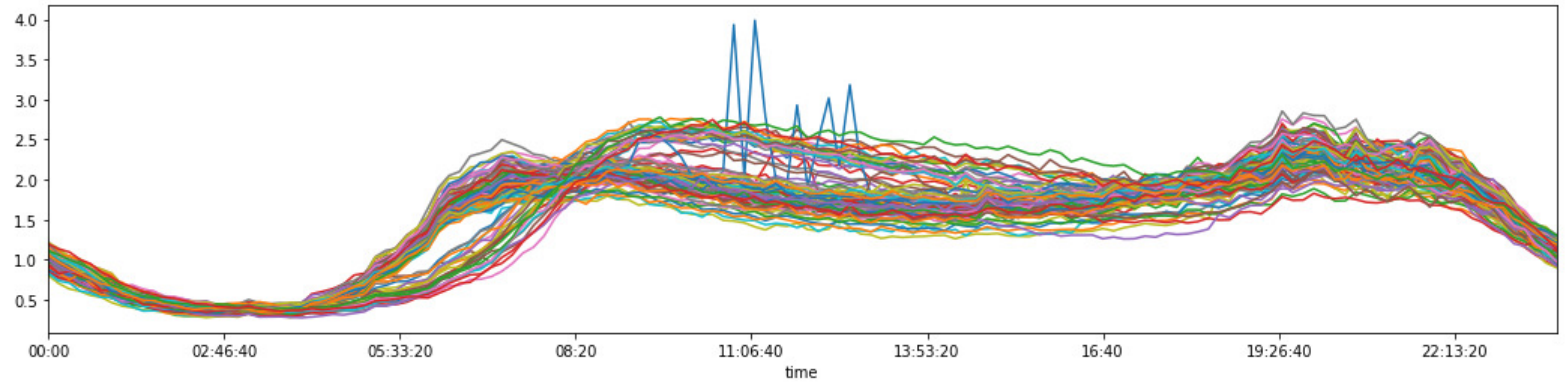
Advanced Data Mining

Piotr Lipiński

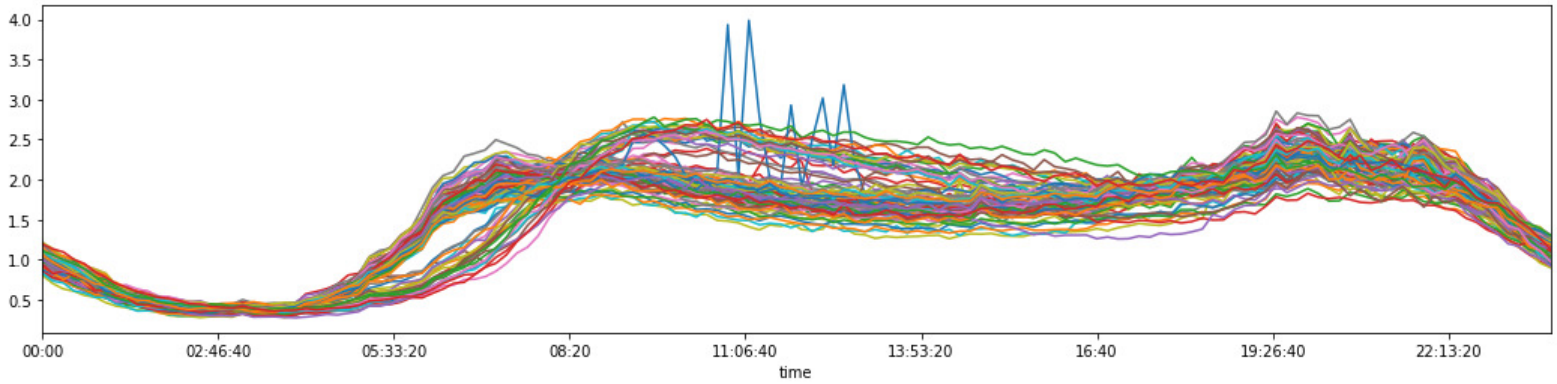
Time Series Clustering

- How to cluster time series?
  - see the jupyter python notebook with some examples



