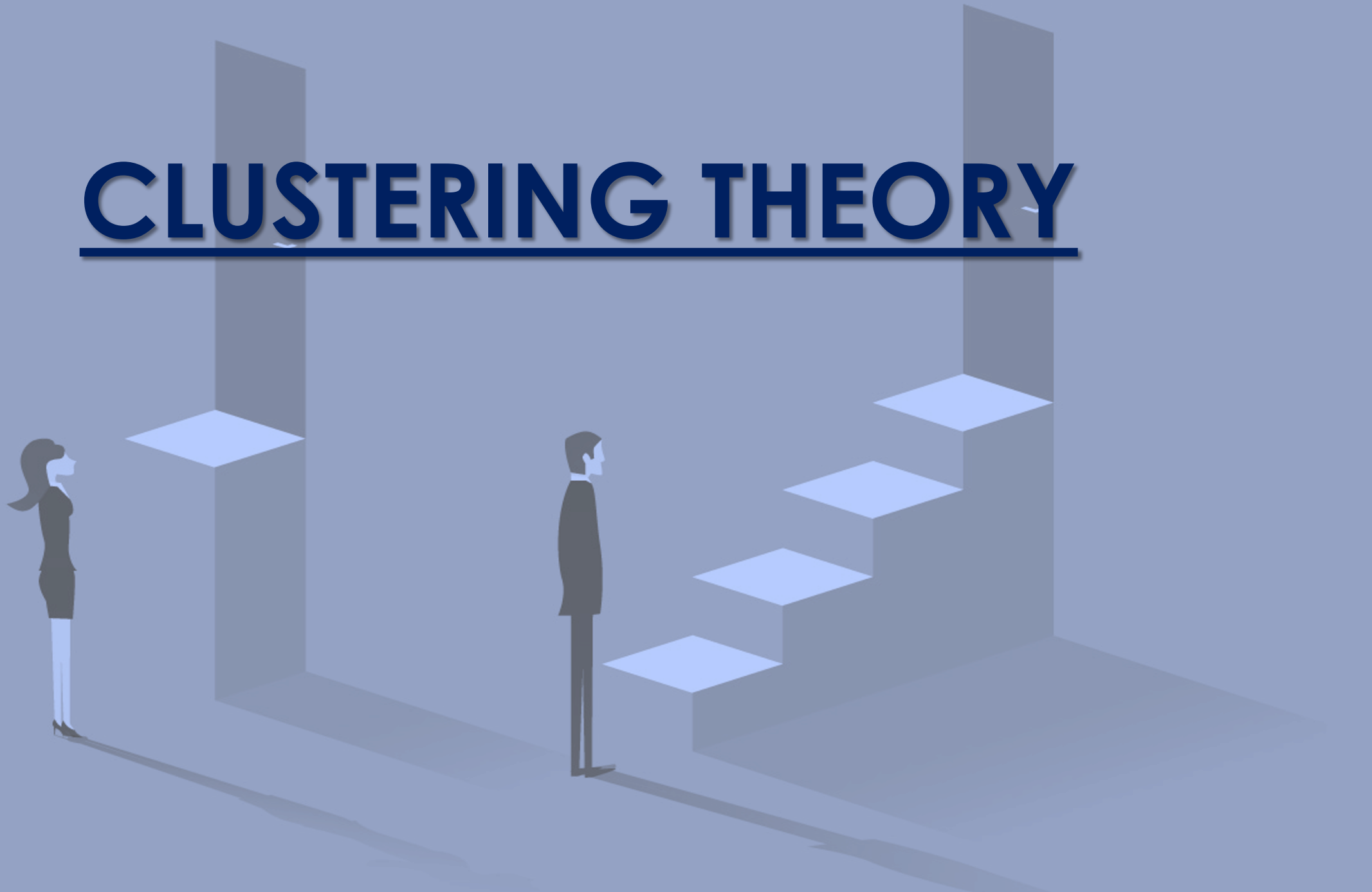
CLUSTERING GLOBAL GENDER GAP INDEX OVER TIME (2006-2019)

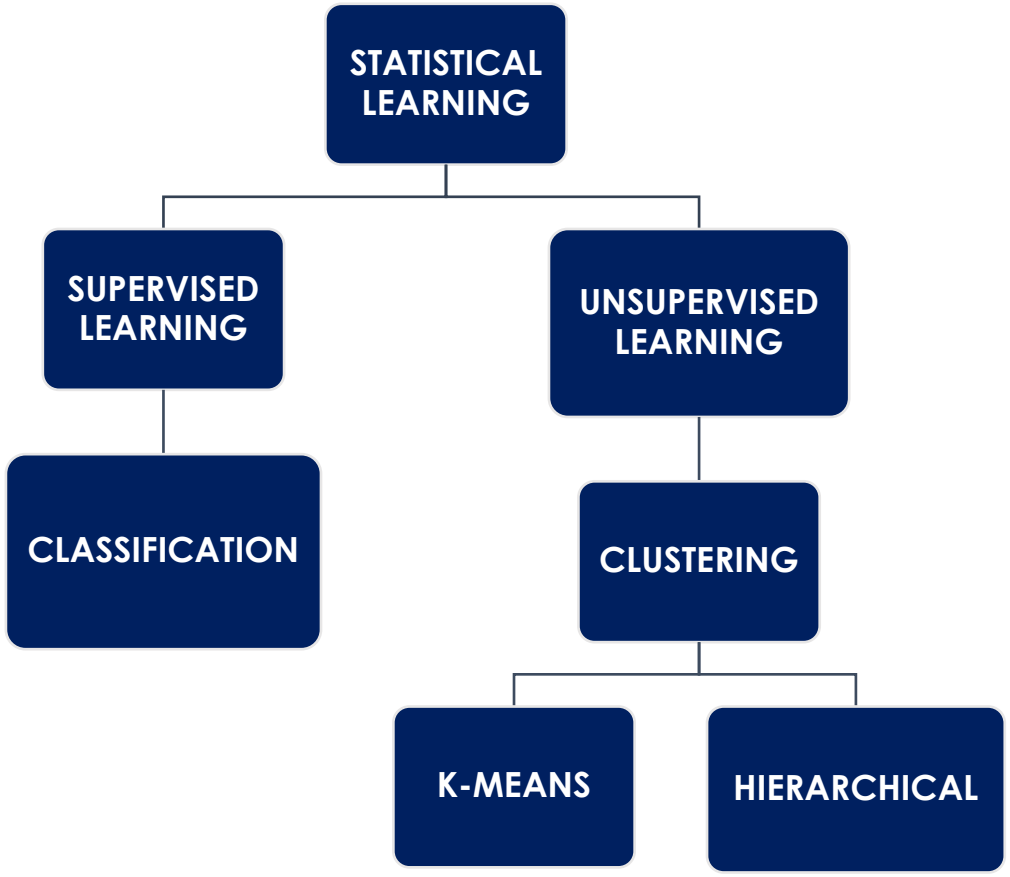
COSTA FONTICHIARI PAULA
GIULIANI MIRIAM

INDEX

CLUSTERING THEORY



INTRODUCING CLUSTERING



- **Unsupervised Learning:** The goal is to find patterns and structures in data that is not previously labeled;
- **Clustering:** a broad class of methods for discovering unknown homogeneous subgroups in data;
- **Classification X Clustering:** supervised learning process where a certain category is known regardless of the objects that can be grouped together in it;
  - **Cancer research:** subgroups among the breast cancer samples in order to obtain a better understanding of the disease;
  - **Marketing:** dividing clients in subgroups, market segmentation.

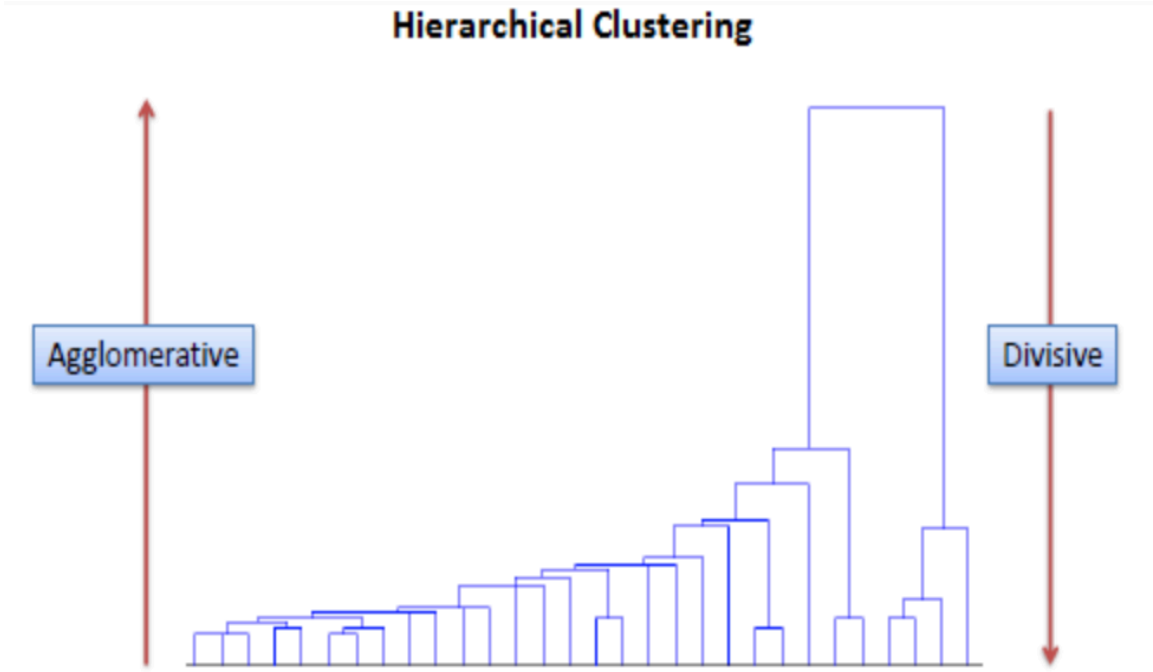
HIERARCHICAL CLUSTERING

- The number of clusters is **not predefined**
- Two different approaches:

  Agglomerative: Each observation is in a distinct (singleton) cluster, and successively merges clusters together until a stopping criterion is satisfied. (This is the one we use);

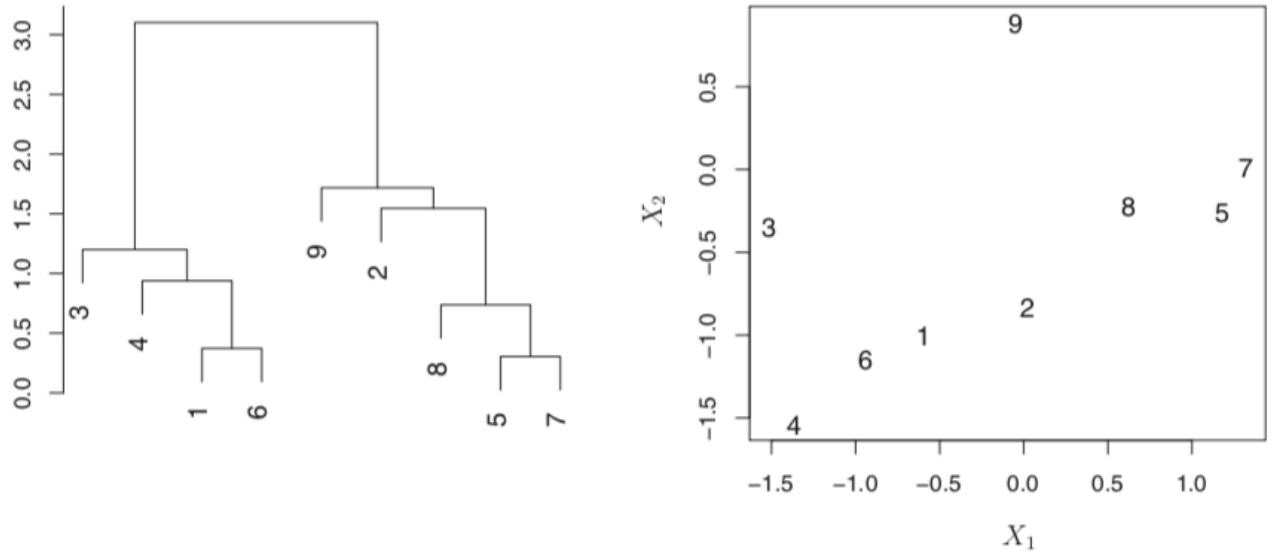
  Divisive: All patterns in a single cluster and performs splitting until a stopping criterion is met

- It results in a **Dendrogram**



DENDROGRAM

- A dendrogram is a **tree-like diagram** that shows how the observations are merged into clusters (from 1 to n)
- **Height**: distance of split or merge shown in the vertical-axis indicates how different two observations are
- The **lower** in the tree fusions occur, the **more similar** the groups of observations are to each other
- To interpret the dendrogram we should look at the height and not at the horizontal distance between observations



Intrepretation: Observation 9 is no more similar to observation 2 than it is to observations 8, 5, and 7, even though observations 9 and 2 are close together in terms of horizontal distance. This is because observations 2, 8, 5, and 7 all fuse with observation 9 at the same height, approximately.

SIMILARITY CONCEPTS

DISTANCE

Defines the similarity measure between each pair of **observations**

METHODS:

- Euclidean
- Manhattan
- Pearson correlation
- Kendall correlation

LINKAGE

Defines the similarity measure between two **clusters**, needed to merge them

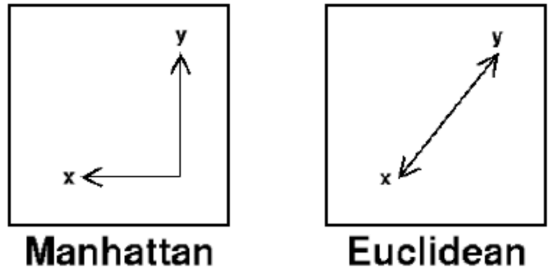
METHODS:

- Single
- Complete
- Average
- Centroid
- Ward

DISTANCES

- **Euclidean:** The "ordinary" (i.e.straight-line) distance between two points in Euclidean space. Is the most used one.

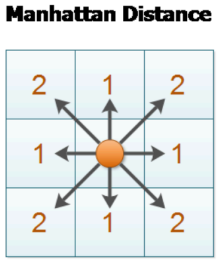
$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



Manhattan Euclidean

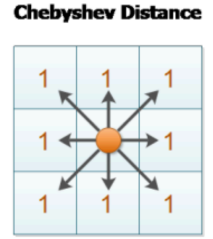
- **Manhattan:** Known as rectilinear distance, city block distance, taxicab metric is defined as the sum of the lengths of the projections of the line segment between the points onto the coordinate axes.

$$d = \sum_{i=1}^{n} |x_i - y_i|$$



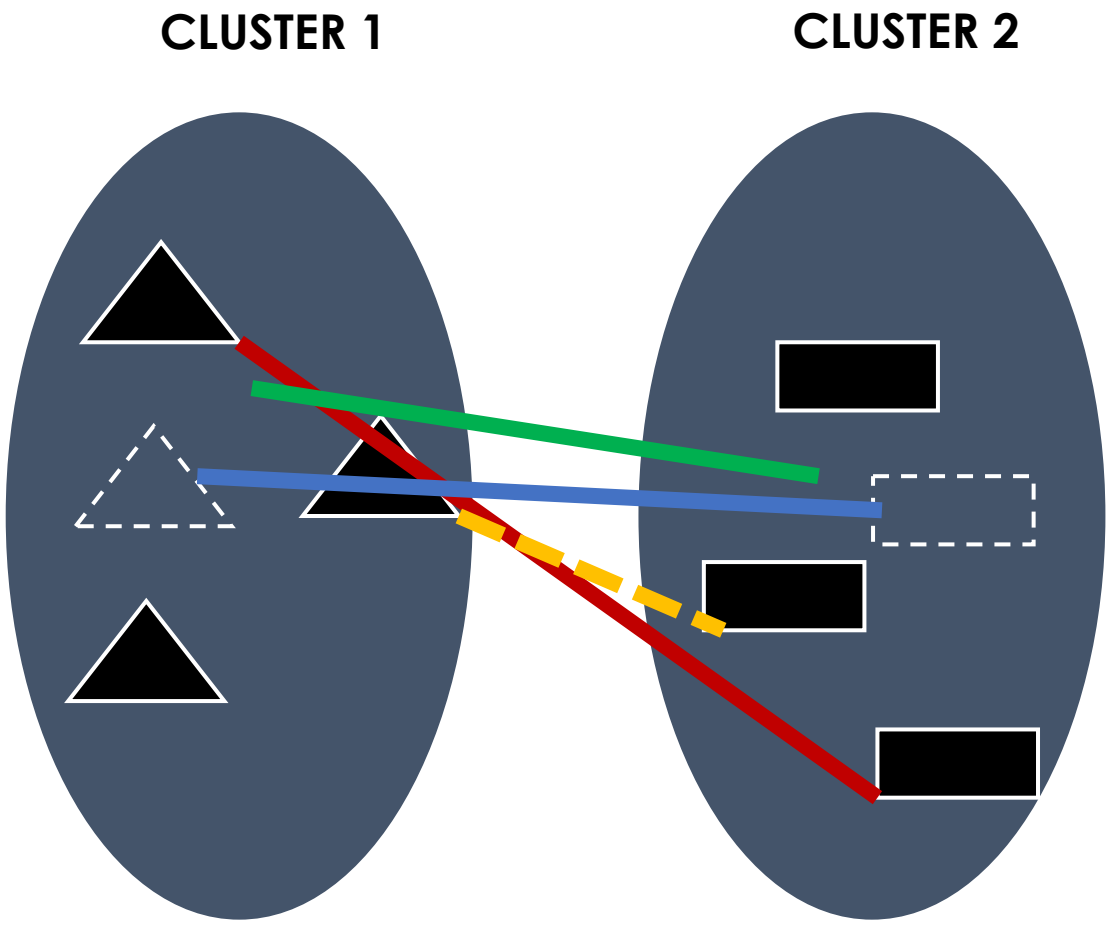
$|x_1 - x_2| + |y_1 - y_2|$

- **Chebyshev (chessboard distance):** A metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension.



$\max(|x_1 - x_2|, |y_1 - y_2|)$

- **Hamming (signal distance):** Method used for categorical data. Distance between two strings of equal length is the number of positions at which the corresponding symbols are different. In another way, it measures the minimum number of substitutions required to change one string into the other.



LINKAGE CLUSTERING METHODS	
MAXIMUM or COMPLETE	It considers the largest value of these dissimilarities as the distance between the two clusters. It tends to produce more compact clusters.
MINIMUM or SINGLE	It combines two clusters that contain the closest pair of elements not yet belonging to the same cluster as each other.It tends to produce long, "loose" clusters.
MEAN or AVERAGE	It considers the average of these dissimilarities as the distance between the two clusters.
CENTROID	It computes the dissimilarity between centroid for cluster 1 (a mean vector of length p variables) and centroid for cluster 2.
WARD'S MINIMUM VARIANCE	It minimizes the total within-cluster variance. The pair of clusters with minimum between-cluster distance are merged at each step.

HOW DOES THE ALGORITHM WORK?

1. Firstly we choose the method for the **distance** and for the **linkage**;
2. Then the algorithm proceeds **iteratively**. Starting out at the bottom of the dendrogram, each of the n observations is treated as its own cluster (agglomerative).
   - The two clusters that are most **similar** to each other are then merged so that there now are **n-1 clusters**. Next the two clusters that are most similar to each other are fused again, so that there now are **n-2 clusters**.
3. The algorithm proceeds in this way until all of the observations belong to one **single cluster**, and the dendrogram is complete.
4. Now we have to choose at which height to **cut the dendrogram**, depending on how many clusters we want.

K MEAN CLUSTERING

- We must first **specify the desired number of clusters** *K* first
- A *good* clustering is one for which the ***within-cluster variation*** ($W(C_k)$) is as small as possible, that is, we want to solve

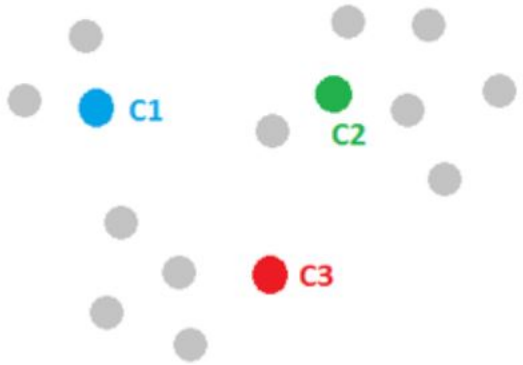
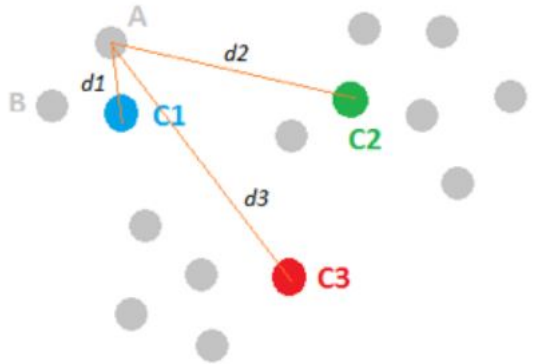
$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

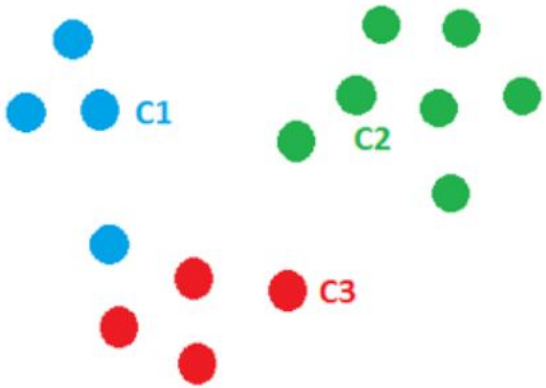
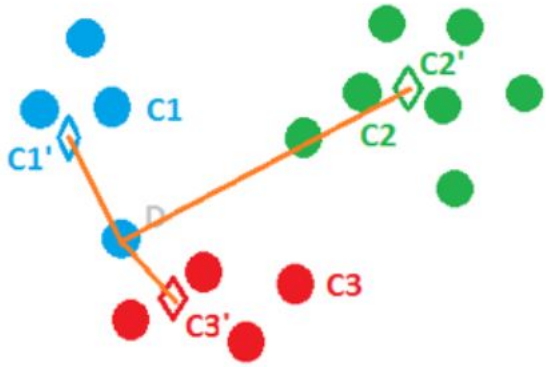
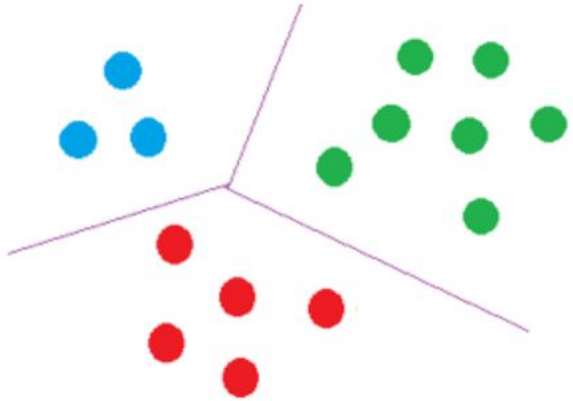
- The within-cluster variation for the *k*th cluster is the sum of all of the pairwise squared **Euclidean distances** between the observations in the *k*th cluster, divided by the total number of observations in the *k*th cluster. We want to solve

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

HOW DOES THE ALGORITM WORK?

