TSUBASA-PLUS: Correlation Matrix Computation on Sliding Windows

Yunlong (Draco) Xu[1], Jinshu Liu[2], Peizhen (Aaliyah) Yang[3], Noah Viso[1] and Fatemeh Nargesian[1]

[1]Department of Computer Science, University of Rochester; [2]Department of Computer Science, Virginia Tech; [3]Simon Business School, University of Rochester

MIT IEEE URTC 2022

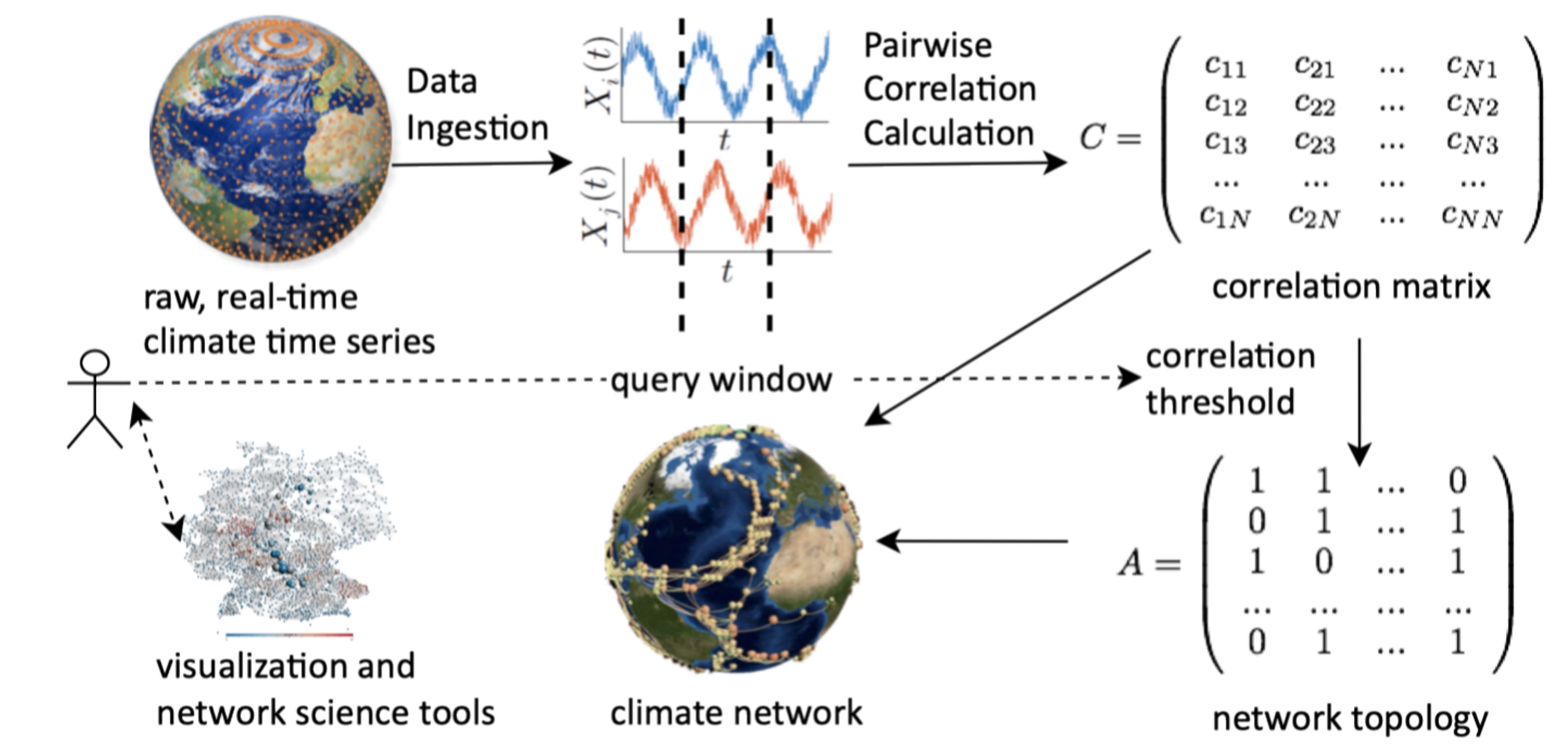
EXACT CALCULATION OF CORRELATION MATRIX

Given a collection $L = \{x^1, \ldots, x^n\}$ of historical or real-time time-series, the goal is to efficiently compute the correlation matrix of L, i.e. $Corr(x^i, y^j)$ for any $i \neq j$.

$$Corr(x, y) = \frac{\sum_{i=1}^{m}(\mathbf{x}_i - \bar{x})(\mathbf{y}_i - \bar{y})}{\sqrt{\sum_{i=1}^{m}(\mathbf{x}_i - \bar{x})^2}\sqrt{\sum_{i=1}^{m}(\mathbf{y}_i - \bar{y})^2}}$$

CORRELATION NETWORK CONSTRUCTION

- Given a query window, a correlation matrix is constructed by computing the pairwise correlation of all time-series on the query window.
- The quadratic complexity of all-pair correlation calculation makes network construction a laborious task, particularly, for interactive data analysis.
- Example: to identify and analyze patterns in global climate, scientists model climate data as climate networks [1, 2, 3]. Nodes in a climate network are geographical locations, characterized by time-series and edges represent correlation between nodes.

