Different Similarity Measures

$$f^p, f^q : timeseries$$

Manhattan Distance

- Quantifies the absolute magnitude of the difference between time series
- Easy to calculate

$$D_{Man} = (\sum_{t=1}^{N} |f_t^p - f_t^q|)$$

- Implementation: scipy.spatial.distance.cityblock
- Value Range [0, ∞]

Euclidean

- Quantifies the Euclidean distance of the difference between time series
- Easy to calculate
- More sensitive to outliers, due to it's nonlinear character

$$D_E = \sqrt{(\sum_{t=1}^{N} |f_t^p - f_t^q|^2)}$$

- Implementation: scipy.spatial.distance.euclidean
- Value Range [0, ∞]

Pearons's Correlation

- Quantifies the degree of linear relationship between time series

$$D_{CC} = \frac{\sum_{t=0}^{N-1}((f_t^p - \overline{f^p}) * (f_{t-s}^q - \overline{f^q}))}{\sqrt{\sum_{t=0}^{N-1}(f_{t-s}^p - \overline{f^p})^2} * \sqrt{\sum_{t=0}^{N-1}(f_{t-s}^q - \overline{f^q})^2}}$$

- Implementation: numpy.corrcoef
- Value Range [-1, 1]

Cosine Distance

- Metric used to determine how similar time series are irrespective of their magnitudes.

$$D_{Cos} = \frac{\sum_{i=1}^{n} f_i^p f_i^q}{\sqrt{\sum_{i=1}^{n}(f_i^p)^2}\sqrt{\sum_{i=1}^{n}(f_i^q)^2}}$$

- Implementation: scipy.spatial.distance.cosine
- Value Range [0, 1]

Principal Component Distance

- Computes the difference between time series mapped into the first m PCs that explain the majority of the variance.
- Selecting m critical

$$D_{PCA} = \sqrt{\sum_{k=1}^{m}(PC_k^p - PC_k^q)^2}$$

- Implementation: sklearn.decomposition.PCA
- Value Range [0, ∞]

Mutual Information

- Measure of the amount of mutual dependence between two random variables

$$MI(f^p, f^q) = -\sum_{f_i^p, f_i^q} p(f_i^p, f_i^q) log_2 \frac{p(f_i^p, f_i^q)}{p(f_i^p)p(f_i^q)}$$

- Implementation: pyinform.mutualinfo.mutual_info
- Value Range [0, ∞]

Transfer Entropy

- Quantify information transfer between an information source and destination, conditioning out shared history effects

$$T_{f^p -> f^q} = H(f_t^q | f_{t-1:t-L}^q) - H(f_t^q | f_{t-1:t-L}^q, f_{t-1:t-L}^p)$$

- Implementation: pyinform.transferentropy.transfer_entropy
- [0, 1] inverted

Conditional Entropy

- Measure of the amount of information required to describe a random variable f^q given knowledge of another random variable f^p

$$H(f^q|f^p) = -\sum_{f_i^p, f_i^q} p(f_i^p, f_i^q) log_2 \frac{p(f_i^p, f_i^q)}{p(f_i^p)}$$

- Implementation: pyinform.conditionalentropy.conditional_entropy
- Value Range [0, ∞]

Dynamic Time Warping

- Dynamic time warping is an algorithm used to measure similarity between two sequences which may vary in time or speed.
- It works as follows:
  1. Divide the two series into equal points.
  2. Calculate the euclidean distance between the first point in the first series and every point in the second series. Store the minimum distance calculated. (this is the ‘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.
- Implementation: similaritymeasures.dtw
- Value Range [0, ∞]

Spearman's Correlation

- A nonparametric measure of rank correlation (statistical dependence between the rankings of two variables). It assesses how well the relationship between two variables can be described using a monotonic function.