1. First, **randomly choose** a number, from 1 to K, of the observations as initial **centroids** of the K clusters
2. Assign each observation to the cluster whose centroid is closest using Euclidean Distance. These serve as **initial cluster assignments** for the observations
3. Then, **iterate** until the cluster assignments stop changing:
    a. For each of the K clusters, compute the cluster centroid. The *kth* cluster centroid is the vector of the *p* feature (variables) means for the observations in the *kth* cluster
    b. Assign each observation to the cluster whose centroid is closest using Euclidean Distance

Data	1. Choose centroids randomly	2. Compute distances
		

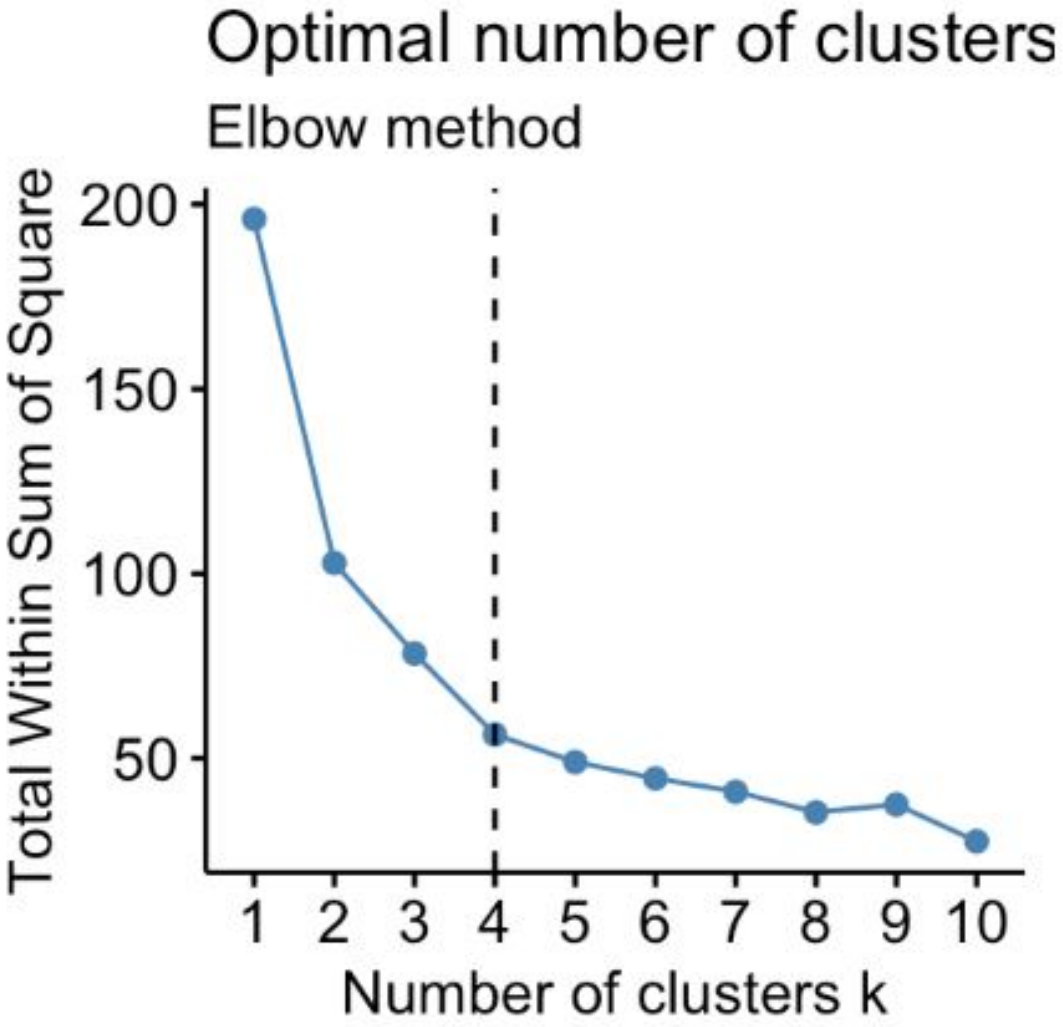
3. Assign observations to clusters	4. Compute new centroids	5. Iterate until the result doesn't change
		

- The algorithm is guaranteed to decrease the value of the within-cluster variation at each step. When the result no longer changes, a **local optimum** has been reached.

- Because the K-means algorithm finds a local rather than a global optimum, the results obtained will depend on the initial (random) cluster assignment of each observation in step 1. For this reason, it is important to **run the algorithm multiple times** from different random initial configurations. Then one selects the best solution, i.e. that for which the within-cluster variation is smallest.

HOW TO CHOOSE THE NUMBER OF CLUSTERS

ELBOW METHOD



Computes the clustering algorithm for different values of k and for each k, calculate the **total within-cluster sum of square** (WSS). Plot the curve of wss according to the number of clusters k. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

The first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph.

HOW TO VALIDATE THE CLUSTERS

SILHOUETTE METHOD

- Compute clustering algorithm for different values of k. For each k, calculate the **average silhouette of observations** (measure of how similar an object is to its own cluster compared to other clusters) .

- This measure has a range of [-1, 1]. Silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the decision boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

HIERARCHICAL X K-MEAN

	HIERARCHICAL	K-MEANS
RUNNING TIME	Slower	Faster
ASSUMPTIONS	Requires DISTANCE and LINKAGE metric	Requires DISTANCE metric
PARAMETERS	None	K (number of cluster)
CLUSTERS	Subjective (only a tree is returned)	Exactly K clusters
FOCUS	DISTANCES between observations and clusters	VALUES of observations

OUR ANALYSIS

GENDER GAP

TIME SERIES CLUSTERING

MEAUSURING THE GLOBAL GENDER GAP

- The **World Economic Forum**, an International Organization for Public-Private Cooperation, has been measuring the **Global Gender Gap Index** since 2006.
- This index varies from 0 to 1; **0** is the worst scenario and **1** indicates gender equality. It is an average of other 4 indexes that measure the gender gap on **4 different areas**:

ECONOMIC PARTICIPATION and OPPORTUNITY:

- Labour force participation rate;
- Wage equality for similar work,;
- Estimated earned income;
- % Legislators, senior officials and managers;
- % Professional and technical workers.

HEALTH and SURVIVAL:

- % Sex ratio at birth;
- Healthy life expectancy, years ;

EDUCATIONAL ATTAINMENT

- % Literacy rate;
- % Enrolment in primary education;
- % Enrolment in secondary education;
- % Enrolment in tertiary education.

POLITICAL EMPOWERMENT:

- % Women in parliament;
- % Women in ministerial positions;
- Years with female/male head of state (last 50).

OUR CLUSTERS: TIME SERIES

We want to look for clusters between different **time series**, by applying first Hierarchical and K-Mean clustering algorithms, and, in the second part of this project, a Bayesian Approach.

Usually, clusters are found for observations based on different features (variables).
For example, one may want to find patterns in terms of patients health condition, based on their physical characteristics and medical measurements.

In our case, we have just **one variable**, measured **in different time periods**. Each observation is not described by different features, but consists in a time series, characterized by a specific trend over time. With clustering we can identify time series that have a similar behaviour.

This changes the way we should look at standard clustering methods and the assumptions behind them.

THE DATA SET

We created our data set selecting the **Global Gender Gap Index** of each country **from 2006 to 2019**. The rows of our data set correspond to the different countries and the columns are the measurements of the index for the different years.

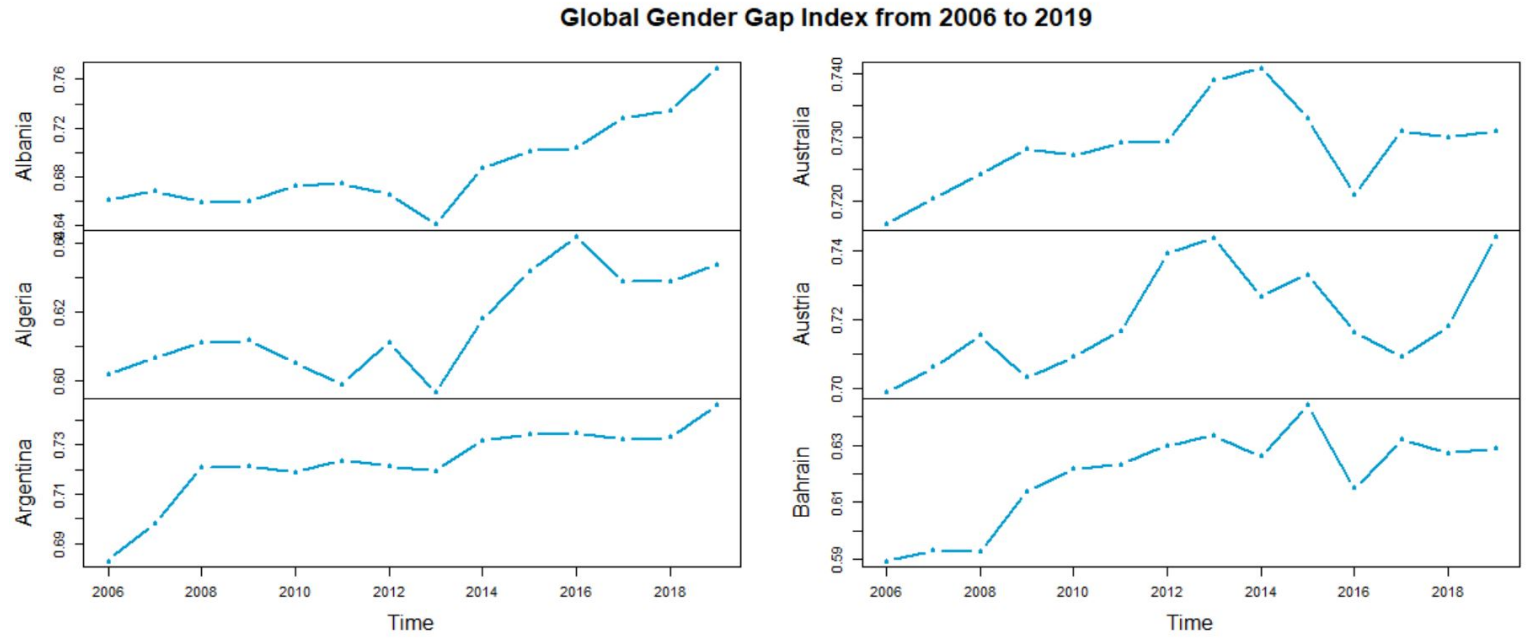
Since we want to compare the results of different algorithms and some of them don't accept NA values, we do not consider countries for which the index has not been measured in the whole period we took into consideration. The final version of our data set comprehends data about **106 countries** over **14 years**.

Since, by construction, the index is always between 0 and 1 and that did not change over the years, we don't need to standardize the data. If this is not the case, this step should be done in order to obtain valid clustering results.

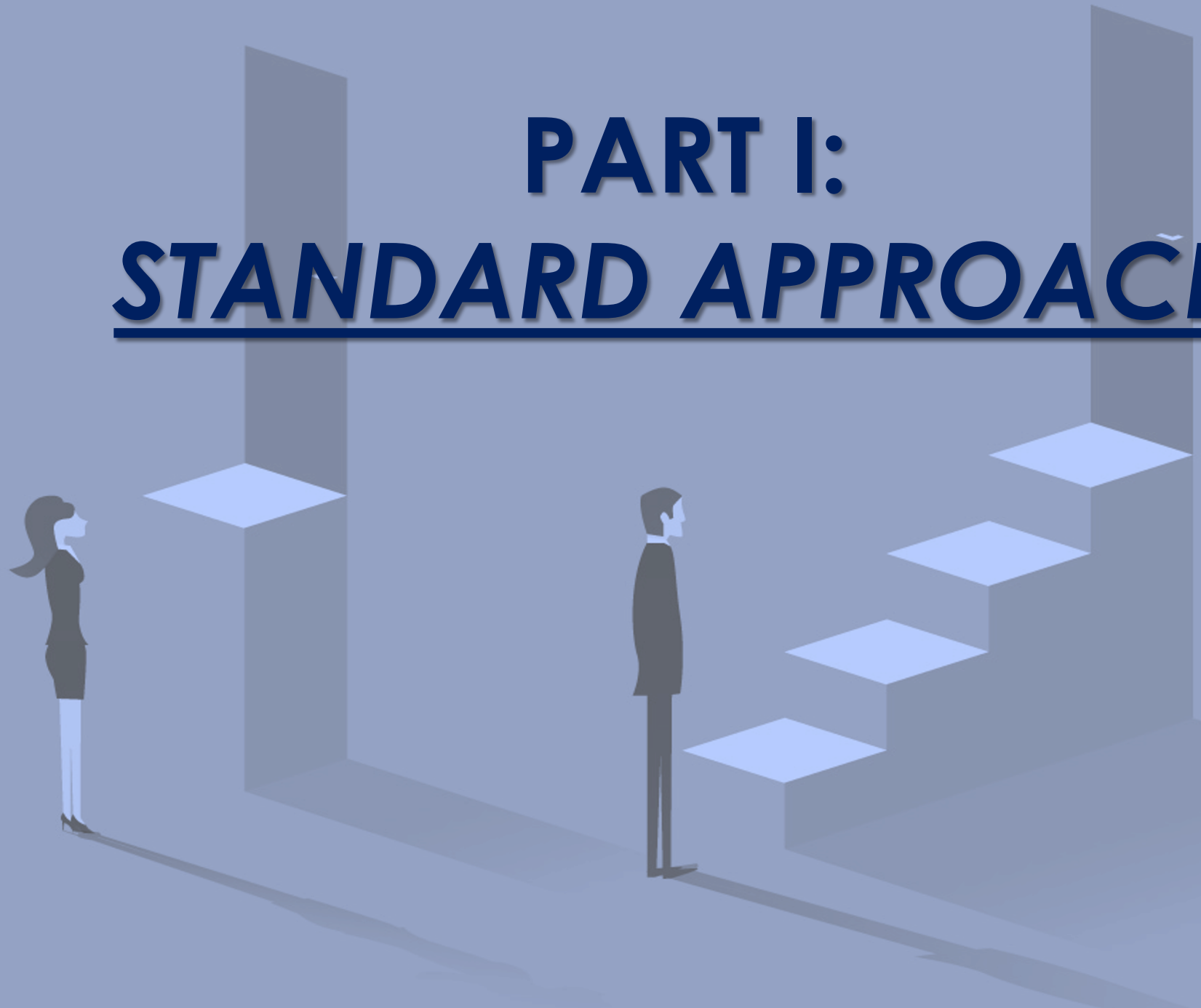
Each row of our data set corresponds to a different time series and a different country.

The **goal** of our analysis is to find patterns between countries in terms of the index trend over time.

	2006 <dbl>	2007 <dbl>	2008 <dbl>	2009 <dbl>	2010 <dbl>	2011 <dbl>	2012 <dbl>	2013 <dbl>	2014 <dbl>	2015 <dbl>	2016 <dbl>	2017 <dbl>	2018 <dbl>	2019 <dbl>
Albania	0.6607	0.6685	0.6591	0.6601	0.6726	0.6748	0.6655	0.6412	0.6869	0.701	0.704	0.728	0.734	0.769
Algeria	0.6018	0.6068	0.6111	0.6119	0.6052	0.5991	0.6112	0.5966	0.6182	0.632	0.642	0.629	0.629	0.634
Argentina	0.6829	0.6982	0.7209	0.7211	0.7187	0.7236	0.7212	0.7195	0.7317	0.734	0.735	0.732	0.733	0.746
Australia	0.7163	0.7204	0.7241	0.7282	0.7271	0.7291	0.7294	0.7390	0.7409	0.733	0.721	0.731	0.730	0.731
Austria	0.6986	0.7060	0.7153	0.7031	0.7091	0.7165	0.7391	0.7437	0.7266	0.733	0.716	0.709	0.718	0.744
Bahrain	0.5894	0.5931	0.5927	0.6136	0.6217	0.6232	0.6298	0.6334	0.6261	0.644	0.615	0.632	0.627	0.629

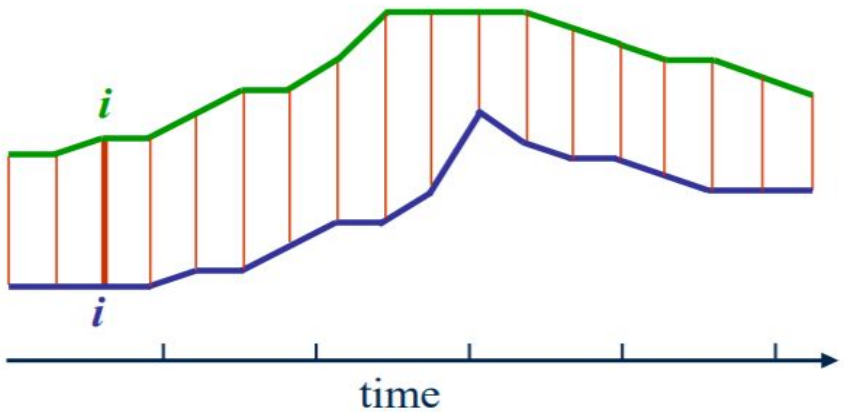


PART I:
STANDARD APPROACH

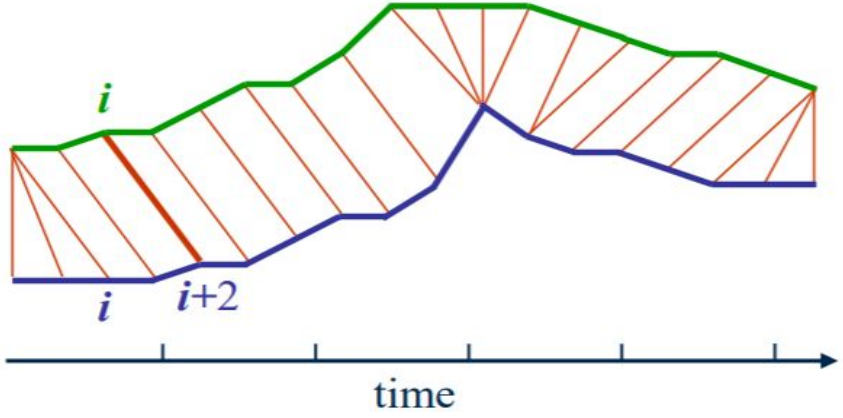


DYNAMIC TIME WARPING (DTW)

- Algorithm that measures similarity between **two temporal sequences**;
- Allows two curves to match up evenly even though the X-axes (i.e. time) are **not necessarily in sync**.
- Another way is to think of this is as a **robust dissimilarity score** where a lower number means the series is more similar.

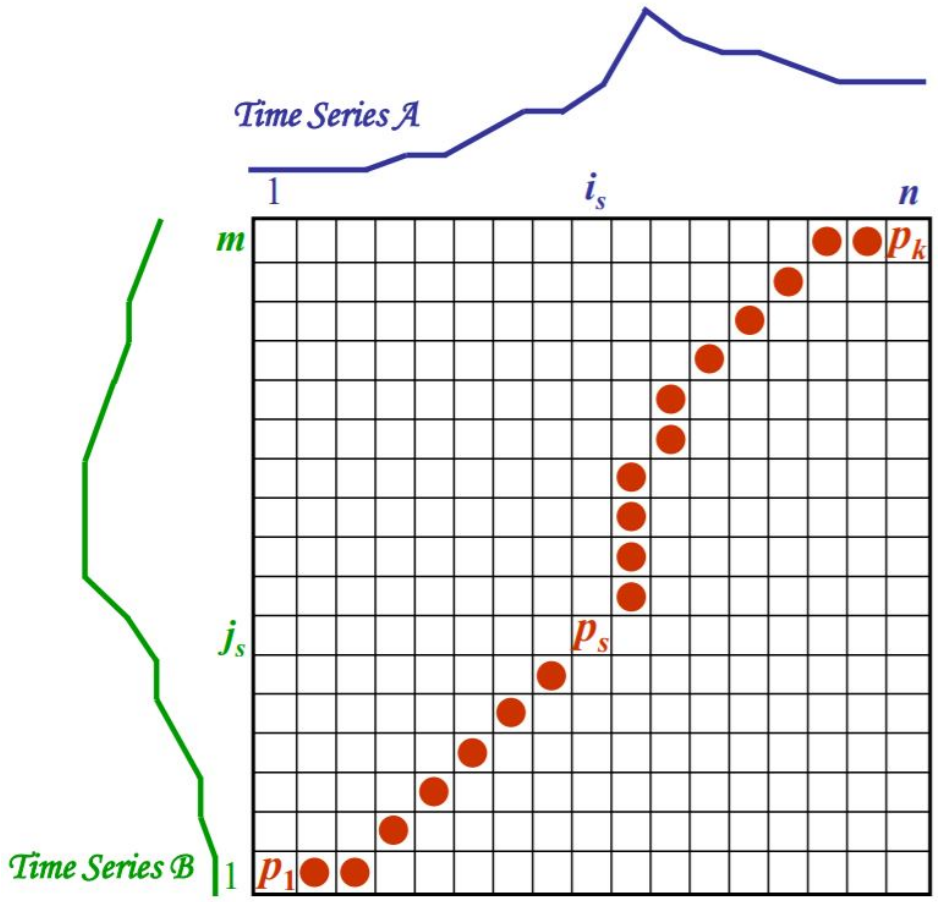


Euclidean distance, which aligns the *i-th* point on one time series with the *i-th* point on the other (poor similarity score)



A **non-linear** (elastic) alignment produces a more intuitive similarity measure, allowing similar shapes to match even if they are out of phase in the time axis

HOW TO COMPUTE DTW



To find the best alignment between A and B one needs to find the **path** through the grid which minimizes the total distance between them. P is called a **warping function**.