Time Series Clustering

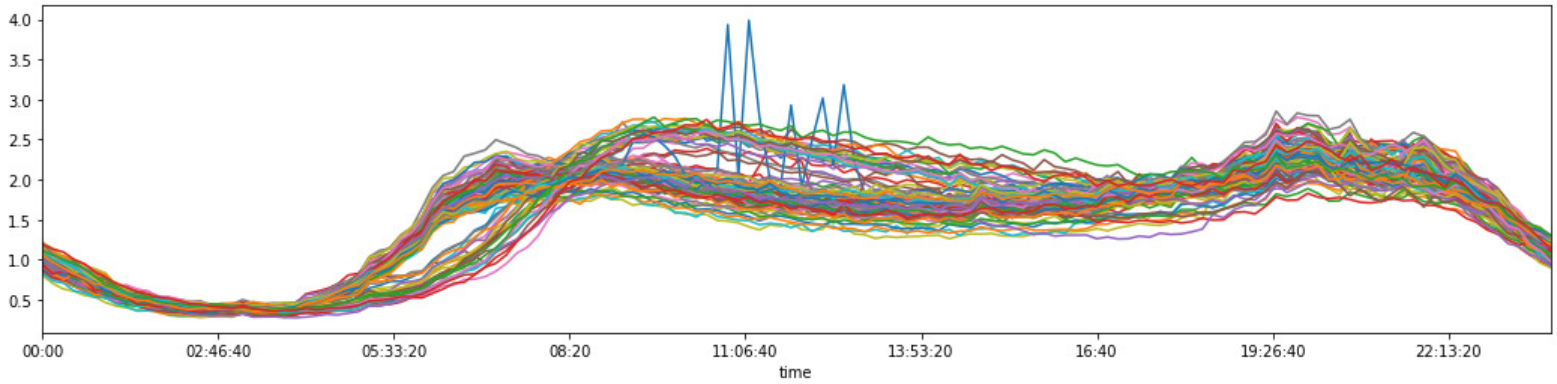
- How to cluster time series?
  - see the jupyter python notebook with some examples



  - **APPROACH 1:** (for regular time series)
    - daily (or weekly, monthly, annual) average profiles may define patterns
    - each day (or week, month, year) time series can be matched to one of these patterns
    - the time series matched to the same pattern define the cluster
    - how to match time series to the patterns?

Time Series Clustering

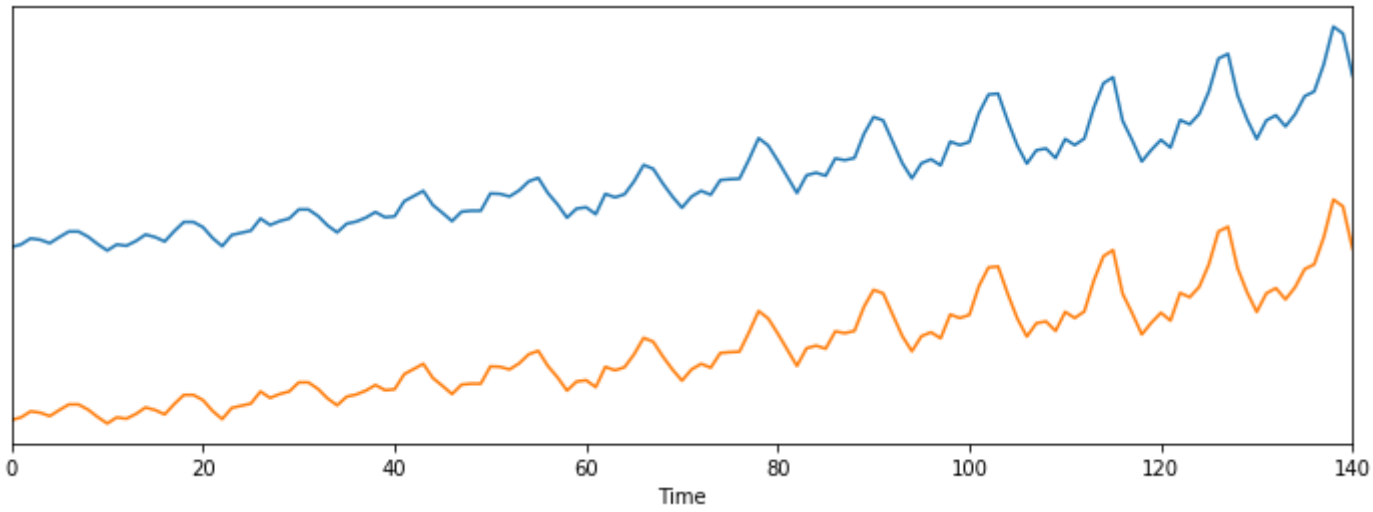
- How to cluster time series?
  - see the jupyter python notebook with some examples



