objects *clustered together* in both
- $b =$ **pairs** of objects *clustered together* in P_1, but not in P_2
- $c =$ **pairs** of objects *separated* in P_1, but *together* in P_2
- $d =$ **pairs** of objects *separated* in both

Rand Index is computed as $\text{RI} = \frac{a+d}{a+b+c+d}$.

```
from sklearn.metrics import rand_score
ri = rand_score(y, labels)
```

Exercice : day5_05_kmeans.ipynb

1. Load the dataset « example_day5.csv »
2. Run k-means on the data with k=2
3. Compute the silhouette coefficient for k=2, 3, 4, and 5
4. What’s the optimal value?
5. Compute the Rand Index between the result of the clustering and the group column in the data

Le Grand Bain