1. Divide the two series (A and B) into equal points.
2. Calculate the Euclidean distance between the first point in the series A and every point in series B. Store the minimum distance computed (*time warp stage*)
3. Move to the second point and repeat 2. Move step by step along points and repeat 2 till all points are exhausted.
4. Repeat 2 and 3 but with the second series as a reference point.
5. Add up all the minimum distances that were stored and this is a true measure of similarity between the two series.

DTW: RULES

In general, **DTW** is a method that calculates an optimal match between two given sequences (e.g. time series) with certain restriction and rules:

- **No left out: every index** from the first sequence must **be matched** with **one or more indices** from the other sequence and vice versa;
- **Head and Tail positionally matched**: the **first index (last index)** from the first sequence must **be matched** with the **first index (last index)** from the other sequence (but it does **not** have to be its **only match**);
- **No cross-match**: the mapping of the indices from the first sequence to indices from the other sequence must be **monotonically increasing**, and **vice versa**;

The **optimal match satisfies** all the **restrictions** and the **rules** and has the **minimal cost**(the sum of absolute differences, for each matched pair of indices, between their values).

HIERARCHICAL

Firstly we create de distance matrix with the function **dist**, choosing the best distance method, in this case we chose the Dynamic Time Warping present in the **dtw package**;

We implemented the Hierarchical algorithm using the function `hclust()`

that has as arguments:

- **d:** we insert the distance matrix previously calculated;
- **method:** we specify which kind of linkage method we are going to use;

Then to visualize the dendrogram we use the function **plot**;

To cut the dendrogram in k clusters you can either use `cutree()` or `rect.hclust()` ;

There's an alternative way using the **dtwclust package** with the function **tsclust**(); which has as arguments the data**, type** of clustering(hierarchical), **method** (dtw), **k** (number of clusters).

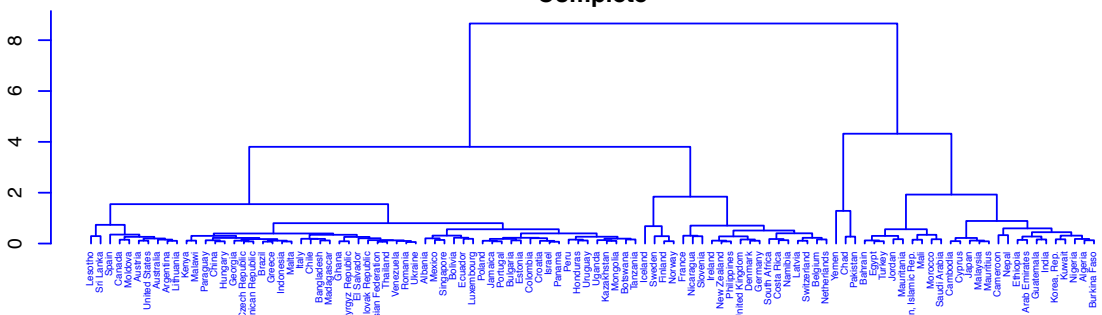
IMPLEMENTATION

- Since we're dealing with **time series** we found, as previously explained, that **DTW** is an useful metric for measuring distance between this type of observations;
- In order to use DTW metric we should use as linkage method a simple one: Single, Complete or Average;
- **Ward** linkage also may give reasonable results in practice, although I would be **cautious of relying** on it exclusively because of the **ambiguity** surrounding the meaning of a **centroid** in the context of DTW similarity measures;
- We did an analysis with both **DTW** and **Euclidean** distances, since we know that the algorithm is really sensitive to the metrics chosen to define dissimilarity;
- We used different methods of linkage: Complete, Single and Average;

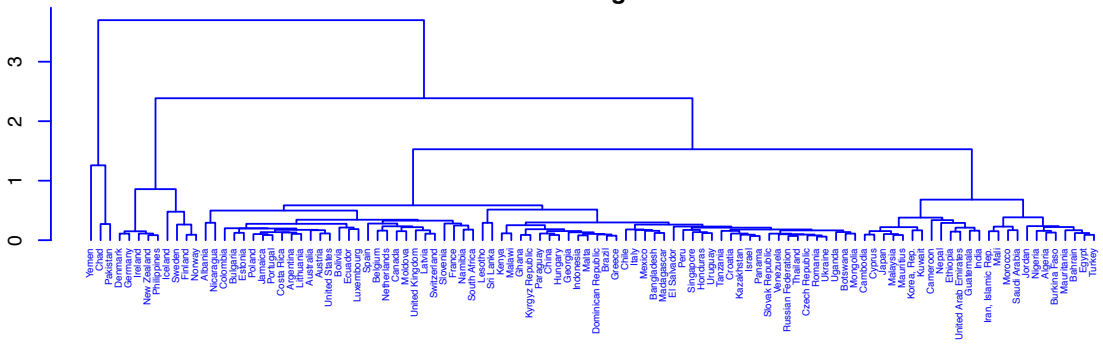
DENDROGRAMS

Distance: DTW

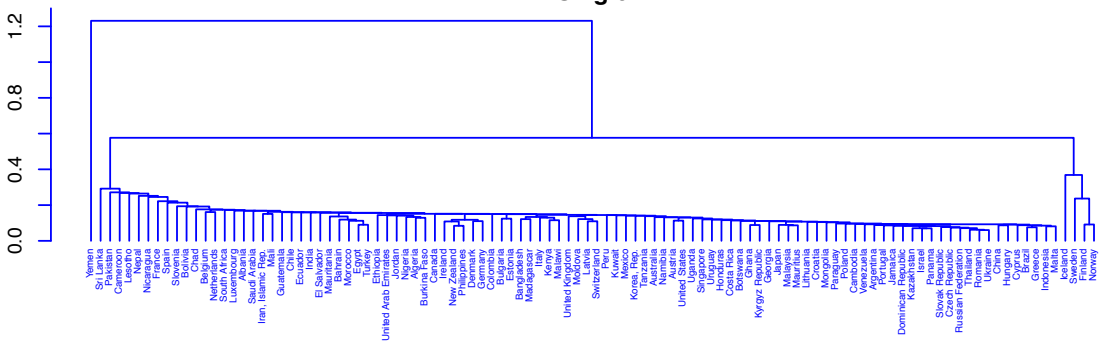
Complete



Average

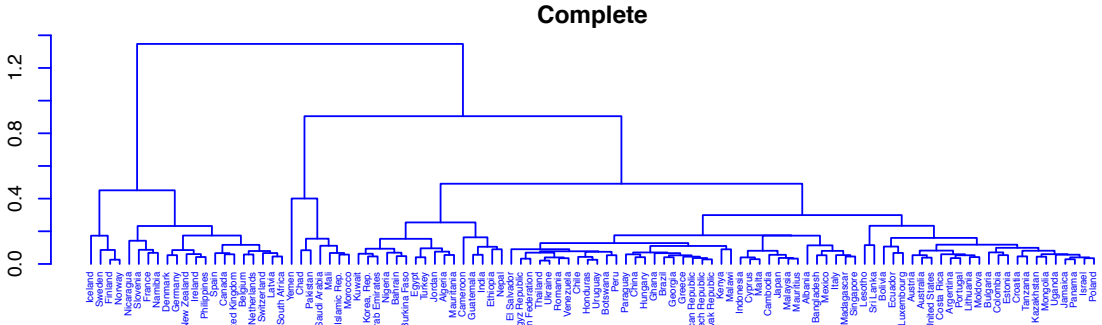


Single

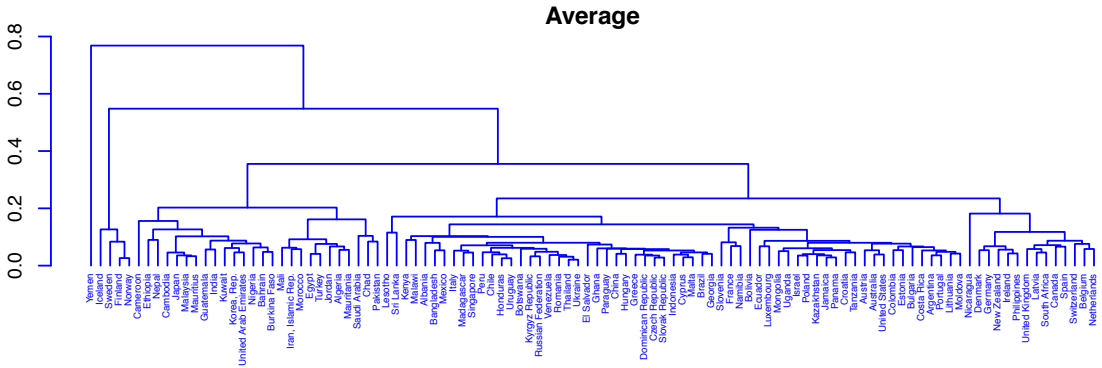


Distance: Euclidean

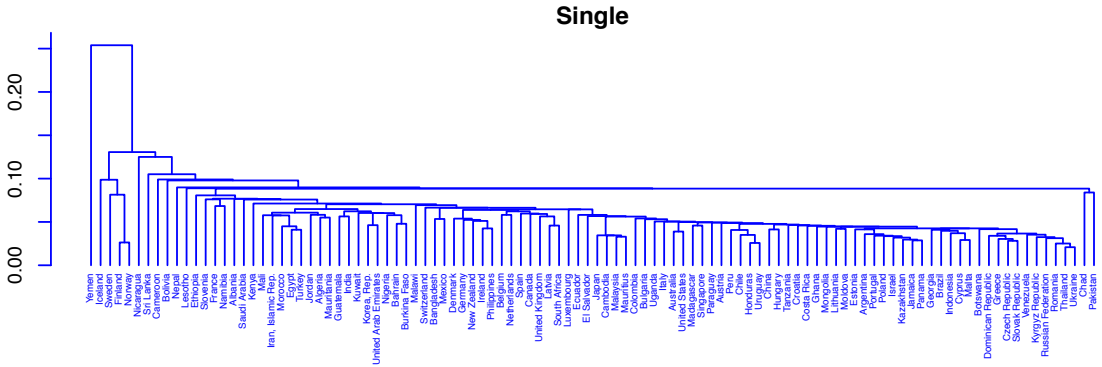
Complete



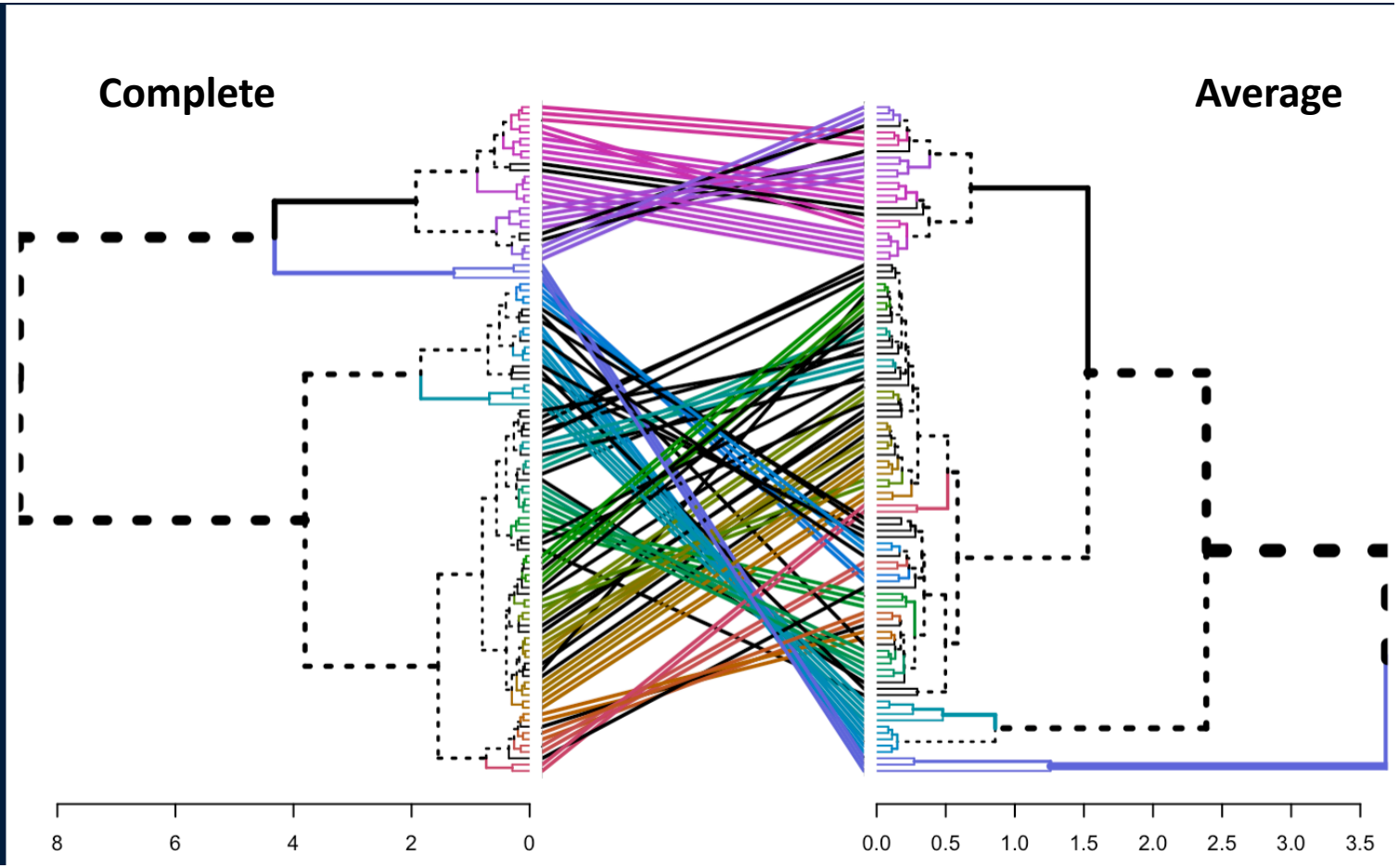
Average



Single



COMPARING DENDROGRAMS



The Fuction **tanglegram( )** plots the two dendrograms, side by side, with their labels connected by lines;

The output displays "unique" nodes, with a combination of labels/items not present in the other tree, highlighted with dashed lines.

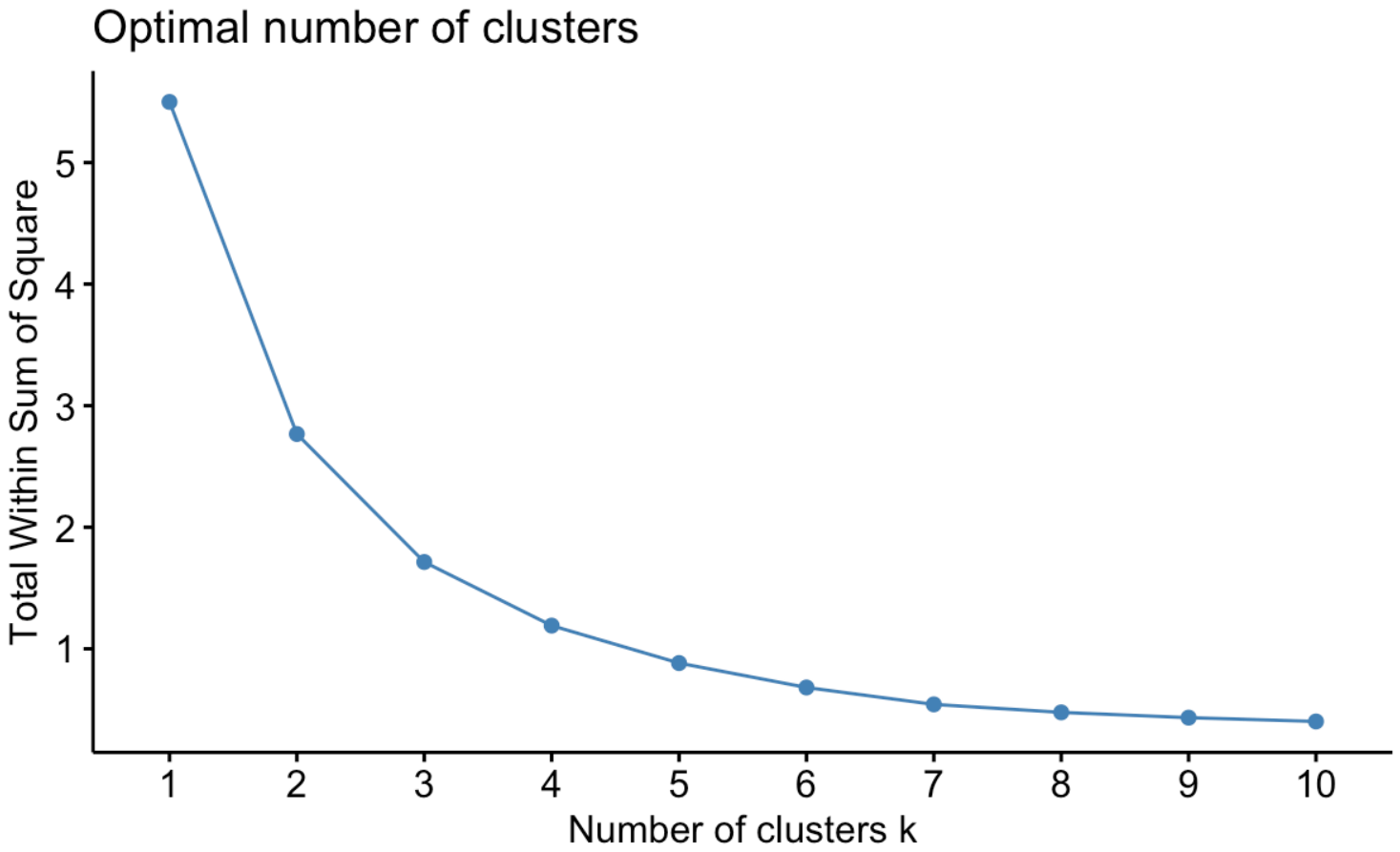
By using different linkage methods, the results may change.

```
[1] 0.3991899
```

Entaglement coefficient: measure between 1 (full entanglement) and 0 (no entanglement); A lower entanglement coefficient corresponds to a good alignment.

VALIDATION

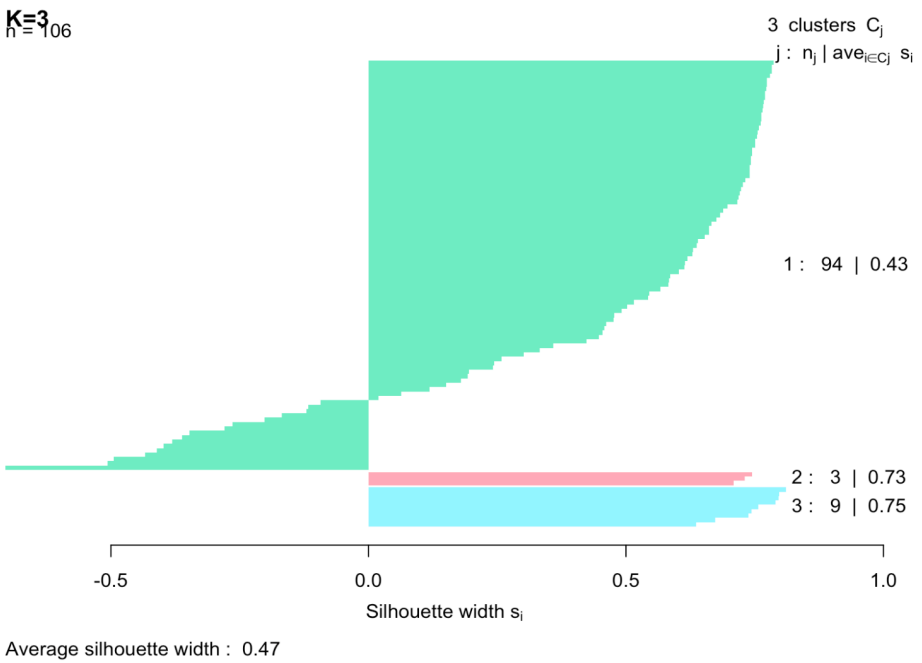
We apply the **Elbow method**. As we can see, the Total Within Sum of Squares decreases significantly for values of k from 1 to 3/4, but after that the variation is smaller. We conclude that our optimal value of k could be **4** or **3**.

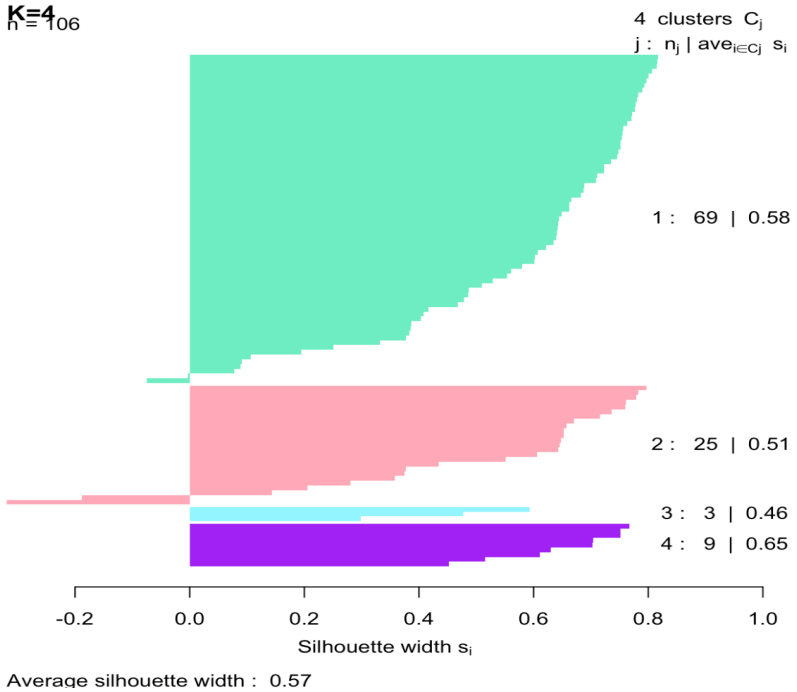


VALIDATION

From the output of the **Silhouette analysis**, we can confirm what we previously found with the Elbow method, since the higher value of the Average Silhouette Width corresponds to K = 4.