- To calculate Spearman's correlation we first need to map each of our data to ranked data values:

$$x \rightarrow x^r$$

- If the raw data are [0, -5, 4, 7], the ranked values will be [2, 1, 3, 4]

$$D_{SPC} = \frac{\sum_{t=0}^{N-1}(((f_t^p)^r - (\overline{f^p})^r) * ((f_{t-s}^q)^r - (\overline{f^q})^r))}{\sqrt{\sum_{t=0}^{N-1}((f_{t-s}^p)^r - (\overline{f^p})^r)^2} * \sqrt{\sum_{t=0}^{N-1}((f_{t-s}^q)^r - (\overline{f^q})^r)^2}}$$

- Implementation: scipy.stats.spearmanr
- Value Range [0, 1]

Kendall's Tau

- A statistic used to measure the ordinal association between two measured quantities. A τ test is a non-parametric hypothesis test for statistical dependence based on the τ coefficient.
- To To calculate Kendall's Tau we first need to map each of our data to ranked data values:

$$x \rightarrow x^r$$

- If the raw data are [0, -5, 4, 7], the ranked values will be [2, 1, 3, 4]

$$TAU = \frac{2}{n(n-1)} \sum_{i<j} sgn((f_i^p)^r - (f_j^p)^r) sgn((f_i^q)^r - (f_j^q)^r)$$

$$sgn : signumfunction$$

- Implementation: scipy.stats.kendalltau
- Value Range [0, 1]

Maximum Information Coefficient

- MIC captures a wide range of associations both functional and not, and for functional relationships provides a score that roughly equals the coefficient of determination (R^2) of the data relative to the regression function

- For a grid G, let Ig denote the mutual information of the probability distribution induced on the boxes of G, where the probability of a box is proportional to the number of data points falling inside the box. The (x,y)-th entry m_x,y of the characteristic matrix equals max{Ig}/(log min{x,y}), where the maximum is taken over all x-by-y grids G. MIC is the maximum of m_x,y over ordered pairs (x,y) such
  that x*y < B, where B is a function of sample size; we usually set B=n^0.6
- Implementation: minepy.MINE.mic()
- Value Range [0, ∞]

Randomized Dependence Coefficient

- Measure of nonlinear dependence between random variables of arbitrary dimension based on the Hirschfeld-Gebelein-Renyi Maximum Correlation Coefficient

$$rdc(f^p, f^q; k, s) := sup_{a,b}(a^T \varphi(P(f^p); k, s), b^T \varphi(P(f^q); k, s)$$

with

$$P : Copula - Transformation$$

$$\varphi(wTx + b) := sin(wTx + b).$$

$$k \in \mathbb{N}_+, s \in \mathbb{R}_+$$

- Implementation: https://github.com/garydoranjr/rdc
- Value Range [0, 1]

Distance Correlation

- in [0, 1] with 0 = completely independent
- sensitive to all types of departures from independence, including nonlinear or nonmonotone dependence structure.
- Distance Correlation

$$D_{Cor} = \sqrt{\frac{V_n^2(f^p, f^q)}{V_n^2(f^p)V_n^2(f^q)}}$$

With Distance Covariance

$$V_n^2(X, Y) = \frac{1}{n^2} \sum_{k,l=1}^{n} A_{k,l} B_{k,l}$$

$$V_n^2(X) = V_n^2(X, X)$$

And

$$A_{k,l} = a_{k,l} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}$$

$$a_{k.l} = |X_k - X_l|_p$$

$$\bar{a}_{k\cdot} = \frac{1}{n} \sum_{l=1}^{n} a_{k,l}$$

$$\bar{a}_{\cdot l} = \frac{1}{n} \sum_{k=1}^{n} a_{k,l}$$

$$\bar{a}_{\cdot\cdot} = \frac{1}{n^2} \sum_{k,l=1}^{n} a_{k,l}$$

B is defined analogously for Y

- Implementation: https://gist.github.com/satra/aa3d19a12b74e9ab7941
- Value Range [0, 1]