  - **APPROACH 2:**
    - consider the time series as a vector of numbers, apply one of classic clustering algorithms, e.g. k-means, DBScan, etc.
    - **Difficulties:**
      - time series may be of various length
      - similar, but shifted or rescaled time series will lead to dissimilar vectors
  - **APPROACH 3:**
    - stay with original time series, but introduce a distance measure over them

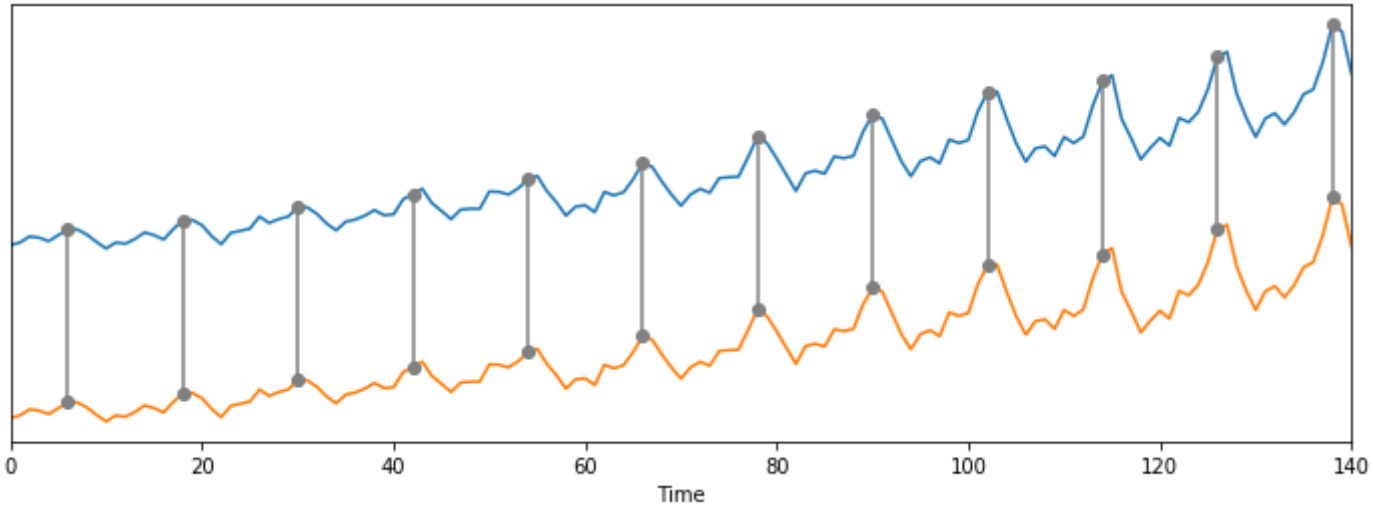
Similarities between Time Series

- How to measure similarities between time series?