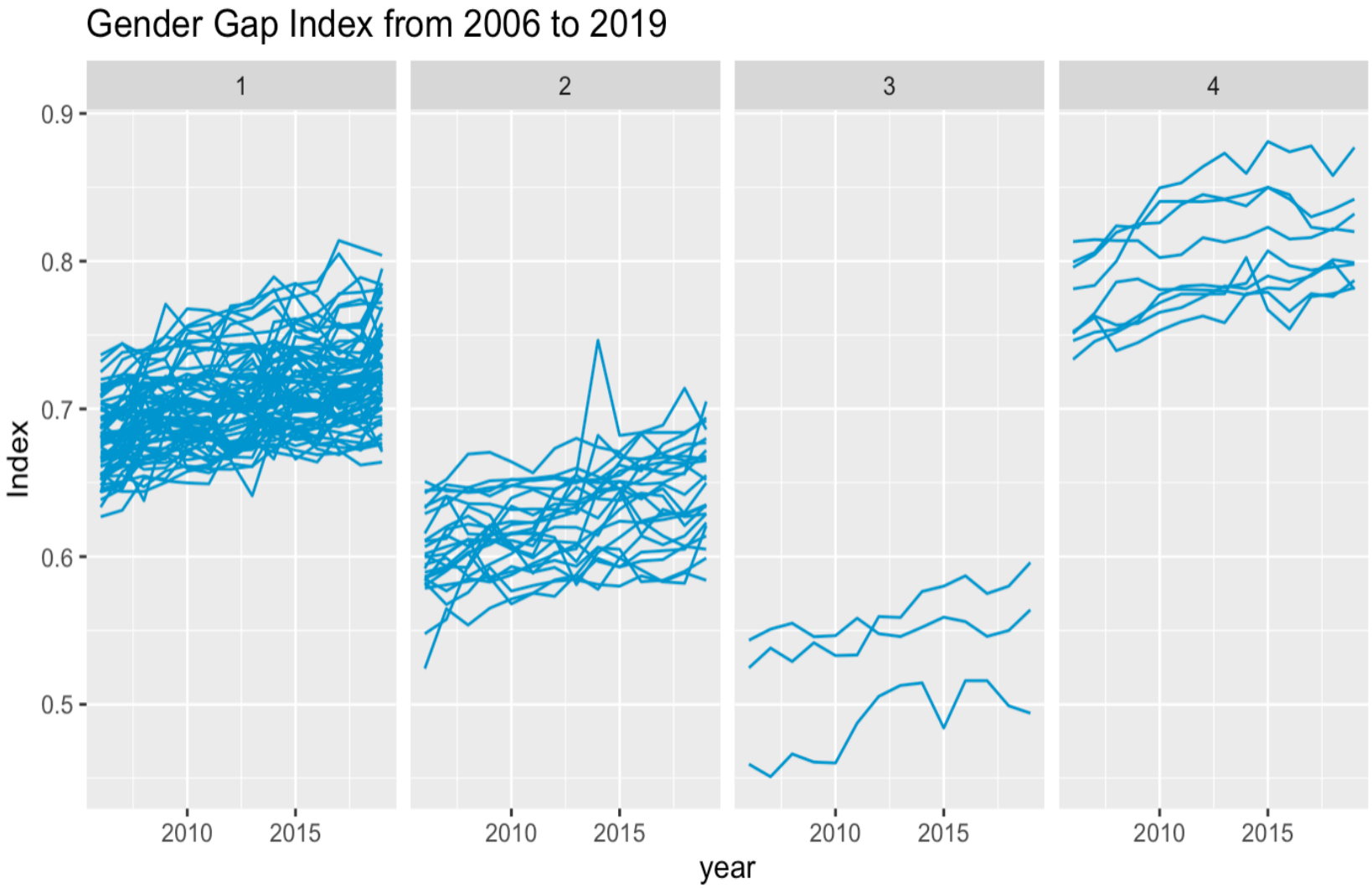
K = 3	K = 4
0.47	**0.57**





CHOOSING

- **Distance metric**: DTW (time series)
- **Linkage method**: Average
- **Number of clusters**: 4



CLUSTERS (HIERACHICAL, DTW - AVERAGE)

CLUSTER 1				CLUSTER 2		CLUSTER 3	CLUSTER 4
Albania	Estonia	Malta	Switzerland	Algeria	Nepal	Chad	Denmark
Argentina	France	Mexico	Tanzania	Bahrain	Nigeria	Pakistan	Finland
Australia	Georgia	Moldova	Thailand	Burkina Faso	Saudi Arabia	Yemen	Germany
Austria	Ghana	Mongolia	Uganda	Cambodia	Turkey		Iceland
Bangladesh	Greece	Namibia	Ukraine	Cameroon	United Arab Emirates		Ireland
Belgium	Honduras	Netherlands	United Kingdom	Cyprus			New Zealand
Bolivia	Hungary	Nicaragua	United States	Egypt			Norway
Botswana	Indonesia	Panama	Uruguay	Ethiopia			Philippines
Brazil	Israel	Paraguay	Venezuela	Guatemala			Sweden
Bulgaria	Italy	Peru		India			
Canada	Jamaica	Poland		Iran			
Chile	Kazakhstan	Portugal		Japan			
China	Kenya	Romania		Jordan			
Colombia	Kyrgyz Republic	Russia		Korea, Rep.			
Costa Rica	Latvia	Singapore		Kuwait			
Croatia	Lesotho	Slovak Republic		Malaysia			
Czech Republic	Lithuania	Slovenia		Mali			
Dominican Republic	Luxembourg	South Africa		Mauritania			
Ecuador	Madagascar	Spain		Mauritius			
El Salvador	Malawi	Sri Lanka		Morocco			

K MEAN AND TIME SERIES

When choosing the algorithm to apply to our data set, we tried to understand how the results of K Mean could be interpreted and compared with the Hierarchical output. The two algorithms are similar, but use different concept of dissimilarities between observations; while Hierarchical clustering focuses more on distance between observation and clusters, K Mean uses the concepts of variance and mean of observation features.

In a regular setting, this difference usually doesn't affect much the results, but since we are working with special objects, such as time series, the way the concept of similarity is defined becomes relevant.

The direct application of K Mean to our data set raises two **main issues**:

- When assigning the observations to clusters, the algorithm minimizes the within cluster variation, which involves the concept of **Euclidian distance** between observation. As we have seen before, this metric is not a good fit when defining a similarity criteria between time series.

- When defining the centroids of clusters, the algorithms computes a **mean** between the observations. This step doesn't have much sense if we are interested in clustering time series based on their trends over time, since the algorithm considers as different variables the measurement of the same index in different time periods. The focus should be more on the distance between time series, than on the atomic value of the index each year.

K-MEDOIDS

We can find a good compromise in the **K Medoids** method.

With K Mean clustering, the centroid of a cluster is computed as the mean value of all the data points in the same cluster. In contrast, the K Medoids chooses data points as centers of the clusters. This points are called **medoids** and are objects for which the average distance (dissimilarity) with respect to all the other observations of the same cluster is minimal (that is, it corresponds to the most centrally located object in the cluster).

Since the algorithm no longer computes the mean in order to find the centroids, the only information needed is the distance between each observations and the others. The possibility to use directly a distance matrix of choice fits better our setting, for which the Euclidian distance results inaccurate.

HOW DOES THE ALGORITHM WORK?

1. Randomly **select K** observations as initial medoids

2. Compute the **dissimilarity matrix** and **assign every observation** to its closest medoid

3. For each cluster obtained, search if any of the objects in the cluster **decreases the average dissimilarity coefficient** if chosen as medoid.

4. If it does, select the observation that decreases the coefficient the most as **new medoid** of the cluster. If it doesn't, the algorithm can stop.

5. If at least one medoid of the original clusters has changed, go to Step 2, else the algorithm ends.

K-MEAN X K-MEDOIDS

Other than the **choice of the distance matrix**, the major advantage of K medoids is the robustness of the result with respect to **outliers**, since the centers of the clusters are observations and not averages.

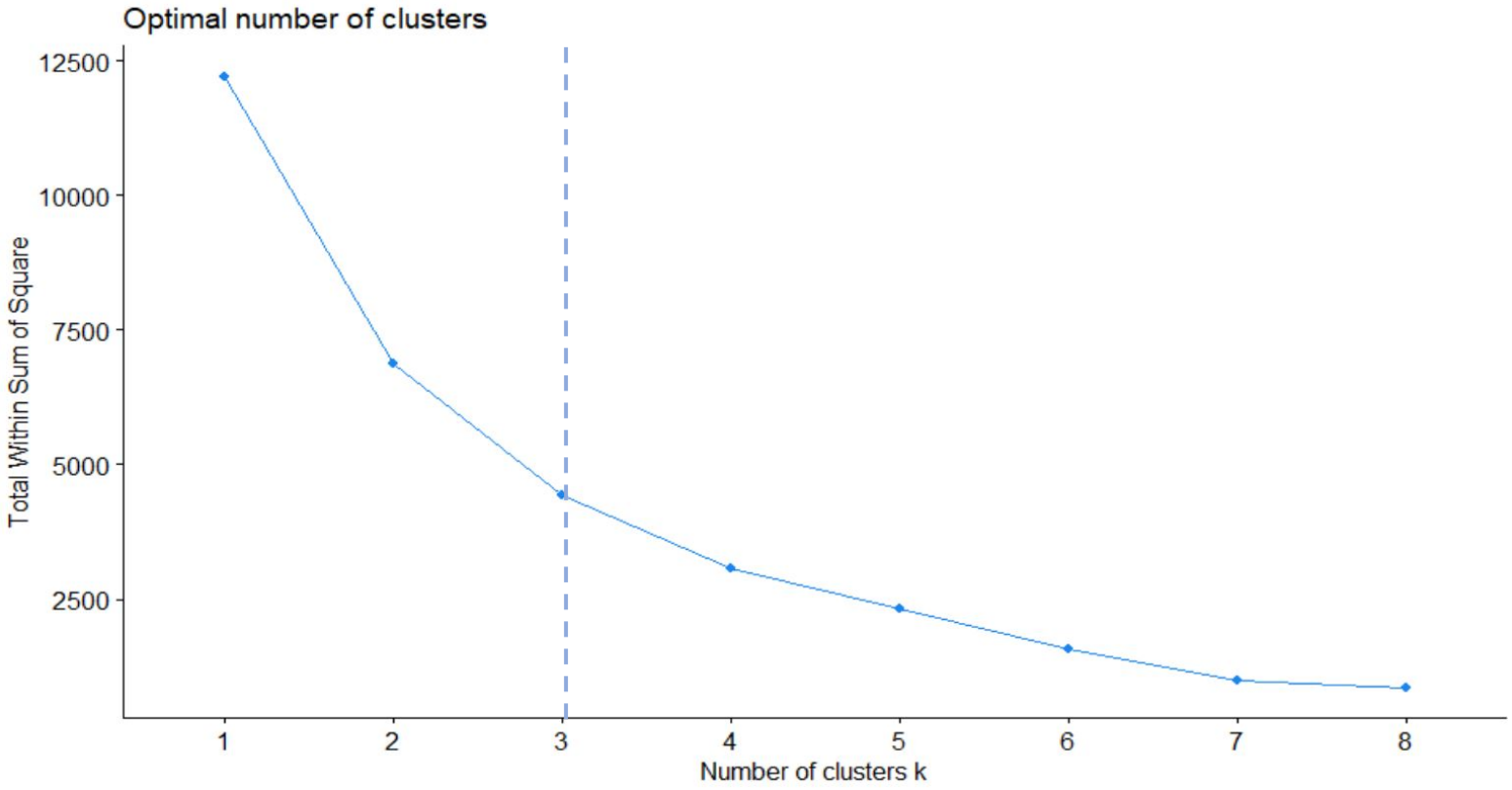
For instance, if we want to cluster a set of people based on the number of products they bought in a store and we have the following measurements (3, 110, 105, 101, 114), K Means computes as the center of the cluster the mean of these values, 86.6. This value is less than the measured value for most of our observations and it's affected by the presence of an outlier (3).
Applying K Medoids algorithm the center would be defined based on distances. We would compute the distance between measurement 3 and the others: (110-3)+(105-3)+(101-3)+(114-3) = 418 and so on for all the other observations. After these calculations, we would choose 105 as medoid, which is a better result than the K mean centroid, since it's less affected by the outlier.

K-MEAN X K-MEDOIDS

	K-MEAN	K-MEDOIDS
RUNNING TIME	Faster	Slower
INFORMATION NEEDED	Requires DATA matrix	Can work also only with DISTANCE matrix
PARAMETERS	K (number of cluster)	K (number of cluster)
CENTERS	Means	Data objects
FOCUS	Values	Distances

NUMBER OF CLUSTERS

As we have seen before for the K-Mean, also this algorithm requires a predefined number of clusters K; since we are in an unsupervised environment, we do not know how many clusters we could find clustering our time series.



We apply the **Elbow method**. As we can see, the Total Within Sum of Squares decreases significantly for values of k from 1 to 3, but after that the variation is smaller. We conclude that our optimal value of k is **3**.

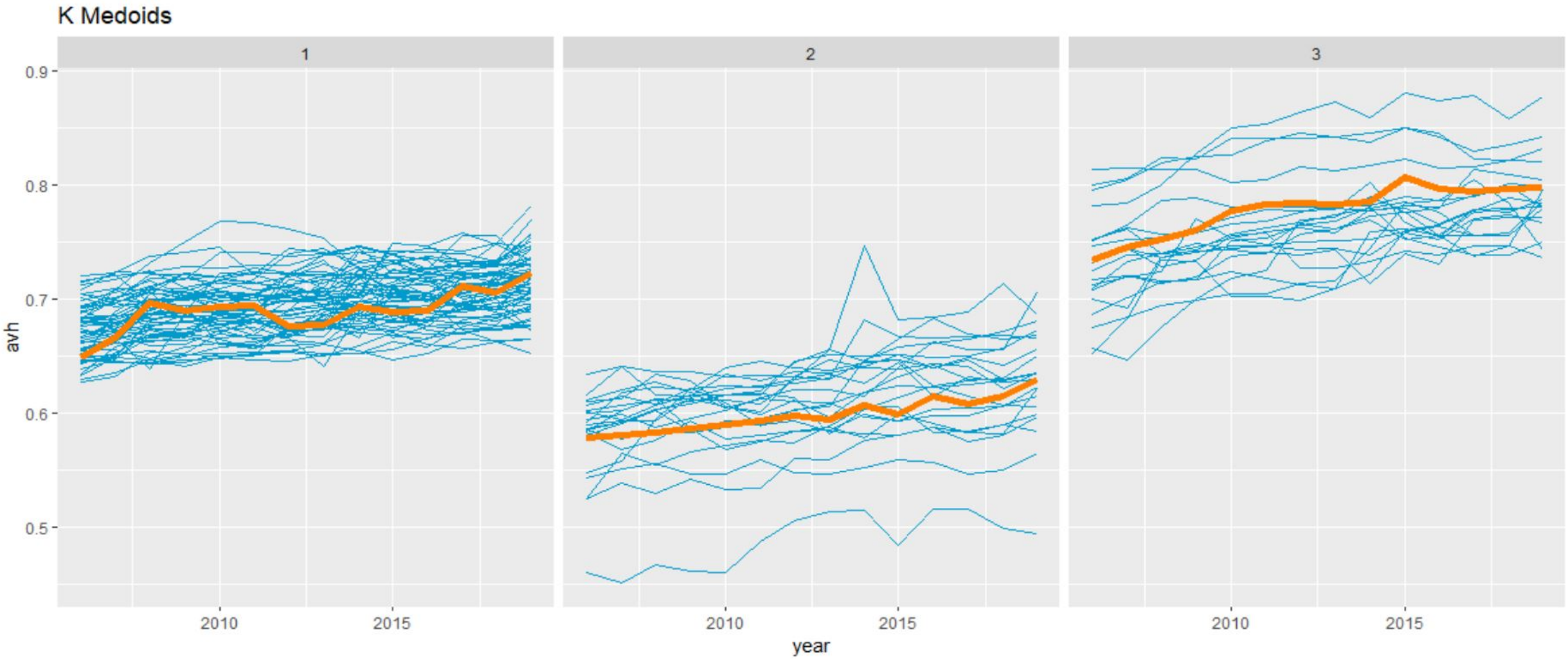
IMPLEMENTATION

Using the same data considered for the Hierarchical case, we implemented the K Medoids algorithm using the function `pam` (from the **cluster** package), which accepts as principal arguments a numeric matrix of the data to be clustered or a distance matrix of choice, the predefined number of clusters k (**k** = 3 in our case) and the **metric** of choice to compute dissimilarities if the first argument is a matrix of data. Since the starting points are chosen randomly, it's recommended to use **set.seed()** before implementing the function in order to obtain stable results.

This time, we didn't use directly the data set; we first computed the **DTW distance matrix** and use it as first argument in the `pam` function.

We obtained 3 clusters of sizes 21, 23 and 62. The medoids of the final result of the algorithm are Honduras, Egypt and Ireland. Recall that in the K Mean output we would have obtained means as centroids; here, the center of each cluster corresponds to a data object.

Visualization of the clusters: the blue lines represent the times series of countries in the same cluster. The orange line correspond to the medoids (Honduras, Egypt and Ireland).

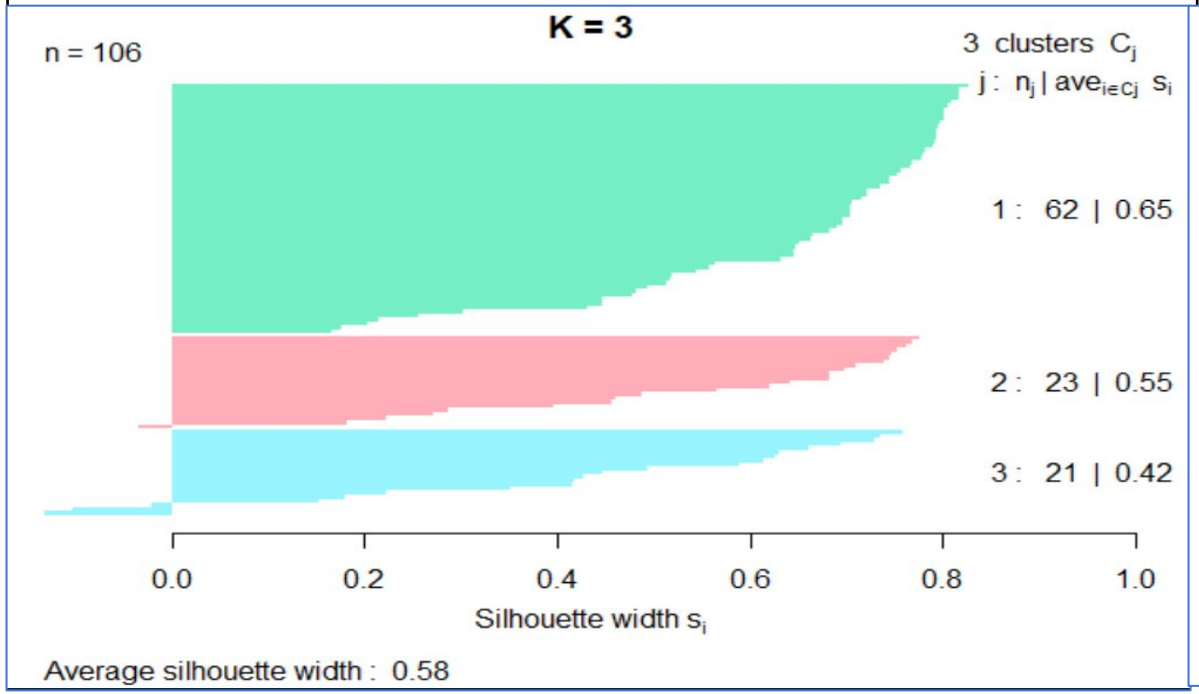


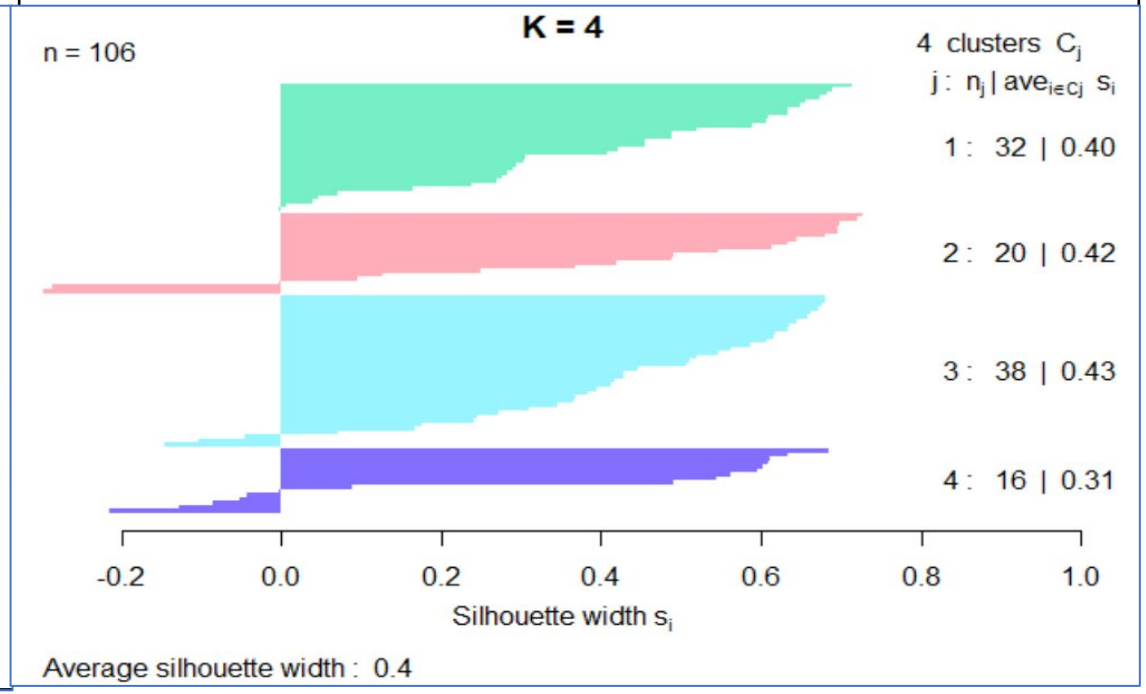
CLUSTER 1			CLUSTER 2	CLUSTER 3
Albania	Greece	Panama	Algeria	Belgium
Argentina	Honduras	Paraguay	Bahrain	Canada
Australia	Hungary	Peru	Burkina Faso	Denmark
Austria	Indonesia	Poland	Cameroon	Finland
Bangladesh	Israel	Portugal	Chad	France
Bolivia	Italy	Romania	Egypt	Germany
Botswana	Jamaica	Russian Fed.	Ethiopia	Iceland
Brazil	Japan	Singapore	Guatemala	Ireland
Bulgaria	Kazakhstan	Slovak Rep.	India	Latvia
Cambodia	Kenya	Sri Lanka	Iran, Islamic Rep.	Namibia
Chile	Kyrgyz Rep	Tanzania	Jordan	Netherlands
China	Lesotho	Thailand	Korea, Rep.	New Zealand
Colombia	Lithuania	Uganda	Kuwait	Nicaragua
Costa Rica	Luxembourg	Ukraine	Mali	Norway
Croatia	Madagascar	United States	Mauritania	Philippines
Cyprus	Malawi	Uruguay	Morocco	Slovenia
Czech Republic	Malaysia	Venezuela	Nepal	South Africa
Dominican Rep	Malta		Nigeria	Spain
Ecuador	Mauritius		Pakistan	Sweden
El Salvador	Mexico		Saudi Arabia	Switzerland
Estonia	Moldova		Turkey	United Kingdom
Georgia	Mongolia		United Arab Emirates	
Ghana			Yemen	

VALIDATION

Also in this case, from the output of the **Silhouette analysis**, we can confirm what we previously found with the Elbow method, since the higher value of the Average Silhouette Width corresponds to K = 3.

K = 3	K = 4
0.58	0.4





VALIDATION BETWEEN AND WITHIN METHODS

With function **cvi** from *dtwclust* package, we are able to compute 7 different index useful to compare different methods and different cluster choices within the same method.

CVI INDEXES		
Sil	Silhouette index	to be maximized
SF	Score Function	to be maximized
CH	Calinski-Harabasz	to be maximized
DB	Davies-Bouldin	to be minimized
D	Dunn index	to be maximized
DB*	Modified Davies-Bouldin	to be minimized
COP	COP index	to be minimized

Optimal Number of Clusters depending on the Clustering Method

HIERARCHICAL							
Number of clusters	SIL	SF	CH	DB	DBstar	D	COP
4	0.567	0.238	59.6	0.643	0.733	0.046	0.067
3	0.469	0.249	32.5	0.472	0.502	0.036	0.127

- **Hierarchica**l algorithm should be run with **4 clusters**

K-MEDOIDS							
Number of clusters	SIL	SF	CH	DB	DBstar	D	COP
4	0.379	0.297	56.08	1.528	2.213	0.023	0.070
3	0.58	0.359	95.2	0.693	0.827	0.029	0.071

- **K Medoids** algorithm should be run for **3 clusters**

Best Clustering Method depending on the number of Clusters

3 CLUSTERS							
Method	SIL	SF	CH	DB	DBstar	D	COP
Hierarchical	0.469	0.249	32.5	0.472	0.502	0.036	0.127
K-Medoids	0.58	0.359	95.2	0.693	0.827	0.029	0.071

- **K medoids** should be chosen if **3 clusters** are computed;

4 CLUSTERS							
Method	SIL	SF	CH	DB	DBstar	D	COP
Hierarchical	0.567	0.238	59.6	0.643	0.733	0.046	0.067
K-Medoids	0.379	0.297	56.08	1.528	2.213	0.023	0.070

- **Hierarchical** clustering should be chosen if **4 clusters** are computed

CONCLUSION OF PART I

- The resulting clusters, both from the hierarchical and k-medoids method, are **reasonable** considering the rankings of the World Economic Forum reports;
- It's possible to see that countries with very high index such as Iceland, Norway, Finland and Sweden were found in the same cluster;
- **Ranking** the Clusters from **top to worst** perfoming countries:

HIERARCHICAL		K-MEDOIDS	
Ranking	**Cluster**	**Ranking**	**Cluster**
1	Cluster 4	1	Cluster 3
2	Cluster 1	2	Cluster 1
3	Cluster 2	3	Cluster 2
4	Cluster 3	-	-

- Even with different number of clusters the two algorithms are **coherent** in the results. For instance, Cluster 3 in the k-medoids output includes the majority of Cluster 2 and the whole Cluster 3 from the Hierarchical output. Also Clusters 1 from both methods are similar and consistent with each other.

- Regarding **Italy's** performance, the dendrogram points out that it is close to Bangladesh, Madagascar, Mexico and Chile. These countries are in the same cluster, which is Cluster 1 (Middle performing one), for both methods.

PART II:
BAYESIAN APPROACH

In the second part of this project, we adopted a new model for our analysis, which follows a Non parametric Bayesian approach.

In order to better understand the algorithm we used, we are going to give a brief introduction about the following **topics**:

1. BAYESIAN SETTING;
2. MODEL BASED CLUSTERING;
3. MARKOV CHAIN MONTE CARLO (MCMC);
4. BAYESIAN NONPARAMETRICS;
5. DIRICHLET PROCESS, STICK-BREAKING PROCESS AND CRP;
6. DIRICHLET PROCESS MIXTURE MODEL.

1. BAYESIAN SETTING

Adopting a the Bayesian approach, we set a prior distribution that describes our beliefs about the phenomenon we are studying and we use observe data to update them and obtain a posterior distribution, which can be used to do inference.

More formally, if we are interested about a certain parameter θ (e.g. the mean of the population), we can fix a **prior distribution** over it and update our beliefs after observing the data. Therefore, using the Bayes' theorem, we can obtain the **posterior distribution** of the parameter of interest as follows:

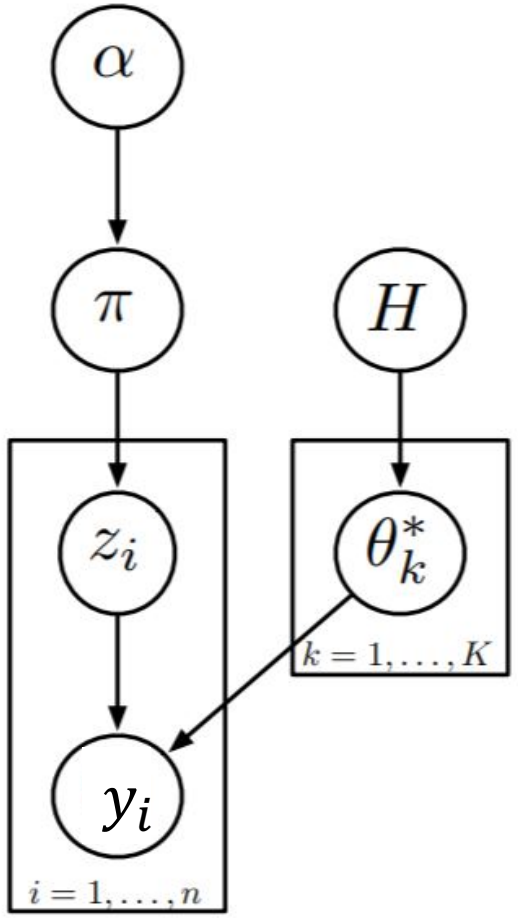
$$p(\theta|y_1, ..., y_n) = \frac{\prod_{i=1}^{n} p(y_i|\theta)\ \ p(\theta)}{\int_\Theta \prod_{i=1}^{n} p(y_i|\theta)\ \ p(\theta)\, d\theta}$$

where the numerator is the product between the likelihood function of the model and the prior distribution and the denominator is the marginal likelihood.

Sampling model	$Y_1, \dots Y_n \mid \theta \sim^{i.i.d.} p(y_i\|\theta)$
prior	$p(\theta)$

2. FINITE MIXTURE MODEL for clustering

A finite mixture model is used to describe the presence of subpopulations within a population. It assumes that there exist a mixture of K components (clusters), each belonging to the same distribution family and characterized by different parameters. Each observation $\boldsymbol{y_i}$ is associated to a latent variable $\boldsymbol{z_i}$ (distributed as a Categorical distribution) that describes its identity, that is which cluster it belongs to, with some probabilities $\boldsymbol{\pi_k} = \boldsymbol{P(K)}$ that sum up to one (*mixture weights*) .



For instance, each cluster k can be modelled as a Normal with mean $\boldsymbol{\theta_k}$. Also, each observation $\boldsymbol{y_i}$ is generated by first assigning it to a cluster k, according to a probability $\boldsymbol{\pi_k}$ and then generating the observation from its corresponding distribution parametrized by $\boldsymbol{\theta_k}$, that is $\boldsymbol{F(y_i|\theta_k)}$.

We are interested in making inference about a latent (unobserved) variable $\boldsymbol{z_i}$, which allows us to identify the clusters of the population.

If we are being Bayesian, we can set **priors** for the mixture weights and for the parameters of the clusters. The choice of these priors can be done following different criteria (e.g. conjugacy).

For instance, assuming the cluster are distributed as a normal, a good choice for the mean $\boldsymbol{\theta_k}$ could be a gaussian distribution.

A common choice for the mixture weights $\boldsymbol{\pi_k}$(probabilities summing up to one) is the Dirichlet distribution, which is a generalization of the Beta distribution and it's a conjugate prior both for the categorical and multinomial distribution. In particular, a Dirichlet distribution with parameter $\alpha = 1$ can be interpreted as a uniform distribution over its support. Its density function is given by

$$f(x_i; \alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} \prod_{i=1}^{K} x^{\alpha_i - 1}$$

where $x_i \in (0,1)$ and $\sum x_i = 1$.

As we have said before, given a data set, we are interested in the cluster assignments given by the latent variable of the model. Using the Bayesian setting, we can obtain the posterior distribution of interest, that is

$$p(z|y) = \frac{p(y|z)\, p(z)}{\sum_K p(y|z)\, p(z)}$$

where the likelihood is obtained marginalizing as follows:

$$p(y|z) = \int_\Theta [\prod_{n=1}^{N} F(y|\theta_k) \prod_{k=1}^{K} G_0(\theta_k)] d\theta \qquad \text{and} \qquad \begin{aligned} \theta_k &\sim G_0 \\ z &\sim p(z) \end{aligned}$$

In particular, if G_0 is conjugate, the integral can be easily computed analytically. On the contrary, the denominator of the posterior is a sum over every possible partition (grouping) of the data into K groups; this computation can be particularly difficult to obtain, especially as K grows. For this reason, in order to obtain the posterior distribution of interest we can use approximation techniques such as MCMC.

3. MCMC INTEGRATION

- **Monte Carlo Simulation:**

Monte Carlo Simulation is a technique that generates random variables for modelling risk or uncertainty of a certain system.

It is grounded on Law of Large Numbers (LLN) and the Central Limit Theorem (CLT);indeed, MC integration samples an **i.i.d** sequence from the density **p(θ)**, so that the two mentioned results hold true. But, Monte Carlo Simulation can be applied also in case of dependent random sequences, like **Markov Chains**;

- **Markov Chains:**

A Markov chain $\{X_{(t)}\}$ is a sequence of dependent random variables $X_{(0)},X_{(1)},X_{(2)},\ldots,X_{(t)},\ldots$ where the probability distribution of $X_{(t)}$ depends only on $X_{(t-1)}$.

The conditional distribution of $X_{(t)} \mid X_{(t-1)}$ is a **transition kernel** K, $(X_{(t+1)} \mid X_{(0)},X_{(1)},X_{(2)},\ldots,X_{(t)}) \sim K(X_{(t)},X_{(t+1)})$.

Markov chain Monte Carlo (MCMC) Markov chains typically have very strong **stability** properties: **irreducibility**; **recurrency** and have a **stationary probability distribution.**

MCMC

- **Markov Property:** Given a process which is at a state **Xn** at a particular point of time, the probability of **Xn+1=k**, where k is any of the M possible states the process can jump to, will **only be dependent** on which state it is **at the given moment;** and not on how it reached the current state.

- The stationary state distribution allows you to define the probability for every state of a system at a random time.

- Markov Chain Monte Carlo (MCMC) methods are a **class of algorithms** for sampling from a probability distribution based on constructing a **Markov chain** that has the desired distribution as its **stationary distribution.** The state of the chain after a number of steps is then used as a sample of the desired distribution. The quality of the sample improves as a function of the number of steps.

Metropolis-Hastings (MH) Algorithm

- The Metropolis-Hastings (MH) algorithm resembles an ***acceptance-rejection sampling algorithm***, while acceptance rejection is used to draw an independent and identically distributed sample from a target density, the MH algorithm yields (under some condition) from an Harris-ergodic Markov chain.

- The MH algorithm is based on **proposing values** sampled from an instrumental **distribution proposal *r*(δ|θ)**, which are then accepted with a certain probability. This probability reflects how likely it is that these values come from the **target density *p*(θ)**, which is the distribution we're interested in studying.

- This algorithm is useful to use when we are not able to retrieve neither the posterior nor the full conditionals; because to set it up we only need to know the target density up to a normalizing constant. In other words, we use a density that is proportional to our target one.

- Choosing the proposal density r(δ|θ) can be quite **challenging**, since it can be any distribution that has an adequate support.

MH ALGORITHM SCHEME

GOAL: Build a Markov Chain $\boldsymbol{\theta_1, \ldots, \theta_k}$ with stationary density $\boldsymbol{p(\theta)}$.

Let $\boldsymbol{r(\delta \mid \theta)}$ be a the proposal density on Θ for each θ;

$\boldsymbol{\delta}$: proposed value/state of θ ; $\boldsymbol{\theta_{g-1}}$: current value/state of θ;

Starting with $\theta_0 \in R^k$ build a Markov chain iterating for $g = 1, 2, \ldots$ as follows:

1. Draw $\delta \sim r(\delta \mid \theta_{g-1})$;

2. Compute the acceptance ratio:

$$\alpha(\delta|\theta_{g-1}) = min\left\{1, \frac{p(\delta)r(\theta_{g-1}|\delta)}{p(\theta_{g-1})r(\delta|\theta_{g-1})}\right\}$$

3. Draw a value from an uniform distribution $U \sim$Unif(0, 1)

4. If $u < a$, then set $\theta_g = \delta$ (accept a move), otherwise set $\theta_g = \theta_{g-1}$ (reject the move).

The Gibbs Sampling

This algorithm is designed to draw a **Markov Chain** on multidimensional state space, $\theta^{(g)}= (\theta_1^{(g)}, \ldots, \theta_k^{(g)})$, where g denotes the g^{th} state of Gibbs-generated Markov Chain.

The **main ingredients** to set up a Gibbs sampling are the so called **full conditonals**, for $i = 1, \cdots, k$

$$p(\theta_i \mid \theta_{-i}) = p(\theta_i \mid \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_k)$$

Each iteration of the basic Gibbs sampling can be interpreted as a cycle of k Metropolis-Hastings steps with acceptance ratio equal to one.

How to get full conditionals:

Suppose we have a **posterior** $p(\theta|y_{1:n})$.

To compute the full conditionals for each component of $\theta= (\theta_1, \ldots, \theta_k)$, folllow these steps:

1. Write out the full posterior density **ignoring constants of proportionality;**

2. Pick a block of parameters (for example, θ_i) and **drop** everything that doesn't depend on θ_i ;

3. By looking at the kernel is possible to figure out what the normalizing constant is and, therefore the full conditional density i $p(\theta_i \mid \theta_{-i}, y_{1:n})$;

The Gibbs Sampling

The Gibbs Sampling in a *k* dimensional space is a **sequential** (in *k*-steps) **algorithm** given by the following transition from , $\theta^{(g-1)}$ to $\theta_1^{(g)}$, for g = 1, 2, …:

Starting with $\theta^{(0)} = (\theta_1^{(0)}, \ldots, \theta_k^{(0)})$

K-STEPS of an iteration g:

1. Draw $\boldsymbol{\theta_1^{(g)}} \sim \mathrm{p}((\theta_1 | \theta_2^{(g-1)}, \ldots, \theta_k^{(g-1)}) = p(\theta_1 | \theta_{-1}^{(g-1)})$
2. Draw $\boldsymbol{\theta_2^{(g)}} \sim \mathrm{p}((\theta_2 | \boldsymbol{\theta_1^{(g)}}, \ldots, \theta_k^{(g-1)}) = p(\theta_2 | \theta_{-2}^{(g-1)})$
3. Draw $\boldsymbol{\theta_3^{(g)}} \sim \mathrm{p}((\theta_3 | \boldsymbol{\theta_1^{(g)}}, \boldsymbol{\theta_2^{(g)}}, \ldots, \theta_k^{(g-1)}) = p(\theta_3 | \theta_{-3}^{(g-1)})$

…

k. Draw $\theta_k^{(g)} \sim \mathrm{p}((\theta_k | \boldsymbol{\theta_1^{(g)}}, \boldsymbol{\theta_2^{(g)}}, \boldsymbol{\theta_3^{(g)}}, \ldots, \theta_{k-1}^{(g)}) = p(\theta_k | \theta_{-k}^{(g-1)})$

At each iteration g the Gibbs sampling produces a simulation for the paramteres of the posterior.

Checking Convergence

Two main issues arise when we are dealing with MCMC outputs:

1. **Convergence** of the Markov Chain to the stationary distribution
2. **Dependence(autocorrelation)** between elements of the Markov chain

In order to make inference a proper analysis of convergence of the MCMC should be performed;

Visual inspection is essential to check the convergence of the algorithm**:**

Trace plot: This is a plot of (1,2,...,*G*) versus the sample $\theta^{(1)}$, $\theta^{(2)}$, …, to $\theta^{(G)}$

Usually, $\theta^{(0)}$ is far away from the major support of the posterior stationary density. Initially then, the chain will often be seen to "migrate" away from $\theta^{(0)}$ towards a region of high posterior probability centred around a mode of ***p*(θ).**

Ergodic mean plot: the ergodic mean refers to the mean value until the current iteration (cumulative mean); if the ergodic mean stabilizes after some iterations, then this is and indication of the convergence the algorithm.

Checking Convergence

BURN IN (convergence issue)

The early samples are strongly influenced by the distribution of $\theta^{(0)}$ (starting values),which presumably is not drawn from $p(\theta)$;

The initial set of *T* samples, $\left\{\theta^{(0)}, \ldots, \theta^{(T-1)}\right\}$ can be **unrepresentative** of the steady state of the chain, so it's useful to discard it.

The time (iteration number) ***T*** is known as the **burn-in .**

Knowing when to start collecting samples is a non-trivial task; usually, in a low dimensional MCMC algorithm, it is suggested to consider *T* = 1000.

THINNING (dependence issue)

If the **autocorrelation** in a chain is high, thin the chain by only saving one in every *m* draws.

For the Metropolis-Hasting algorithm, the autocorrelation can be drastically reduced tuning the variance of the proposal distibution.

4. BAYESIAN NONPARAMETRICS

Despite the name, Nonparametrics Bayesian models are not models without parameters; they are really **large parametric models**, where the number of parameters grows as the data grow. Nonparametric models still have parameters; they just have an infinite number of them. The intuition is that *we learn more as we observe more*.

This approach can offer major advantages in many fields. For instance, when classifying species, such a flexible model can be very useful in a setting in which we discover new species every day and we are not planning to stop these discoveries anytime soon.

Another major reason for which this kind of approach can be chosen is due to the fact that it eliminates problems linked to **model selection**.

In the specific case of clustering, for example, we have seen different algorithms for which it was necessary to specify the number of clusters in advance. This can lead to phenomena as **overfitting** or **underfitting** when the choice of the parameter K is not right (e.g. we obtain more or less clusters than needed). Although well specified parametric Bayesian models typically do not present overfitting effects, using a non parametric approach underfitting can be avoided as well.

A first intuition of this new approach can be achieved interpreting the nonparametric clustering model as a **limit of the finite mixture model** we described before.

Using a Gibbs sampler, we can obtain the following posterior distribution for the latent variable:

$$p(z_i = k|rest) \propto \frac{\frac{\alpha}{K} + n_k^{-1}}{\alpha + n - 1} \cdot f(y_i|\{y_j : j \neq i, z_j = k\})$$

that is, proportional to the conditional probability of y_i given all the other observations already assigned to cluster k.

Now assume a very large value for the integer K (that is, $K \to \infty$); since the sample size n is now smaller than K, there are at most $n < K$ occupied clusters and most components are empty (that is, have no data items in them). Then, we can group the occupied and empty clusters and write

$$p(z_i = k_{OCC}|rest) \propto \frac{\frac{\alpha}{K} + n_k^{-1}}{\alpha + n - 1} \cdot f(y_i|\{y_j : j \neq i, z_j = k\})$$

$$p(z_i = k_{EMPTY}|rest) \propto \frac{\alpha + \frac{K - K_{OCC}}{K}}{\alpha + n - 1} \cdot f(y_i|\{y_j : j \neq i, z_j = k\})$$

Then, taking the limit for $K \to \infty$, since ${}^{\alpha}/_{k} \to 0$ and ${}^{(K-K_{OCC})}/_{K} \to 1$, we obtain the following

$$p(z_i = k_{OCC}|rest) \propto \frac{n_k^{-1}}{\alpha + n - 1} \cdot f(y_i|\{y_j : j \neq i, z_j = k\})$$

$$p(z_i = k_{EMPTY}|rest) \propto \frac{\alpha}{\alpha + n - 1} \cdot f(y_i|\{y_j : j \neq i, z_j = k\})$$

that is, the probability of the observation y_i belonging to cluster k is proportional to the number of data items already in that cluster (n_k^{-1}) and the probability of the same observation belonging to a (new) empty cluster is proportional to a positive number α.

This limit case of the Gibbs sampler procedure is the one used in Dirichlet Process mixture models, that are nonparametric Bayesian clustering models.

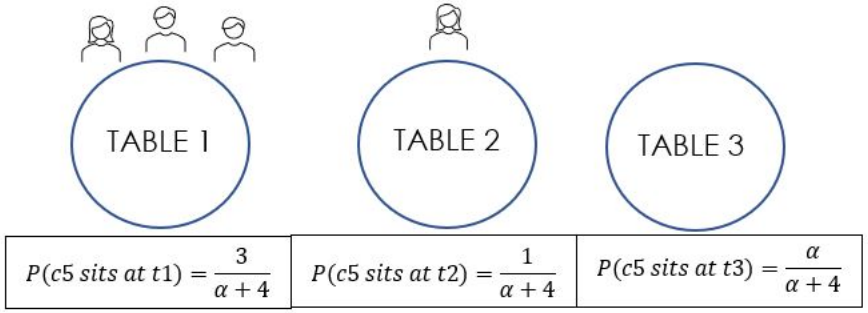
It is clear why this limit case of the Gibbs sampler result goes in the direction of Bayesian nonparametrics. Indeed, when modelling the probability of the latent variable, we are avoiding model selection, since we can always add a new empty cluster to the model.
However, it should be noted that, while under and over-fitting effects are easily avoided, a miss-fitting is still possible, since the probability of a data point being assigned to a new cluster depends on α, which is a prior hyperparameter. It is easy to see how larger values of α are associated with an increase of the number of clusters in the model. To solve this problem, one can fix an hyper-prior on the hyperparameter α, so that Bayesian inference can be done on this parameter too.