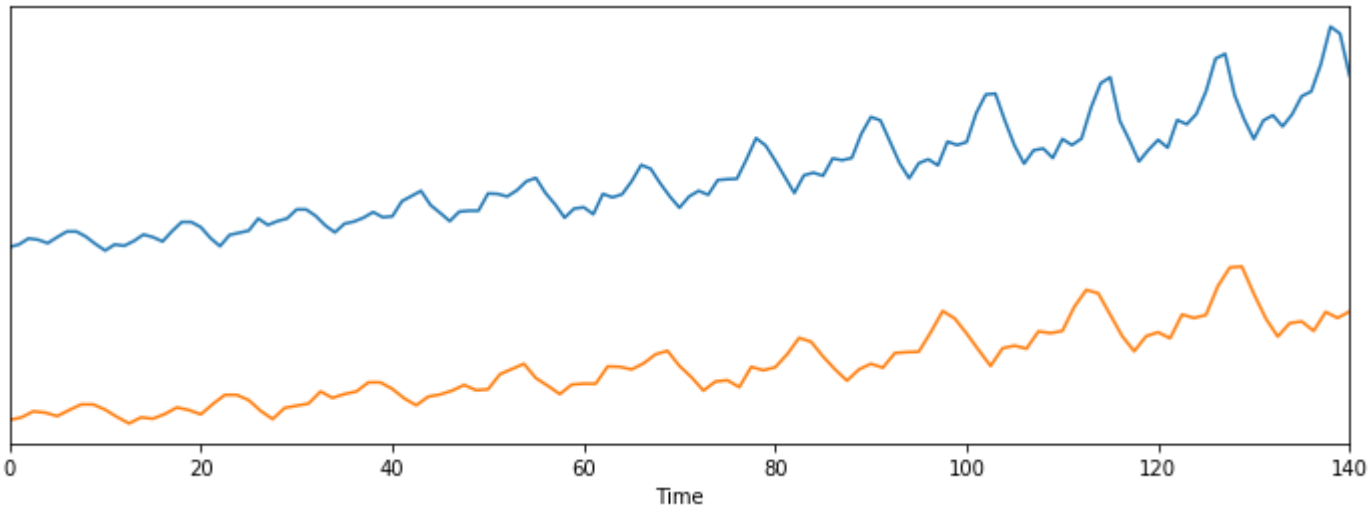
Similarities between Time Series

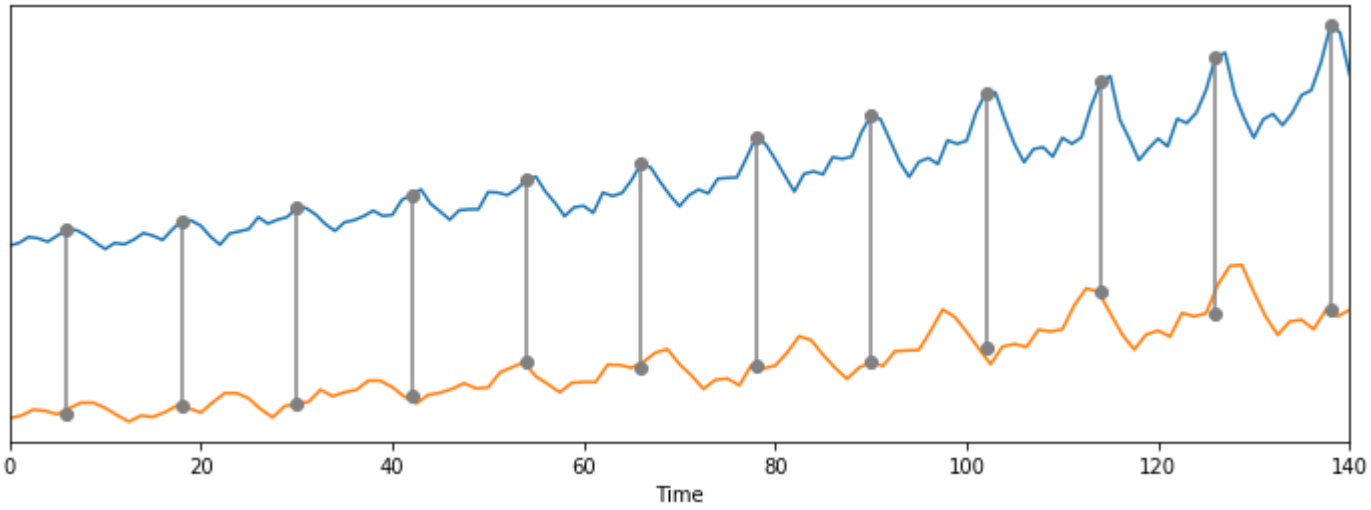
- How to measure similarities between time series?
  - Euclidean distance



Similarities between Time Series

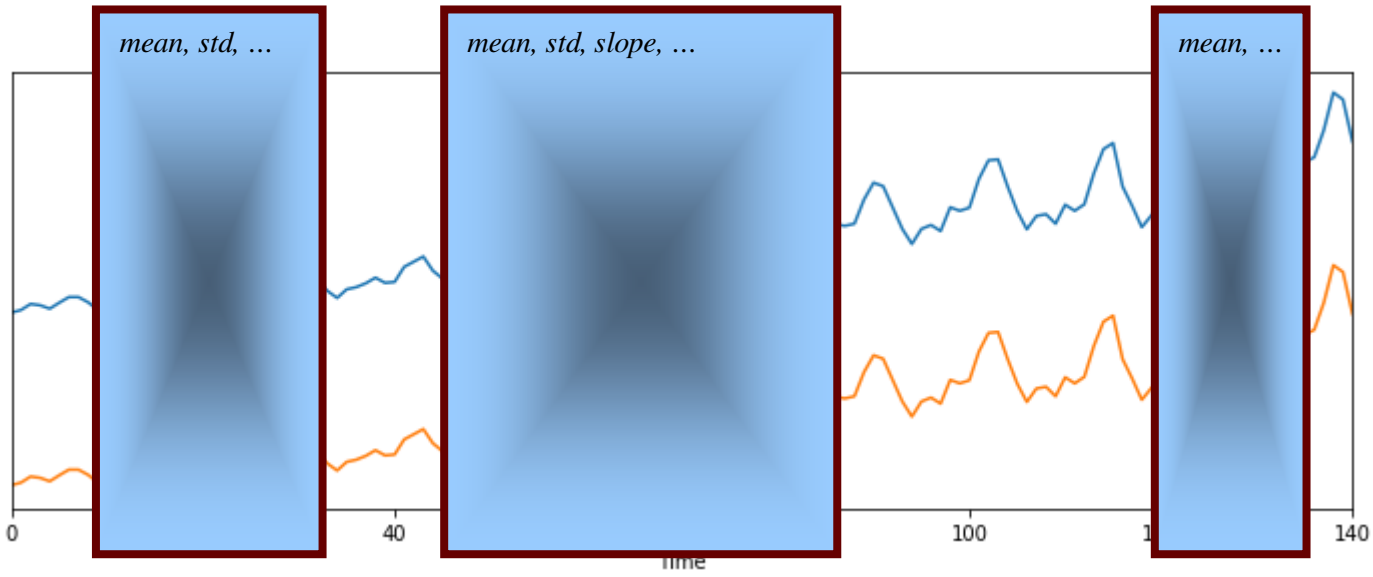
- How to measure similarities between time series?
  - Euclidean distance





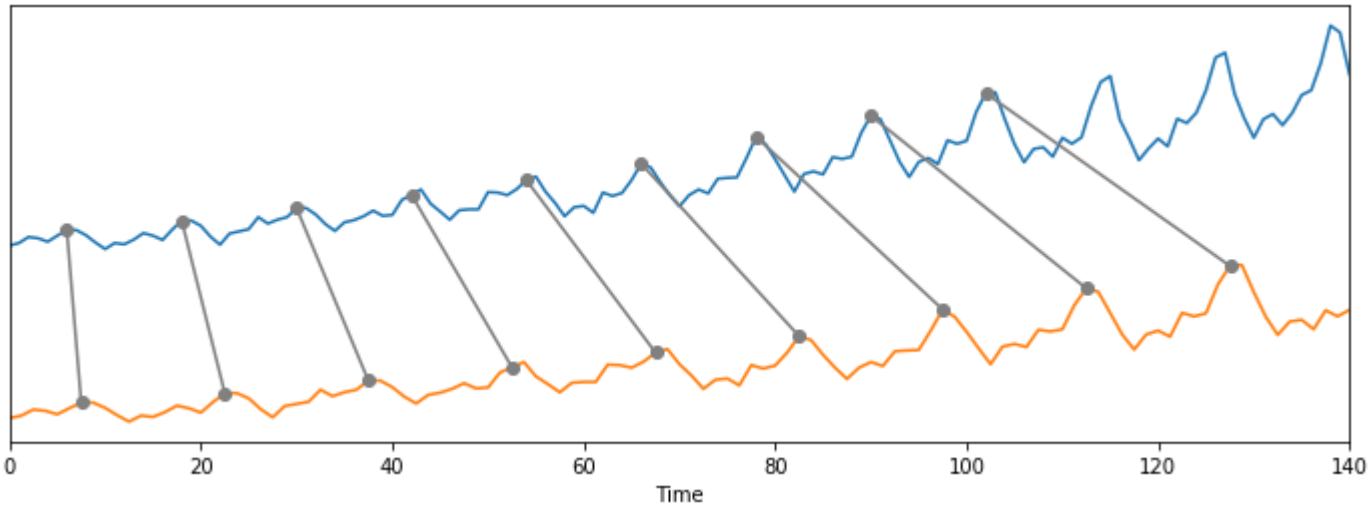
Similarities between Time Series

- How to measure similarities between time series?
  - Euclidean distance / Manhattan distance / cosine distance / etc.
  - Euclidean distance with feature-based representation