Bayesian Nonparametric clustering addresses the problem of choosing the number of clusters in advance by assuming that there is an **infinite number of latent clusters**, but that a finite number of them is actually used to generate the observe data points. Also, a prior over infinite groupings is specified in such a way that the model favours assigning data to a small number of groups.
Such a prior is typically a **Dirichlet process**; to better understand it, we introduce the so called Chinese Restaurant Process (CRP), which is a representation of the Dirichlet process itself.

5. CHINESE RESTAURANT PROCESS (CRP)

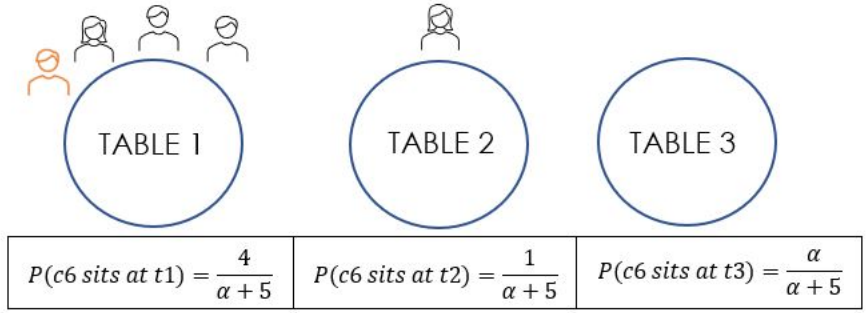
This process takes the name from the following analogy. Imagine a restaurant with an infinite number of tables, each of which with an infinite number of seats.

Now imagine a sequence of individual customers entering and sitting down.



The very first customer comes in and sits (alone)

at the first available table (say, table 1); the second customer can either choose to sit together with the first customer at table 1 with probability $\frac{1}{1+\alpha}$



or choose a new empty table with probability $\frac{\alpha}{1+\alpha}$, where $\alpha > 0$ and $\alpha \in \mathbb{R}$. The *nth* customer that comes in can either choose an already occupied table with probability proportional to the number of people already sitting at that table or choose a new table with probability proportional to α.

At any point of this process, the assignment of customers to tables defines a **random partition**, that is, a possible (random) clustering result.

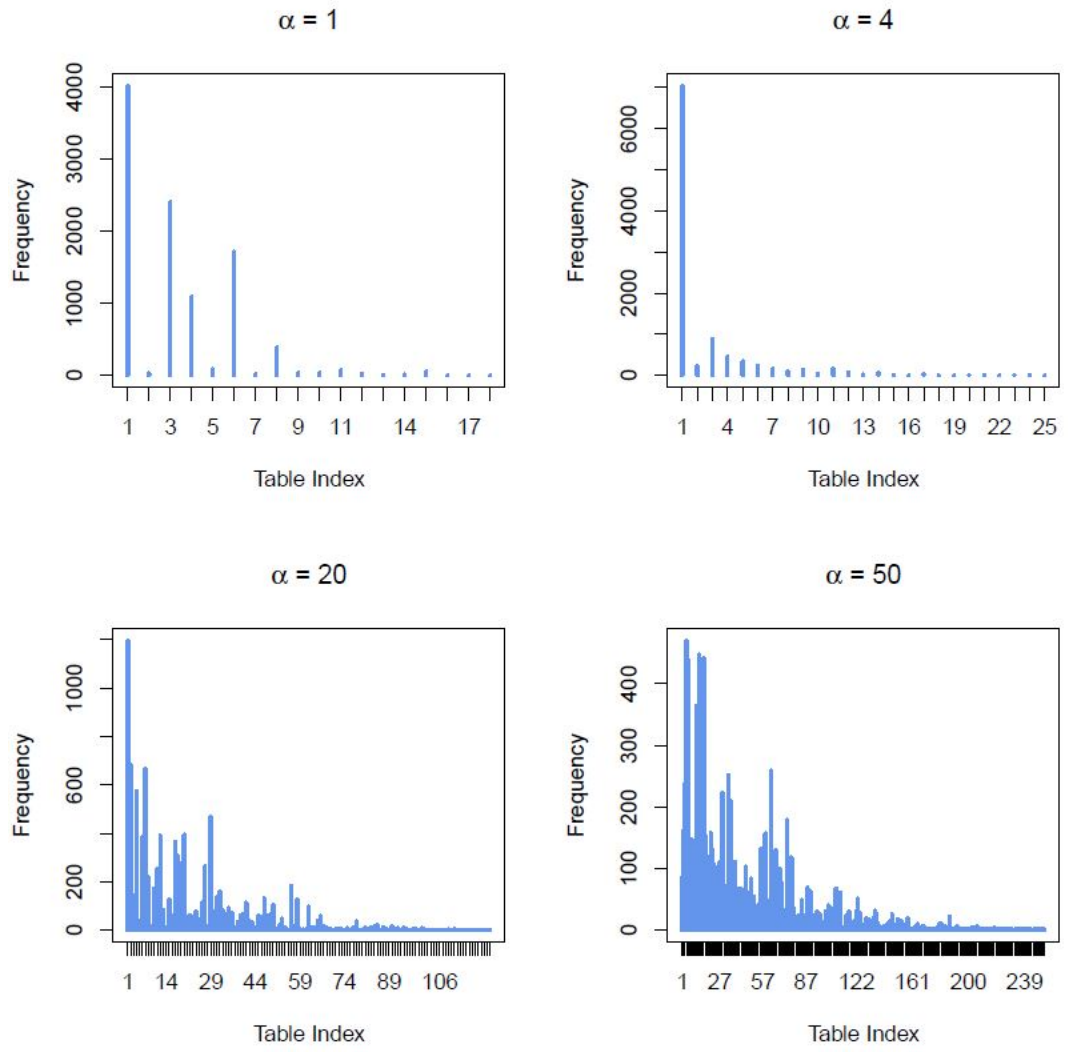
More formally, let C_n be the table (*cluster*) assignment for the *nth* customer (*data point*). A draw from this distribution can be generated by sequentially assigning observations to clusters with the following probability:

$$p(C_n = k | C_{1:n}) \propto \begin{cases} \frac{n_k^{-1}}{\alpha+n-1} & \text{if } k \text{ is a previously occupied table} \\ \frac{\alpha}{\alpha+n-1} & \text{if } k \text{ is a new empty table} \end{cases}$$

where n_k^{-1} is the number of customers already sitting at the table; recall that this is exactly the result we obtained taking the infinite limit of a finite mixture model.
The parameter α is called **concentration parameter**.

The CRP exhibits an important property: the cluster assignment are **exchangeable**. This means that $p(C)$ does not change if the order of customers entering the restaurant is shuffled; in other words, the probability of a particular seating configuration $C_{1:n}$ depends only on the number of groups k and the size of each group n_k and not on the order in which the customers came in.

As we have seen for the limit case of the Gibbs sampler, a larger value of the concentration parameter α will provide more (new) tables and less customers per table.



DIRICHLET PROCESS

A Dirichlet process is a distribution over distributions parametrized by a concentration parameter $\alpha > 0$ and a base distribution G_0 over the space Θ; that is, a draw from a DP is itself a (discrete) distribution in the same space. A random distribution G drawn from a Dirichlet Process is $G \sim DP(\alpha, G_0)$

Distributions drawn from a Dirichlet process are discrete but, cannot be defined using a finite number of parameters; for this reason, we can talk about a nonparametric model.

Assuming a measurable partition of Θ, $\{R_1, \ldots, R_K\}$; if $G \sim DP(\alpha, G_0)$, then every measurable partition of Θ is Dirichlet distributed, that is $(G(R_1), \ldots, G(R_K)) \sim Dir(\alpha G_0(R_1), \ldots, \alpha G_0(R_K))$ Then, if we draw a random distribution from the Dirichlet process and add up the probability mass in a region $R \, \epsilon \, \Theta$, there will be on average $G_0(R)$ mass in that region.

Random distributions drawn from a DP are discrete and place probability mass on a countable infinite collection of points (atoms), that is, $G = \sum_{i=1}^{\infty} \pi_k \, \delta_{\theta_k}$, where π_k is the probability assigned to the kth atom and θ_k is the location (value) of that atom; δ is an indicator function. Atoms are drawn independently from the base distribution G_0.

There is a property that connects the Dirichlet process with the CRP; consider a random draw from a DP and repeated draws from this random distribution:

$$G \sim DP(\alpha, G_0)$$

$$\theta_i \sim G, \quad i \sim \{1, ..., n\}$$

We can obtain the joint distribution of $\boldsymbol{\theta_{i:n}}$ marginalizing out the random distribution $\boldsymbol{G}$, that is

$$p(\theta_1, ..., \theta_n | \alpha, G_0) = \int (\prod_{i=1}^{n} p(\theta_i, G)) \; dP(G | \alpha, G_0)$$

One can show that under this joint distribution, $\boldsymbol{\theta_i}$ exhibits a **clustering property,** that is they share repeated values with positive probability. The structure of the shared values describes a partition of integers from 1 to n and the distribution of such partition is a CRP with parameter α.
This is another way to say that the DP is characterized by **exchangeability** of $\boldsymbol{\theta_i}$.

The Dirichlet process is also characterized by a **self-reinforcing property**, that is the more often a given value has been sampled in previous iterations, the more likely it is to be sampled again.
Even if G is a distribution over an uncountable set, there exists a non null probability of two samples having the same value, since the probability mass concentrates on a small number of clusters.
For these reasons, the Dirichlet Process is one of the most used priors in nonparametrics Bayesian statistics and it is a very good solution for clustering models.

STICK-BREAKING PROCESS

We have defined draws from a DP as composed of a weighted sum of point masses

$$G = \sum_{i=1}^{\infty} \pi_k \, \delta_{\theta_k}$$

Draws from a Dirichlet process are distributions over a set S which is infinite in size. This can seem just a strong theoretical result, which cannot be exploited in practice; how can we sample from such a distribution?

The solution consists in truncating the set dimension in order to approximate a DP process. In particular, there exists a constructive definition of the DP as such, the *stick-breaking construction*.

Consider a stick of length 1; then, divide it into an infinite number of segments $\boldsymbol{\pi_k}$ as follows:

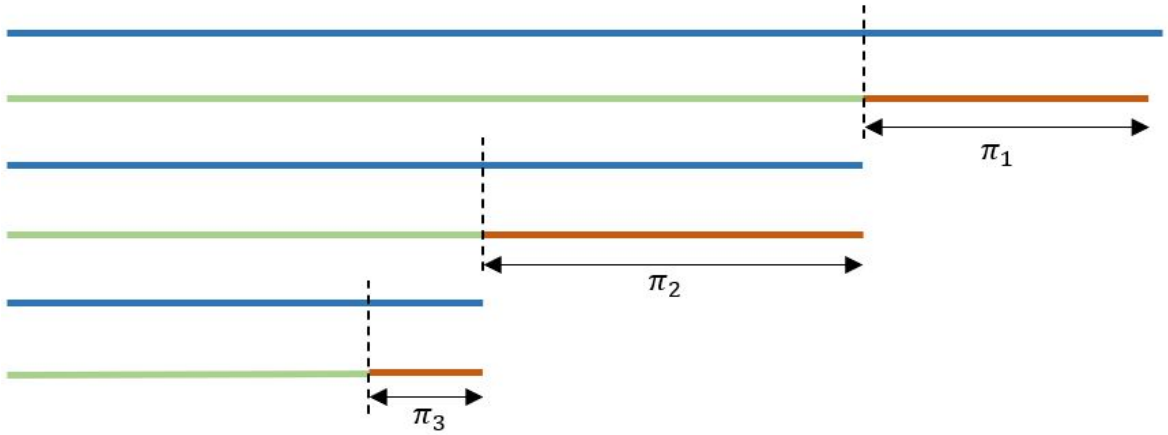
- draw from a Beta r.v. $\beta_1 \sim beta(1, \alpha)$ and break the stick of β_1
- for the remaining stick, draw another β_2 and break it of β_2
- ….

This allows to obtain an infinite sequence of weights π_k:

$$\beta_k \sim beta(1, \alpha)$$

$$\pi_k = \beta_k \prod_{j=1}^{k-1}(1 - \beta_j) \;\; k = 1, 2, 3....$$

The stick-breaking distribution over π can be written as $\pi \sim GEM(\alpha)$, named after Griffiths, Engen and McCloskey. The stick-breaking construction has led to a variety of extensions of the Dirichlet process



6. DIRICHLET PROCESS MIXTURE MODEL

Recall the Bayesian setting of a generic Finite Mixture Model with k clusters:

$$\pi = (\pi_1, ..., \pi_k)|\alpha \sim Dirichlet(\tfrac{\alpha}{k}, ..., \tfrac{\alpha}{k}) \qquad z \sim Multinomial(\pi)$$

$$\theta_k|G_0 \sim G_0 \qquad y_i|z_i, \theta_k \sim F(\theta_k)$$

In this setting, we define priors over the mixture weights and the parameters of the clusters. We also need to specify the number of clusters in advance.

We can transform this model in order to exploit the properties of nonparametric models as follows:

$$\pi|\alpha \sim GEM(\alpha) \qquad z \sim Multinomial(\pi)$$

$$\theta_i|G \sim G \qquad y_i|z_i, \theta_i \sim F(\theta_i)$$

$$G \sim DP(\alpha, G_0)$$

Note that the change in the subscript (from *k* to *i*) for the parameter θ; shifting to a nonparametric setting, each observation is associated to a parameter θ_i and since G is discrete, multiple θ_i's can take on the same value θ_k^* (*self-reinforcing property*) and the above model can be interpreted as a mixture model, where y_i's with the same value of θ_i are in the same cluster. In particular, the components are the latent groups, while the clusters are the components actually represented in the data. The number of clusters is random and grows as the data grow.

Also, $G = \sum_{i=1}^{\infty} \pi_k \, \delta_{\theta_k}$; we can sample from this using the stick – breaking process. Regarding the latent variable z_i the Multinomial distribution it's just a generalization of the Categorical.

The DP mixture model is an **infinite mixture model** with a (countably) infinite number of clusters. However, since the weights decrease exponentially quickly, a small number of components are used to model the data a priori (the expected value of components used a priori increases in a logarithmic trend wrt the number of observations).
In the DP mixture model, the number of clusters k is not fixed and can be inferred from the data using a Bayesian posterior inference framework.

Let $\theta_1, \dots, \theta_n$ a sequence of independent draws from G. We are interested in the posterior distribution of G given observed values of $\theta_1, \dots, \theta_n$. Let $R_1, \dots, R_s$ be a finite measurable partition of Θ, and let n_k be the number of observed values in R_k. By the conjugacy between the Dirichlet and the multinomial distributions, we obtain:

$$(G(R_1), \dots, G(R_s))|\theta_1, \dots \theta_n \sim Dir(\alpha G_0(R_1) + n_1, \dots, \alpha G_0(R_s) + n_s)$$

This is the **posterior distribution of a Dirichlet Process**, which is a Dirichlet Process itself.

It can be easily shown that the updated concentration parameter for the posterior is $\alpha + n$ and that the base distribution is given by $\frac{\alpha G_0 + \sum_{i=1}^{n} \delta_{\theta_i}}{\alpha + n}$; this means that the **DP** provides a **conjugate family of priors** over distributions, closed under posterior updates given the observations.

Then, we can write $G|\theta_1, \dots, \theta_n \sim DP(\alpha + n, \frac{\alpha}{\alpha + n} G_0 + \frac{n}{\alpha + n} \frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n})$

The **posterior base distribution** can be interpreted as a **weighted average** between the **prior** base distribution G_0 and the **empirical** distribution $\frac{\sum_{i=1}^{n} \delta_{\theta_i}}{n}$. The weight of the prior base distribution is proportional to α, while the empirical distribution has weight proportional to the number of observations n. Then, we can say α is the **strength (mass) of the prior**; if it tends to zero, the prior becomes non-informative. As n grows, the posterior is dominated by the empirical distribution.

One can show that the **predictive distribution** for θ_{n+1}, conditioned on $\theta_1, \ldots, \theta_n$ and with G marginalized out coincides with the base distribution of the posterior of the DP, that is

$$\theta_{n+1}|\theta_1, \ldots, \theta_n \sim \frac{1}{\alpha + n}(\alpha G_0 + \sum_{i=1}^{n} \delta_{\theta_i})$$

which can be written as

$$\theta_{n+1}|\theta_1, \ldots, \theta_n \sim \frac{1}{\alpha + n}(\alpha G_0 + \sum_{k=1}^{m} n_k \delta_{\theta_k^*})$$

where $\theta_1^*, \ldots, \theta_m^*$ are the unique values among $\theta_1, \ldots, \theta_n$ and n_k is the number of repeats of θ_k^*. In particular, θ_i's with same values θ_k^* are in the same cluster. Also, θ_k^* will be repeated by θ_{n+1} with probability proportional to n_k (number of times it has already been observed). The larger n_k, the higher the probability that it will grow; this is the *rich-gets-richer phenomenon*, for which large clusters grow larger faster.

The value m represents the number of clusters.

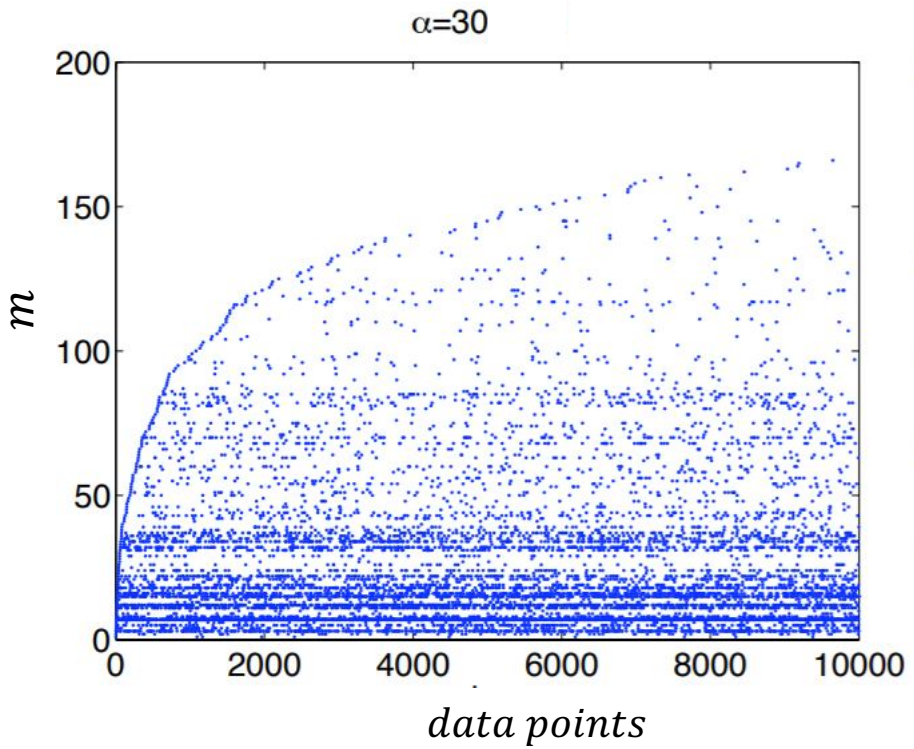
Moreover, for $i \geq 1$, the observation θ_i takes on a new value not observed before (incrementing m by one) with probability $\frac{\alpha}{\alpha + i - 1}$, independently of the number of clusters among previous values of θ.

Then, we can compute the mean of the number of clusters m as:

$$E[m|n] = \sum_{i=1}^{n} \frac{\alpha}{\alpha + i - 1} = \alpha(\psi(\alpha + n) - \psi(\alpha)) \approx \alpha log(1 + \frac{n}{\alpha})$$

where $\psi(\cdot)$ is the digamma function.

The **number of clusters** grows logarithmically in the number of observations; this makes sense because of the rich-gets-richer phenomenon (we expect there to be large clusters, so m needs to be smaller than n). Also, as noted before, α controls the number of clusters



As for the Finite Mixture Model case, we can compute the posterior distribution of interest exploiting MCMC methods.

NON PARAMETRIC BAYESIAN MODELS FOR TIME SERIES CLUSTERING

In order to extend the analysis of our data set using Bayesian non parametric procedures, we used the model described in the paper ***A Bayesian Nonparametric Approach for Time Series Clustering*** by Luis E. Nieto-Barajas and Alberto Contreras-Cristan.

We also used the suggested R package **BNPTSclust** in order to implement the model for our data set.