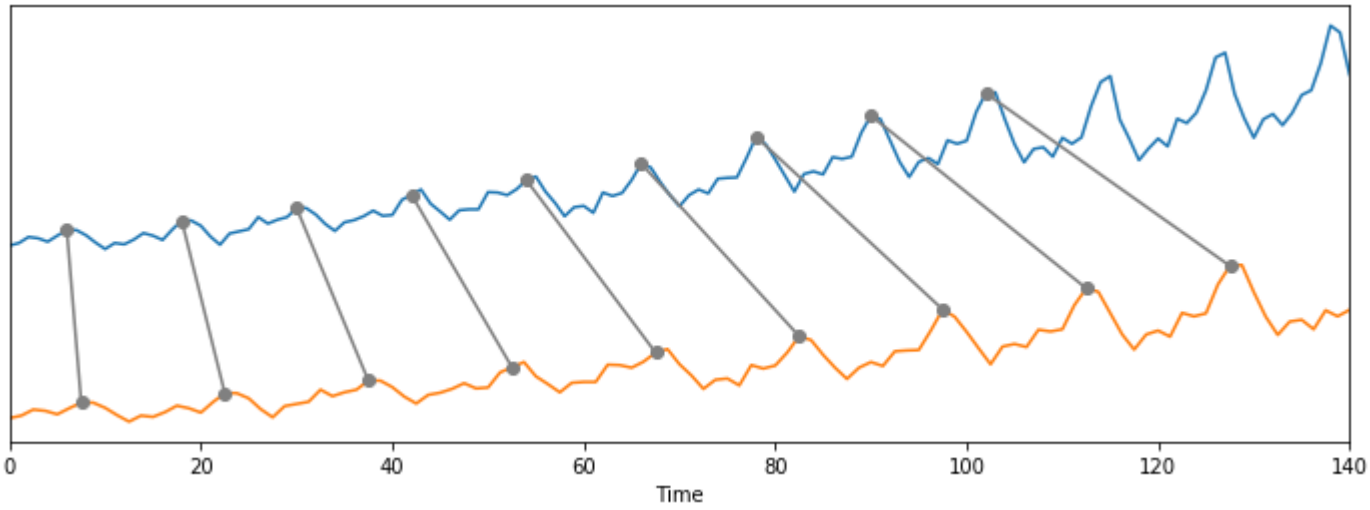
Similarities between Time Series

- How to measure similarities between time series?
  - Euclidean distance / Manhattan distance / cosine distance / etc.
  - Euclidean distance with feature-based representation
  - **Dynamic Time Warping (DTW)**
    - DTW tries to synchronize time series before comparing



Dynamic Time Warping (DTW)

- **Dynamic Time Warping (DTW)**
  - consider s[1..n] and t[1..m]
  - for each i = 1..n, s[i] must be matched with one or more elements of t
  - for each k = 1..m, t[k] must be matched with one or more elements of s
  - s[1] must be matched with t[1]
    (but may be also matched with other elements of t, t[1] similarly)
  - s[n] must be matched with t[m]
    (but may be also matched with other elements of t , t[m] similarly)
  - the mapping of elements of s to elements of t must be monotonically increasing,
    - for each i, j = 1..n, if i < j, then for each k, l = 1..m if s[i] matches t[k] and s[j] matches t[l], then k <= l



Dynamic Time Warping (DTW)

```
DTW-Distance(s: array[1..n], t: array[1..m])

    DTW := array[0..n, 0..m]
    for i := 0 to n
        for j := 0 to m
            DTW[i, j] := infinity

    DTW[0, 0] := 0
    for i := 1 to n
        for j := 1 to m
            DTW[i, j] := d(s[i], t[j]) + min(
                DTW[i-1, j],
                DTW[i, j-1],
                DTW[i-1, j-1])

    return DTW[n, m]
```

Time Series Clustering

- How to cluster time series?
  - **APPROACH 3:**
    - stay with original time series, but introduce a distance measure over them
    - **Difficulties:**
      - **DTW:** may be inefficient for larger time series
        - O(nm)
        - possible improvements, e.g. PrunedDTW, SparseDTW, FastDTW, MultiscaleDTW, etc.
      - **K-means:** seemingly easy, but
        - how to define the center of the cluster for DTW distance?
        - minimizes the quantisation error, related to Euclidean distance, not to DTW distance

REMINDER: k-means

- Let D = $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ be the dataset of N data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.
  Let K be the number of cluster to define.
- Each cluster C_k is defined by a point $\mathbf{r}_k$, called the center of the cluster.
  Each data vector is assigned to the cluster of the closest center.
  - in the case of equal distances to a few centers, they may be considered in a predefined order or by random
- The goal is to discover a partition $C = \{C_1, C_2, \ldots, C_K\}$ of the set D of the size K (i.e. K parwise disjoint sets $C_1, C_2, \ldots, C_K$ such that $C_1 \cup C_2 \cup \ldots \cup C_K = D$) minimizing the criterium function

$$F(C) = \sum_{k=1}^{K} \sum_{\mathbf{x} \in C_k} \| \mathbf{x} - \mathbf{r}_k \|^2$$

- k-means is one of the algorithms solving such a problem

REMINDER: k-means

- Minimization of the criteria function

$$F(C) = \sum_{k=1}^{K} \sum_{\mathbf{x} \in C_k} \| \mathbf{x} - \mathbf{r}_k \|^2$$

may be performed iteratively in two steps:

- having vectors $\mathbf{r}_k$, find the optimal assignment of data vector to the clusters – obviously: each data vector should be assigned to the cluster of the closest center $\mathbf{r}_k$
- having the assignment of data vector to the clusters, find vectors $\mathbf{r}_k$
  - it is less obvious – mathematical approach based on derivatives
  - the solution is to set $\mathbf{r}_k$ in the barycenter of the set of vectors forming the cluster

REMINDER: k-means

- k-means

FOR k = 1, 2, …, K

r_k = a random point from D

WHILE there are still changes in C_k

FOR k = 1, 2, …, K

$C_k = \{x \in D : d(x, r_k) < d(x, r_l)$ for each $l = 1, 2, \ldots, K, l \neq k\}$

FOR k = 1, 2, …, K

r_k = barycenter of C_k

Time Series Clustering

- How to cluster time series?
  - **APPROACH 3:**
    - stay with original time series, but introduce a distance measure over them
    - **Difficulties:**
      - **DTW:** may be inefficient for larger time series
        - O(nm)
        - possible improvements, e.g. PrunedDTW, SparseDTW, FastDTW, MultiscaleDTW, etc.
      - **K-means:** seemingly easy, but
        - how to define the center of the cluster for DTW distance?
        - minimizes the quantisation error, related to Euclidean distance, not to DTW distance
  - **APPROACH 4:**
    - DTW Barycenter Averaging k-means (**DBA-k-means**)

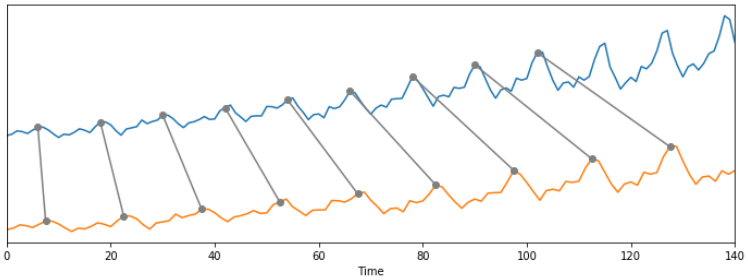
DTW Barycenter Averaging k-means (DBA-k-means)

$C = [C_1, C_2, \ldots, C_{T'}]$ // the initial average sequence
$S_1 = [s_{11}, s_{12}, \ldots, s_{1T}]$ // the 1st sequence to average
$S_2 = [s_{21}, s_{22}, \ldots, s_{2T}]$ // the 2nd sequence to average
…
$S_n = [s_{n1}, s_{n2}, \ldots, s_{nT}]$ // the nth sequence to average

A := array[1..T'] // a table of sets
// A[i] contains the set of coordinates matched with C[i]

A := [0, 0, …, 0]
for S in $[S_1, S_2, \ldots, S_n]$
    M := DTW(C, S) // a temporary DTW table
    i := T'
    j := T
    while i >= 1 **and** j >= 1
        A[i] := A[i] ∪ {S[j]}
        (i, j) := NextAssociation(M[i, j])
for i := 1 **to** T
    C[i] := mean(A[i])

return C



F. Petitjean, A. Ketterlin, P. Gancarski, "A global averaging method for dynamic time warping, with applications to clustering". Pattern Recognition, 3(44), 2011, pp.678-693.