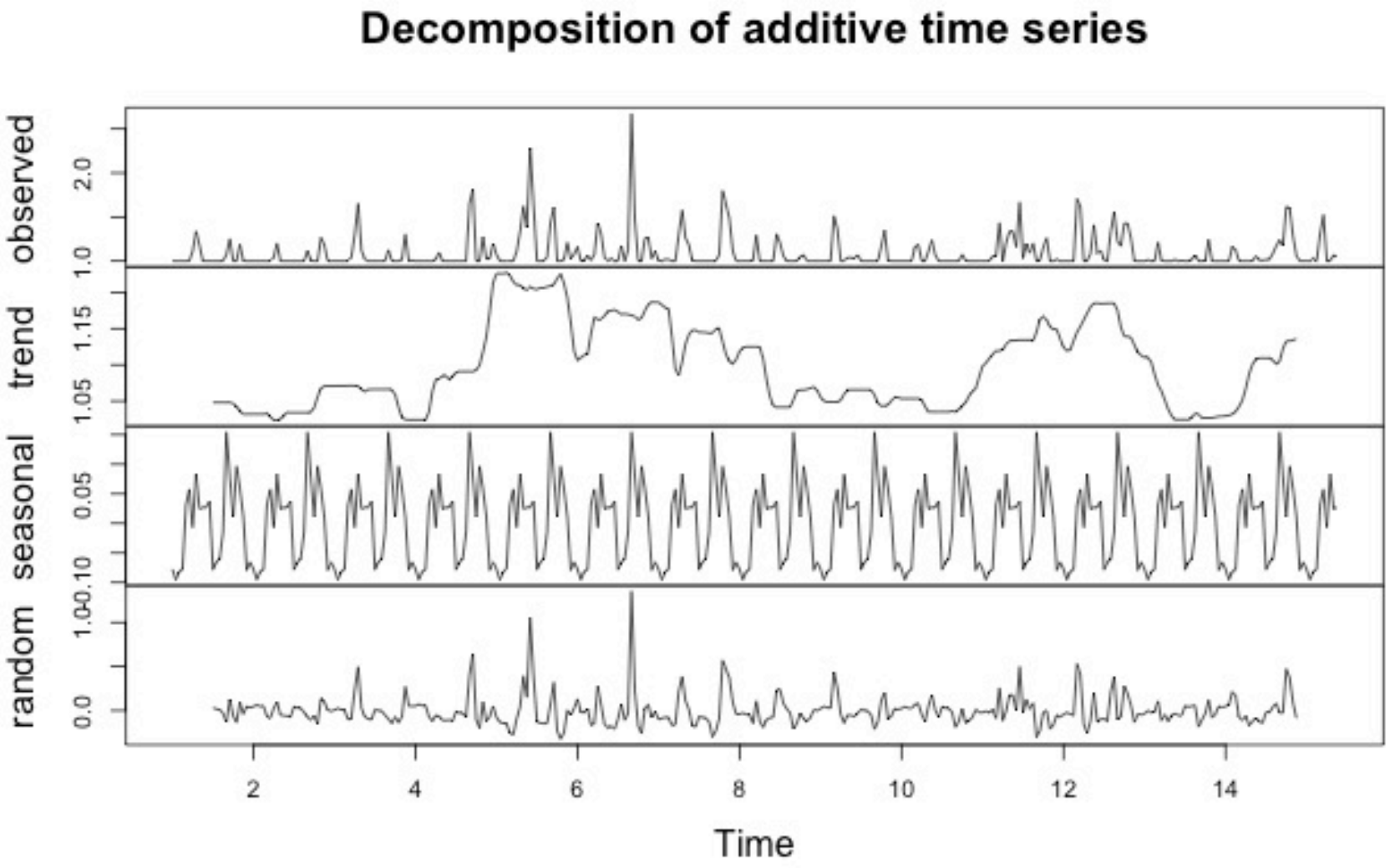
We will cover the following topics:

1. SAMPLING MODEL;
2. PRIOR DISTRIBUTIONS;
3. HYPER-PRIOR DISTRIBUTIONS;
4. POSTERIOR CHARACTERIZATION;
5. CLUSTERING, SELECTION AND FITTING MEASURES;
6. BNPTSclust PACKAGE;
7. OUR RESULTS;
8. INTERPRETATION.

1. SAMPLING MODEL

- A useful abstraction to analyse time series is to **decompose** them into at least one systematic **pattern** that describes them;
- The most **common patterns/components** are:
  - **Systematic**: components that have consistency or recurrence and can be described and modeled;
    - **Level:** the average value in the series.
    - **Trend**: the increasing or decreasing value in the series.
    - **Seasonality**: the repeating short-term cycle in the series.
  - **Non-Systematic**: components that cannot be directly modeled.
    - **Noise**: the random variation in the series.
- Since we are dealing with **yearly data**, we should disregard seasonality and focus on **level** and **trends**;
- These components usually are specified in **additive** and **multiplicative** models, that have a **satisfactory performance** for time series analysis, in the vast majority of cases.

TIME SERIES DECOMPOSITION



TIME SERIES DECOMPOSITION

TREND:

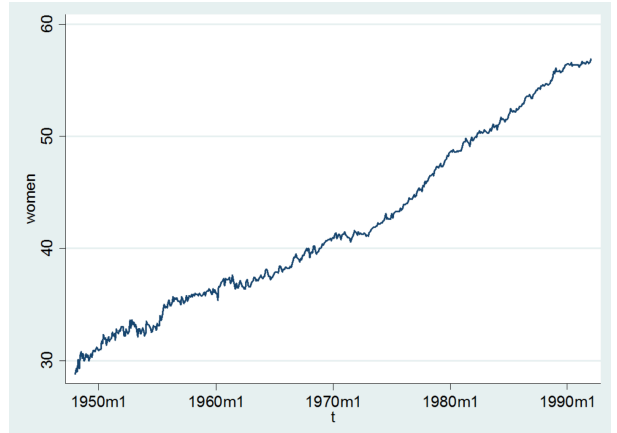
- The trend shows the general tendency of the data **to increase or decrease** during a **long period** of time. A trend is a **smooth, general, long-term, average** tendency.
- It is not always necessary that the increase or decrease is in the same direction throughout the given period of time. It is observable that the tendencies may increase, decrease or are stable in different sections of time. But the **overall trend** must be **upward, downward or stable**.
- Two different approaches could be used for **estimating** the trend;
  - **One approach** is to estimate the trend with a smoothing procedure such as **moving averages**. With this approach, an equation is not used to describe trend.
  - The **second approach** is to model the trend with a **regression equation**.

TIME SERIES DECOMPOSITION

TREND MODEL:

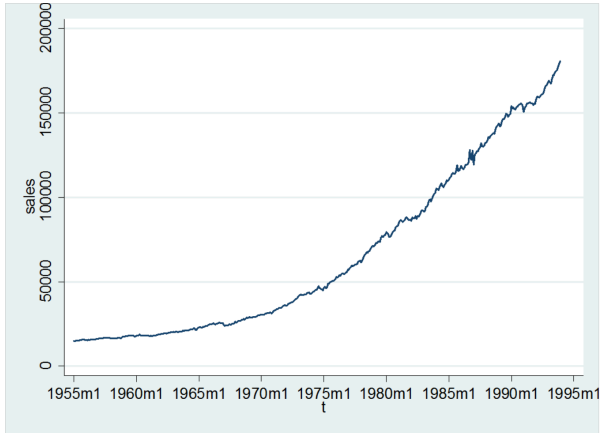
- There are several models to specify the trend of data: linear , quadratic, exponential;
- How do we choose between them? A way is to do **visual inspection** of your variables and of **residuals**.
- **Linear Trend Model**:

$$T_{t=} \beta_0+\beta_1 Time_t$$



- **Quadratic Trend Model**:

$$T_{t=} \beta_0+\beta_1 Time_t+ \beta_2 Time_t^2$$



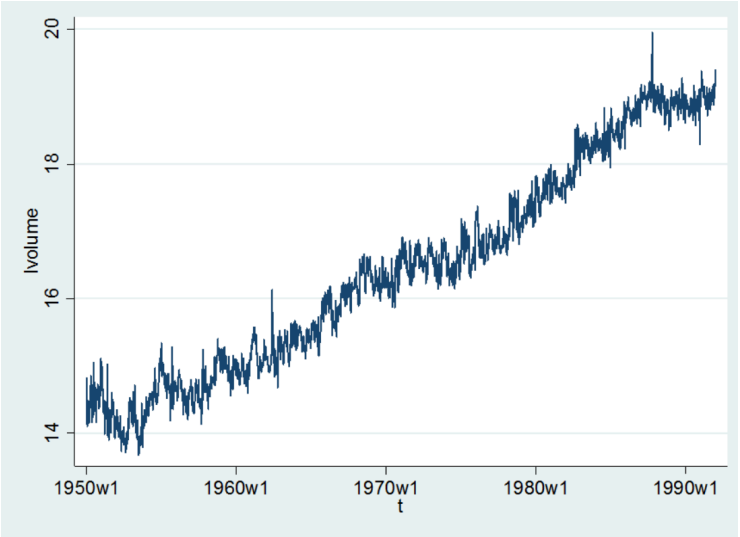
TIME SERIES DECOMPOSITION

- **Exponential Trend Model**:

$$T_t = e^{\beta 0 + \beta 1 Timet}$$

  - Becomes linear after taking natural logarithms

$$\ln(T_t) = \beta_0 + \beta_1 Time_t$$



  - Most economic series which are growing (aggregate output, such as GDP, investment, consumption) are exponentially increasing;
  - These series cannot be fit by a linear trend, but it is possible to **fit a linear trend** to their **(natural) logarithm**;

- The trend in the package we used can be specified as **polynomial**, by setting **deg** (degree) equal to the integer that represents the polynomial order of your trend model;

DYNAMIC LINEAR MODEL

The **BNPTSclust** applies, one of the most powerful **Bayesian models** for the analysis of time series , the the **dynamic linear model**, which is specified by two main equations**:**

Observation Equation

$$y_{it} = E(y_{it}) + \varepsilon_{it}$$

Where $\mathbf{y_{it}}$ is the value of the time series i at time t; **ε**=error term;

To accomodate the time series **components** into our observation equation, the **BNPTSclust** algorithm uses an **additive model** we write:

$$E(y_{it}) = \mu_i + \omega_i' \, g(t) + \theta_{it}$$

Where **μ:** level; **ω:** polynomial trend, **g(t):** trend model, **θ:** temporal component: plays the role of a dynamic intercept that accounts for time dependence in the observations;

Evolution equation

$$\theta_{it} = \rho\theta_{i,t-1} + v_{it}$$

Describes a dynamic behavior in the coefficients θ_{it} as an **autoregressive process** of order one AR(1)

DYNAMIC LINEAR MODEL

Depending on the data characteristics, not all of the parameters/components will be useful for clustering purposes. For instance two series that share the same trend, seasonalities and temporal components but differ in the level might be desired to belong to the same cluster.

Thus, we will write our general **sampling model** as

$$\mathbf{y_i = Z\alpha_i + X\beta_i + \theta_i + \varepsilon_i,}\ \ i=1,2,...n$$

- **Z** and **X** are **two design matrices** of dimension T × p and T × d respectively.
- $\boldsymbol{\alpha_i}$ (p×1 dimensional vector) includes all the parameter **not** used for clustering;
- $\boldsymbol{\beta_i}$ (d×1 dimensional vector) includes the parameters/component in which we base our clustering;

Only $\boldsymbol{\beta_i}$ and $\boldsymbol{\theta_i}$ will be considered for clustering.

For example, if the clustering is to be based on everything else rather than the level $\boldsymbol{\mu_i}$ then we would take $\boldsymbol{\alpha_i = \mu_i}$ and $\boldsymbol{\beta_i = (\omega_i, \upsilon_i)}$.

Finally, $\boldsymbol{\varepsilon' = (\varepsilon, ..., \varepsilon) \sim N(0, \sigma^2 I)}$ is the vector of measurement errors such that I is the identity matrix of dimension T × T .

2. PRIOR DISTRIBUTIONS

As stated before, according to the case, only certain components of the time series could be considered as criteria for clustering the observations. We can define $\boldsymbol{\gamma_i'} = (\boldsymbol{\beta_i'}, \boldsymbol{\theta_i'})$, which the vector of coefficients that will be used for clustering; the goal is to obtain a joint prior distribution $(\boldsymbol{\gamma_1}, \dots, \boldsymbol{\gamma_n})$ that allows for ties and respects the evolution equation described before; as we have seen when talking about non parametric clustering, a prior with these characteristics is the Dirichlet process.

In this particular case, we use a two parameter generalization of the DP, that is the **Poisson Dirichlet Process** (also known as Pitman – Yor process). In particular, a probability measure $\boldsymbol{G}$ with a **PD** prior with parameters $\boldsymbol{a} \in [\boldsymbol{0}, \boldsymbol{1}), \boldsymbol{b} \succ -\boldsymbol{a}$ and mean parameter $\boldsymbol{G_0}$ is denoted $\boldsymbol{G} \sim \boldsymbol{DP}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{G_0})$ and defined as follows:

$$G(\cdot) = \sum_{k=1}^{\infty} w_k \delta_{\xi_k}(\cdot)$$

$$\xi_k \sim^{i.i.d} G_0$$

$$w_k = v_k \prod_{l<k} (1 - v_l)$$

$$v_k \sim^{ind} Beta(1 - a, b + ka)$$

It is easy to see how this stochastic process and the DP are close; in particular, it is clear from the formulation in the last slide, that also the **PD** is generated by a stick-breaking process.

A **PD** with $a = 0$ is a **Dirichlet Process**. Another special case of the Poisson Dirichlet process when $b = 0$ is the so called **Normalized stable process**.

According to the Poisson-Dirichlet process, each γ_i is a copy from an element in the vector $\boldsymbol{\gamma_{-i}} = (\boldsymbol{\gamma_1}, \ldots, \boldsymbol{\gamma_{i-1}}, \boldsymbol{\gamma_{i+1}}, \ldots, \boldsymbol{\gamma_n})$ with certain probability and it is a drawing from the density g_0 associated to G_0 with probability:

$$f(\gamma_i|\gamma_{-i}) = \frac{b + am_i}{b + n - 1} g_o(\gamma_i) + \sum_{j=1}^{m_i} \frac{n^*_{j,i} - a}{b + n - 1} I_{\gamma^*_{j,i}}(\gamma_i), \quad i = 1, \ldots, n$$

where $(\boldsymbol{\gamma_{1,i}}*, \ldots, \boldsymbol{\gamma_{i,m}}*)$ are unique values in $\boldsymbol{\gamma_{-i}}$ occurring with frequency $\boldsymbol{n}*_{\boldsymbol{i,j}}$. Note that the above probability is a generalization of the CRP described in the previous slides.

Then, $\gamma_i|G \sim^{i.i.d.} G$, $G \sim PD(a, b, G_0)$.

If for two series $(i), (g)$, $\boldsymbol{\gamma_i = \gamma_g}$, then they belong to the same cluster. However, to define the final clusters, we need the posterior distribution of this parameter, which cannot be expressed analytically, but can be approximated by the implementation of a Gibbs Sampler.

Also, integrating G, $\boldsymbol{\beta_i}$ and $\boldsymbol{\theta_i}$ are independent with marginal distributions (respectively) $\boldsymbol{N_d(0, \Sigma_\beta)}$ and $\boldsymbol{N_T(0, R)}$, where $\boldsymbol{\Sigma_\beta = diag(\sigma^2_{\beta 1}, \dots, \sigma^2_{\beta d})}$ and $\boldsymbol{R = (R_{jk})}$ with typical element $\boldsymbol{R_{jk} = \sigma^2_\theta \rho^{|j-k|}}$.

Finally, for the parameter $\boldsymbol{\alpha_i}$, which contains all the coefficient not used for clustering, we set a normal prior with mean zero and variance matrix $\boldsymbol{\Sigma_\alpha = diag(\sigma^2_{\alpha 1}, \dots, \sigma^2_{\alpha p})}$.

3. HYPER - PRIOR DISTRIBUTIONS

In order to obtain the posterior distribution of interest, we must compute the posterior of all the hyperparameters too (approximating them with a Gibbs Sampler); to do so, we need to fix prior distributions over them (i.e. *hyper- priors*):

$$\sigma^2_{\epsilon_i} \sim IGa(c^{\epsilon}_0, c^{\epsilon}_1), i = 1, ..., n$$

$$\sigma^2_{\beta_j} \sim IGa(c^{\beta}_0, c^{\beta}_1), j = 1, ..., d$$

$$\sigma^2_{\alpha_k} \sim IGa(c^{\alpha}_0, c^{\alpha}_1), k = 1, ..., p$$

$$f(\sigma^2_\theta, \rho) \propto (\sigma^2_\theta)^{-1} \frac{\sqrt{1+\rho^2}}{1-\rho^2}$$

$$f(a) = \pi I_0(a) + (1-\pi) Be(a|q^a_0, q^a_1)$$

$$f(b|a) = Ga(b + a|q^b_0, q^b_1)$$

4. POSTERIOR CHARACTERIZATION

The posterior distribution of interest can be obtained through posterior draws from the marginal distribution of the parameter vector $\boldsymbol{\alpha}$ and the Gibbs sampler over the full conditional of the other hyperparameters, with the exception of $\boldsymbol{a}, \boldsymbol{b}$ and $\boldsymbol{\rho}$, for which the implementation of a Metropolis algorithm is required.

The posterior distribution for $\boldsymbol{\gamma}$ is characterized by a generalized Polya urn, which gives as full conditional the following distribution:

$$f(\boldsymbol{\gamma}_i \,|\, \mathbf{y}, \boldsymbol{\gamma}_{-i}, \boldsymbol{\sigma}_\epsilon, \boldsymbol{\Sigma}_\beta, \mathbf{R}) = q_0 \, g_0(\boldsymbol{\gamma}_i \,|\, \mathbf{y}_i, \sigma^2_{\epsilon_i}, \boldsymbol{\Sigma}_\beta, \mathbf{R}) + \sum_{j=1}^{m_i} q_j \delta_{\boldsymbol{\gamma}^*_{j,i}}(\boldsymbol{\gamma}_i)$$

$$g_0(\boldsymbol{\gamma}_i \,|\, \mathbf{y}_i, \sigma^2_{\epsilon_i}, \boldsymbol{\Sigma}_\beta, \mathbf{R}) = \mathrm{N}_T(\boldsymbol{\theta}_i \,|\, \boldsymbol{\mu}_{\theta_i}, \mathbf{S}_{\theta_i}) \times \mathrm{N}_d(\boldsymbol{\beta}_i \,|\, \boldsymbol{\mu}_{\beta_i}, \mathbf{V}_{\beta_i})$$

$$q_0 = \frac{D_0}{D_0 + \sum_{j=1}^{m_i} D_j}, \quad q_j = \frac{D_j}{D_0 + \sum_{j=1}^{m_i} D_j}$$

For completeness, we also report the other distributions used within the algorithm:

$$f(\sigma^2_{\epsilon_i} \,|\, \mathbf{y}, \text{rest}) = \text{IGa}\left(\sigma^2_{\epsilon_i} \,\middle|\, c_0^{\epsilon} + \frac{T}{2}\,,\, c_1^{\epsilon} + \frac{1}{2}\mathbf{M}_i'\mathbf{M}_i\right)$$

$$f(\sigma^2_{\theta} \,|\, \mathbf{y}, \text{rest}) = \text{IGa}\left(\sigma^2_{\theta} \,\middle|\, \frac{mT}{2}\,,\, \frac{1}{2}\sum_{j=1}^{m}(\boldsymbol{\theta}_j^*)'\mathbf{P}^{-1}\boldsymbol{\theta}_j^*\right)$$

$$f(\sigma^2_{\beta_k} \,|\, \mathbf{y}, \text{rest}) = \text{IGa}\left(\sigma^2_{\beta_k} \,\middle|\, c_0^{\beta} + \frac{m}{2}\,,\, c_1^{\beta} + \frac{1}{2}\sum_{j=1}^{m}(\beta_{jk}^*)^2\right)$$

$$f(\sigma^2_{\alpha_j} \,|\, \mathbf{y}, \text{rest}) = \text{IGa}\left(\sigma^2_{\alpha_j} \,\middle|\, c_0^{\alpha} + \frac{n}{2},\;\; c_1^{\alpha} + \frac{1}{2}\sum_{i=1}^{n}\alpha_{ij}^2\right)$$

$$f(a|b, \text{rest}) = \left\{\prod_{j=1}^{m-1}(b+ja)\right\}\left\{\prod_{j=1}^{m}\frac{\Gamma(n_j^*-a)}{\Gamma(1-a)}\right\} f(a)$$

$$f(b|a, \text{rest}) = \frac{\Gamma(b+1)}{\Gamma(b+n)}\left\{\prod_{j=1}^{m-1}(b+ja)\right\} f(b|a)$$

$$f(\rho \,|\, \mathbf{y}, \text{rest}) \propto |\mathbf{P}|^{-m/2}\exp\left\{-\frac{1}{2\sigma_\theta^2}\sum_{j=1}^{m}(\boldsymbol{\theta}_j^*)'\mathbf{P}^{-1}\boldsymbol{\theta}_j^*\right\}\frac{\sqrt{1+\rho^2}}{1-\rho^2}$$

$$f(\boldsymbol{\alpha}_i \,|\, \mathbf{y}, \sigma^2_{\epsilon_i}, \boldsymbol{\Sigma}_\alpha, \boldsymbol{\Sigma}_\beta) = \text{N}_p(\boldsymbol{\alpha}_i \,|\, \boldsymbol{\mu}_\alpha, \mathbf{V}_\alpha)$$

For the parameter a the following proposal distribution is suggested in order to obtain an irreducible chain when applying the Metropolis algorithm:

$$f(a) = 0.5I_{\{0\}}(a) + 0.5\text{Be}(a|1,1)$$

5. CLUSTERING, SELECTION and FITTING MEASURES

Once the convergence of the MCMC algorithms is assessed, it is possible to make inference using the posterior distribution of interest. In this particular case, we are interested in clustering the observations (that is, our time series).

At each iteration of the Gibbs sampler, a clustering of the parameters $\boldsymbol{\gamma_i}$ is produced (and stored); this partitions are registered not by cluster membership, but by counting the number of iterations that two parameters (e.g. $\boldsymbol{\gamma_i}$ and $\boldsymbol{\gamma_g}$) belong to the same cluster.

Then, a **similarity matrix** can be defined using this information; each element represents the relative frequency of pairwise clustering corresponding to the event that $\boldsymbol{y_i}$ and $\boldsymbol{y_g}$ have the same value of $\boldsymbol{\gamma}$, that is $\boldsymbol{\gamma_i = \gamma_g}$; **each element** is interpreted as **the probability of two time series belonging to the same cluster.**

One possibility is to use the similarity matrix in order to apply a **hierarchical clustering** procedure (as we have seen for the classical hierarchical clustering at the beginning of this presentation).
In this paper, the authors choose to exploit the fact that at **each iteration of the algorithm** produces a ***clustering proposal***. They suggest to select the cluster iteration that minimizes the square deviations with respect to the pairwise clustering matrix (this is in some way like what we have seen about selecting between k-mean results from different random initial configurations, minimizing the within-cluster variation).

In order to do so, a **heterogeneity measure** is defined:

$$\mathrm{HM}(G_1, \ldots, G_m) = \sum_{k=1}^{m} \frac{2}{n_k - 1} \sum_{i<j \in G_k} \sum_{t=1}^{T} (y_{it} - y_{jt})^2$$

where $G_1, \ldots, G_m$ is the sets of indices for a clustering of m clusters with sizes $n_1, \ldots, n_m$. The **bigger HM**, the **more heterogeneous** each clusters is within itself; in the extreme case in which a single time series is a cluster itself (singletons), HM is equal to zero.

Since the algorithm is very flexible and allows different models (e.g. DP, PD, Nstable), we need a model selection criteria. A popular choice in Bayesian statistics is the **Bayes factor**. Assuming that in a model M the data $y_1, \dots, y_n$ arise independently given the model parameter θ, we can obtain

$$BF = \frac{P(y_{1:n}|M_1)}{P(y_{1:n}|M_2)} = \frac{\int P(\theta_1|M_1)P(y_{1:n}|\theta_1, M_1)d\theta_1}{\int P(\theta_2|M_2)P(y_{1:n}|\theta_2, M_2)d\theta_2} = \frac{P(M_1|y_{1:n})}{P(M_2|y_{1:n})} \cdot \frac{P(M_2)}{P(M_1)}$$

This ratio could be very difficult to compute for complex models or even impossible if the prior is not proper. In particular, the density $f(y|M)$ may not exists; then, we can obtain a *pseudo marginal likelihood:*

$$\hat{f}(y|M) = \prod_{i=1}^{n} f_i(y_i|y_{-i}, M)$$

where $f_i(y_i|y_{-i}, M)$ is the *ith* **Conditional Predictive Ordinate (CPO)** , that is the predictive density based on all the data except *i*, evaluated in the observed y_i. From a posterior sample, a Monte Carlo estimation can be obtained (for our case) as follows:

$$\widehat{\text{CPO}}_i = \left(\frac{1}{L} \sum_{l=1}^{L} \frac{1}{f(\mathbf{y}_i|\boldsymbol{\alpha}^{(l)}, \boldsymbol{\gamma}^{(l)}, \boldsymbol{\sigma}_\epsilon^{(l)})} \right)^{-1}$$

The measure proposed by the authors to assess the model fit, is the **Logarithm of the Pseudo Marginal Likelihood (LPML)**, which is a predictive measure for model performance, defined as follows:

$$\widehat{\text{LPML}} = \sum_{i=1}^{n} \log(\widehat{\text{CPO}}_i)$$

Bigger values of LPML indicate a better fit; indeed we could obtain a *pseudo Bayes Factor*

$$\hat{BF} = exp\{LPML_{M_1} - LPML_{M2}\}$$

A value of the BF bigger than 1 means that M_1 is more strongly supported by the data than M_2. According to the table proposed by *Kass and Raftery*:

BF	Strength of evidence
1 to 3.2	Not worth more than a bare mention
3.2 to 10	Substantial
10 to 100	Strong
> 100	Decisive

A value smaller than 1 indicates data evidence in favor of M_2.

6. BNPTSclust PACKAGE

FUNCTIONS

- **Clustering functions:** implement a **Gibbs sampler** to approximate the **posterior distribution** of the model parameters.

    1. **tseriescm:** perform time series clustering for monthly data;
    2. **tseriescq:** perform time series clustering for quarterly data;
    3. **tseriesca:** perform time series clustering for annual data;

In our analysis we employed the **tseriesca function,** since we're dealing with annual data;

- **Plotting functions:**

    1. **Clusterplots:** generate the **plots** of the **time series clustering** provided by the model
    2. **Diagplots:** generate the the **diagnostic plots** to assess the convergence of the Gibbs sampler.

6. BNPTSclust PACKAGE

ARGUMENTS OF TSERIESCA FUNCTION:

1. data: data frame that contains the time series information to be clustered;

2. maxiter: the number of iterations of the Gibbs Sampler;

3. burnin: number of initial iterations to discard;

4. thinning: if set equal to 5 it saves an iteration every five iterations;

5. scale: allows the user to decide whether the time series data should be scaled into the [0,1];

6. level trend and seasonality: the time series are defined as a function of their level, trend, seasonal and temporal components, but that not all of them may be taken into account for clustering. If you want to consider level you set it equal to TRUE;

7. deg: degree of the trend polynomial desired for the model;

8. c0eps, c1eps, c0beta, c0alpha, c1alpha: These are the parameters of the inverse gamma prior on σ^2_ϵ, σ^2_α, σ^2_β;

9. priora, **priorb:** These arguments indicate whether a prior distribution on parameters a and b should be assigned or not. If their value is **FALSE**, then no prior is assigned on the parameters and their **value is fixed** throughout the algorithm;

10. pia, q0a,q1a, q0b,q1b: This are the parameters of the prior distribution proposed for a and b.
pia must be between 0 and 1, and the others must be positive;

11. a, b: if priora or priorb are FALSE, a and b are the fixed values the parameters take throughout the algorithm. If priora or priorb are TRUE, then a and b are the initial values that the parameters take for the algorithm. Please note that a and b must always satisfy that: $0 \leq a < 1$ and $a + b > 0$;

12. lndlpml: This argument indicates if the LPML has to be computed.

6. BNPTSclust PACKAGE

OUTPUT:

1. **mstar:** the number of clusters;

2. **gnstar:** contains the cluster number to which each time series belongs in the final cluster configuration;

3. **HM:** heterogeneity measure;

4. **arrho**, **ara**,**arb**: the parameters ρ, a and b , are sampled whith a MH algorithm. Hence, the variables arrho, ara and arb contain the **acceptance rate** of the simulations for each parameter;

5. **sig2epssample**, **sig2alphasample**, **sig2thesample:** matrices containing in its columns the posterior distribution sample of $\sigma^2_{\epsilon i}$, $\sigma^2_{\alpha i}$, $\sigma^2_{\beta i}$;

6. **rhosample**, **asample**, **bsample** and **msample:** vectors that contain the posterior distribution sample of $\sigma^2_\theta, \rho, a, b$ and the sample of the number groups at each Gibbs sampling iteration saved;

7. **lpml:** the variable contains the value of LPML calculated for the model;

7. OUR RESULTS

In order to choose the more adequate model, we run the algorithm with different **combinations of parameters** and **structure** of the clustering process;

CHOOSING THE COMPONENTS, that describe our times series data, to be included in our clustering process;

Since we're dealing with **annual** data, our time series can be decomposed into **level** and **trend;**

	Combinations of the components to be included in the clustering process
1	Trend with polynomial degree = 1 and Level
2	Trend with polynomial degree = 2 and Level
3	Trend with polynomial degree = 1
4	Trend with polynomial degree = 2
5	Level

PRIOR VARIANCES SELECTION: c_0^k c_1^k $(k = \epsilon, \alpha, \beta)$

The parameters of the hyper-prior distributions influence the number of clusters. To increase the number of groups these parameters should be given values close to zero.

We used $c_0^k = c_1^k =$ **0.01** and $c_0^k = c_1^k =$ **0.001**

TYPES OF PRIOR:

We set our prior to be a (two parameters: a and b)

Poisson-Dirichlet process;

We also analysed two **special cases** of this prior:

1. **Dirichlet process** prior when **a = 0;**
2. **Normalized stable process** when **b = 0**.

Output Table

We stored the outputs of each algorithm run in the following **table**:

				$(c_0^k, c_1^k) = (0.001, 0.001)$			$(c_0^k, c_1^k) = (0.01, 0.01)$		
Trend	Deg	Level	Model	Nr. clust	HM	LPML	Nr. clust	HM	LPML
T	1	T	Dirichlet	2	5.63	2638.95	1	11.10	2128,16
T	1	T	Poi-Dirichlet	2	5.63	2635.5	2	5.13	2324.57
T	1	T	Nstable	2	5.63	2684.27	1	11.10	2136.46
T	2	T	Dirichlet	5	1.89	3138.57	2	2.14	2667.37
T	2	T	Poi-Dirichlet	5	1.89	2669.55	2	2.14	2267.37
T	2	T	Nstable	5	1.89	3141.68	2	5.13	2327.01
T	1	F	Dirichlet	11	10.36	3591.27	1	11.10	2622.65
T	1	F	Poi-Dirichlet	6	10.83	3591.27	1	11.10	2615.18
T	1	F	Nstable	4	10.74	3591.24	1	11.10	2582.31
T	2	F	Dirichlet	3	10.97	3705.73	1	11.10	2664.54
T	2	F	Poi-Dirichlet	4	10.81	3718.37	1	11.10	2664.43
T	2	F	Nstable	4	10.84	3736,16	1	11.10	2655.83
F		T	Dirichlet	50	3.84	2317.99	32	8.03	1104.73
F		T	Poi-Dirichlet	53	5.23	2374.17	30	7.43	1110.00
F		T	Nstable	51	3.6	2330.45	32	8.03	1138.00

* The table above has been obtained setting the same seed (*set.seed(123)*)

Looking at the **output table** we can observe the following:

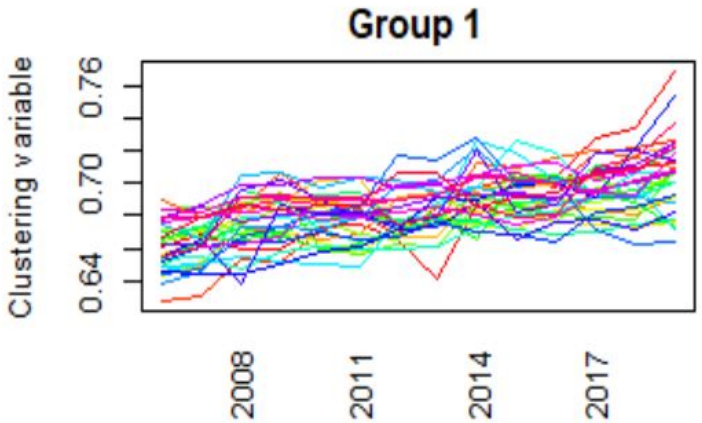
For our data set the **optimal choice** for the parameters of the variance prior distributions (that is, the parameters of the Inverse Gammas) is $\boldsymbol{c_0^k = c_1^k = 0.001}$. As we said before, the value of these parameters affects the result in a strong way; in particular, as we can see from the table, larger values of c produce a smaller number of clusters, sometimes leading to just one cluster (that is, the entire population belonging to the same group), with a **very large heterogeneity measure**.

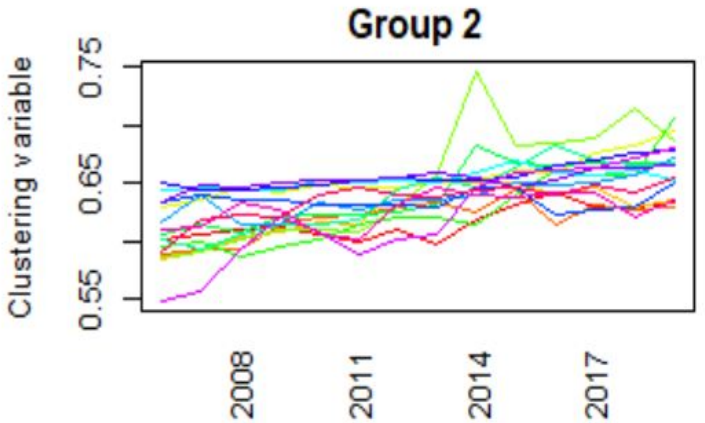
Also, for values of the parameter close to zero, the algorithm is computationally more intensive (on average, one hour of running time). Finally, the **LPML with** $\boldsymbol{c_0^k = c_1^k = 0.01}$ is **always smalle**r than the corresponding case with a smaller value of the parameter, showing a **worse fit** of the model.

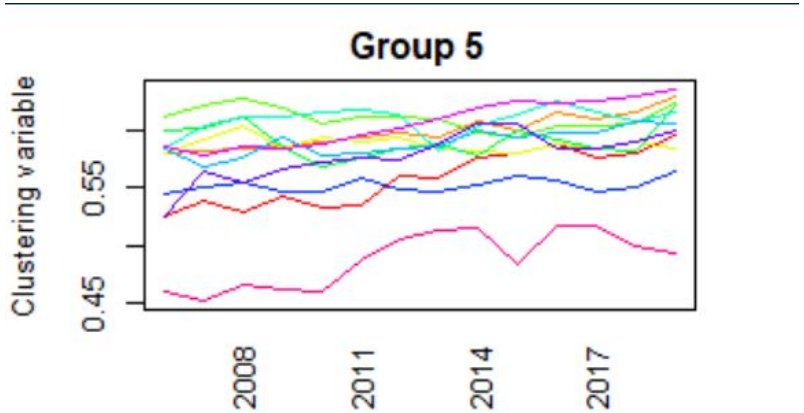
The result that is most similar to the one obtained with the standard clustering methods used in the first part of the presentation is the one obtained with setting as clustering criteria a **quadratic trend** and the **level** of the time series $(c_0^k = c_1^k = 0.001)$. Looking at the model with the biggest LPML, that is the **Nstable one**:

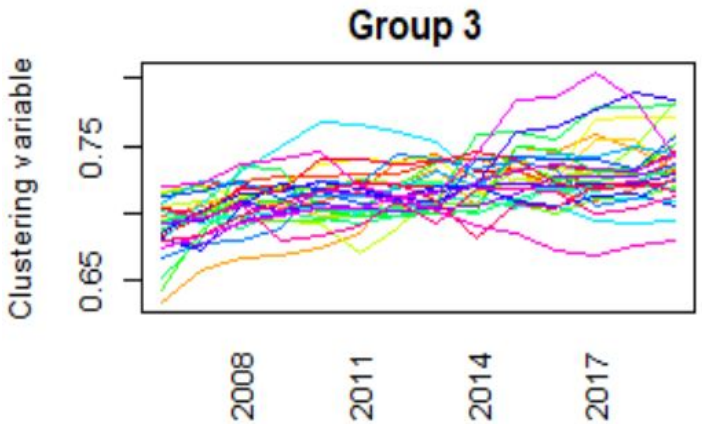
CLUSTER 1		CLUSTER 2	CLUSTER 3		CLUSTER 4	CLUSTER 5
Albania	Kenya	Algeria	Argentina	Lesotho	Belgium	Chad
Bangladesh	Kyrgyz Republic	Bahrain	Australia	Lithuania	Denmark	Egypt
Botswana	Madagascar	Burkina Faso	Austria	Luxembourg	Finland	Iran Islamic
Brazil	Malawi	Cambodia	Bolivia	Moldova	Germany	Rep. Jordan
Chile	Malta	Cameroon	Bulgaria	Mongolia	Iceland	Mali
China	Mexico	Ethiopia	Canada	Namibia	Ireland	Mauritania
Cyprus	Paraguay	Guatemala	Colombia	Panama	Latvia	Morocco
Czech Republic	Peru	India	Costa Rica	Poland	Netherlands	Pakistan
Dominican	Romania	Japan	Croatia	Portugal	New Zealand	Saudi Arabia
Republic	Singapore	Korea Rep.	Ecuador	Slovenia	Nicaragua	Turkey
El.Salvador	Slovak Republic	Kuwait	Estonia	Sri Lanka	Norway	Yemen
Georgia	Thailand	Malaysia	France	Tanzania	Philippines	
Ghana	Ukraine	Mauritius	Israel	Uganda	South Africa	
Greece	Uruguay	Nepal	Jamaica	United States	Spain	
Honduras	Venezuela	Nigeria	Kazakhstan		Sweden	
Hungary		United Arab			Switzerland	
Indonesia		Emirates				
Italy						
Russian						
Federation						

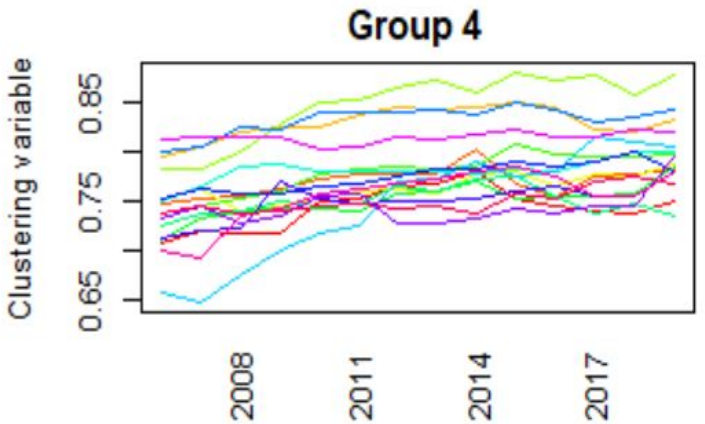
With the function Clusterplots, we can obtain the following plots of the clusters:











Including both the level and the trend (with polynomial degree 2), the algorithm produces 5 clusters.
In particular, the cluster of top performer countries in the hierarchical method (that is, cluster 4), is contained in **group 4** obtained with the **non parametric model**; also, cluster 3 obtained with the K-Medoid algorithm contains group 4 of the non parametric output. So, we can say that the three results are somewhat coherent with each other and with the World Economic Forum ranking.

The same relationship between clustering outputs can be observed for the worst performer group of countries, that is **group 5** and **group 2** of the **non parametric method**, cluster 2 and cluster 3 of hierarchical model and cluster 2 of K-Medoid algorithm.

Also, this model structure produced the clusters with the **smallest** heterogeneity measure (**HM**) when compared with the other ones.

Therefore, we conclude that considering both the (quadratic) trend and the level of the time series of our data set we can obtain reasonable and **coherent results**, knowing the general ranking proposed by the Gender Gap Index annual reports.

The analyses we have done so far were based on distances between time series or measures of dissimilarity. However, with the flexibility of this last method we were able to observe data characteristics that standard methods (frequentist approach) didn't highlight. For instance, taking the result of the algorithm including the **(linear) trend** of the series, we obtain a very interesting and surprising result. Taking the best result, that is, the one with smaller HM and larger LPML (**Dirichlet model**), we obtain the following output:

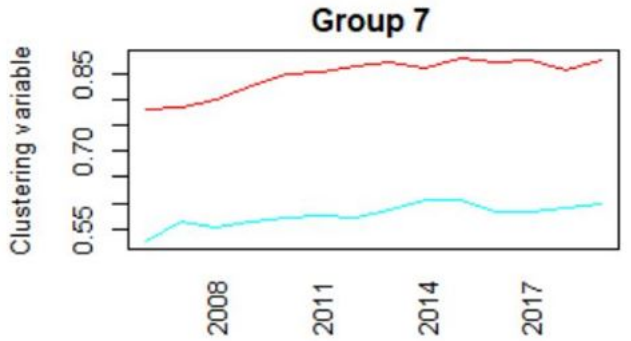
CLUSTERS/ GROUPS										
1	2	3	4	5	6	7	8	9	10	11
Albania Bangladesh Bolivia Cameroon Ethiopia France Guatemala India Kenya Mexico Namibia Nepal Nicaragua Slovenia Turkey	Algeria Argentina Bahrain Belgium Botswana Brazil Bulgaria Burkina Faso Canada Chad Chile Colombia Costa Rica Croatia Cyprus Czech Republic Denmark Ecuador	Australia Austria China Finland Ireland Japan Kazakhstan Kuwait Kyrgyz Republic Luxembourg Malawi Netherlands Russia Sweden Switzerland	Cambodia Dominican Republic Jamaica	Ghana	Hungary Iran Mali Morocco Slovak Republic	Iceland Saudi Arabia	Jordan Lesotho Sri Lanka	Mauritania Tanzania	New Zealand	Pakistan

In our opinion, it is very **interesting** to observe how the possibility to set the criteria of clustering allowed us to obtain a clustering result that groups the countries based only on the trend of the Gender Gap index over time, independently on the atomic value of the index itself.

We find **cluster 7** a very curious result in this matter. **Iceland** has been the **top performer** in the majority of the reports, while **Saudi Arabia** is still one of the countries in which the gender gap is the biggest. This explains why using the standard clustering methods these two countries are always assigned to two different clusters.

Nevertheless, it is interesting to see how the Bayesian non parametric model it's able to grasp a similarity in trend between these two very different countries.

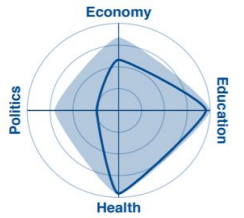
Although the atomic value of the index of these two countries is very different, we can say that for the last 14 years, their gender gap measures has **moved** in the **same way**.



Iceland

rank (out of 153 countries) **1**

score (0.00 = imparity, 1.00 = parity) **0.877**

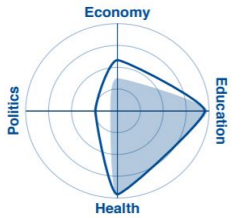


	2006 score		2020 score	
Global Gender Gap Index	**4**	**0.781**	**1**	**0.877**
Economic participation and opportunity	17	0.711	2	0.839
Educational attainment	50	0.991	36	0.999
Health and survival	92	0.968	123	0.968
Political empowerment	4	0.456	1	0.701

Saudi Arabia

rank (out of 153 countries) **146**

score (0.00 = imparity, 1.00 = parity) **0.599**



	2006 score		2020 score	
Global Gender Gap Index	**114**	**0.524**	**146**	**0.599**
Economic participation and opportunity	115	0.240	148	0.375
Educational attainment	93	0.880	92	0.983
Health and survival	54	0.977	139	0.963
Political empowerment	115	0.000	136	0.077

We also tried to use include **just** the **level** as clustering criteria; the output resulted in a lot of **singletons**, even with larger prior variance parameters (see last three rows of the table above).
We think that with including only the level, we are not exploiting the full potential of the model; indeed, we are **losing information**, considering we are dealing with time series data.

In some way, this is the same problem we had with the **K-Means** algorithm in the very first part of this presentation, since we were looking for an approach that did not focus (only) on the atomic value of the index.
Focusing on the atomic value of variables is useful when dealing with **steady object**; we find that with time series it's more appropriate to apply a model that captures also similarities over time.

Looking at the **LPML**, the outputs of the **last three rows** of the table are the **worst ones**; this means that it is **more likely** for the data to be **generated** according to the **other models**.
Also, we noted that for these models the running time of the algorithm is the highest.

FINAL CONSIDERATIONS

The field of Time Series Clustering is still growing, as each year new models arise;

We were able to appreciate the **computational efficiency** of the **Standard Approach**, and the instruments provided to deal with time series, like **Distance Time Warping**.

On the other hand, we realized that when dealing with such special object, like time series that describe a social phenomenon such as Gender Gap, **Non Parametrics Bayesian Statistics** provides **highly flexible models,** that allow to better manage all the different factors that influence the problem. The Bayesian algorithms are **computationally intensive** and, therefore, a trade-off arises.

In the end, the results we obtained from both the approaches are coherent and possess informational value. However, the last analysis gave us a broader perspective of the phenomenon we analyzed.

FINAL CONSIDERATIONS

Our analysis confirmed the content of the World Economic Forum Reports. In particular, the scandinavian countries were mostly in the same cluster as **top performers** with other countries, such as: Philippines, New Zealand, Nicaragua, South Africa…;

Also, the **worst performing** countries were mostly in the same cluster (e.g. Chad, Egypt, Iran, Pakistan, Saudi Arabia, Turkey, Yemen);

Also, having extended the analysis over 14 years we could appreciate a **temporal perspective** of the change of the gender gap around the world;

Very different countries may have the **same trend of change over time**, which can provide **relevant insights** to all the countries for understanding the matter and how to improve;
Due to the **social importance** of this phenomenon, we think that a **broader view** of this problem could be **helpful** to the **policy makers**.