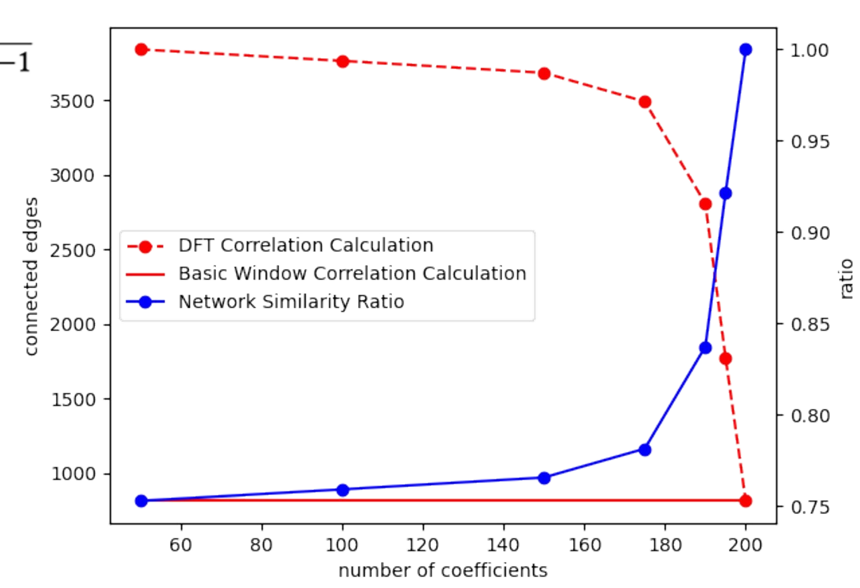


EXISTING APPROXIMATE TECHNIQUES

- The existing solutions [9,10] divide time-series into basic windows and use the Discrete Fourier Transform (DFT) to reduce the dimensionality of the data.

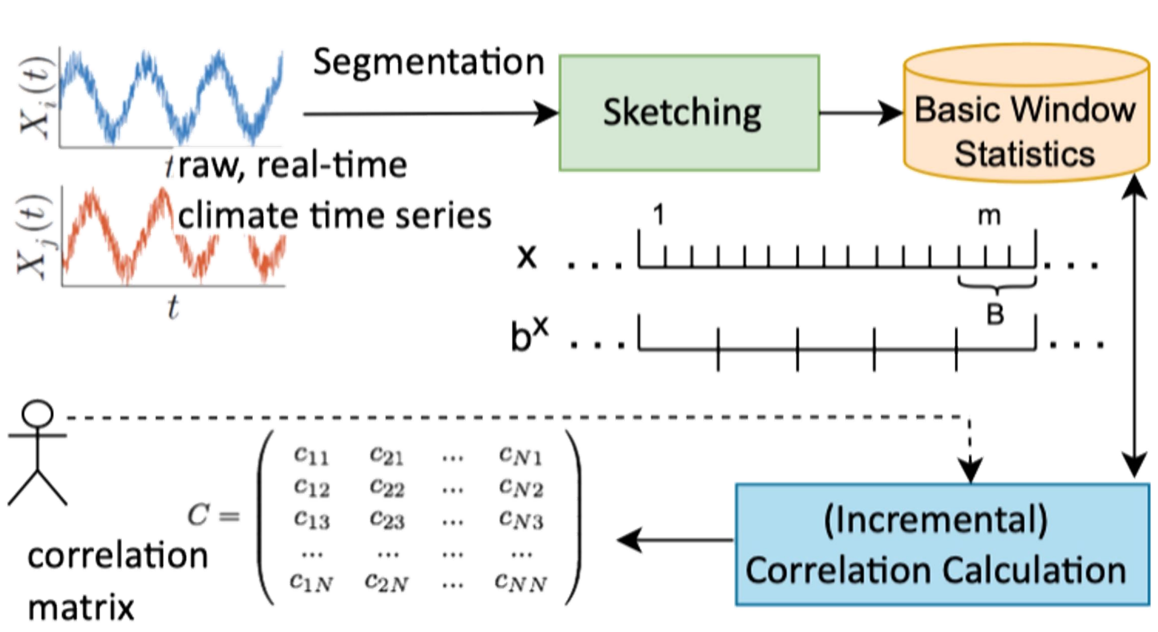
$$\mathbf{X}_f = \frac{1}{\sqrt{k}}\sum_{i=1}^{k}\mathbf{x}_i e^{\frac{-j2\pi f i}{k}}, f = 1, \ldots, k \text{ and } j = \sqrt{-1}$$

- The vast majority of coefficients are Required when working with climate datasets, which are recalcitrant ts.
- Limitations on query length and start and end points apply when subdividing time-series into fundamental windows.



TSUBASA ARCHITECTURE

- **Sketch time:** dividing every time-series into basic windows; computing and storing statistics
- **Query time:** all-pair correlations are calculated using the sketched statistics, without the need to access the raw data.



- **Matrix Update:** constructing the initial matrix; ingesting the real-time raw data in chunks and sketching on the fly; and incrementally updating the matrix

PAIRWISE CORRELATION AND INCREMENTAL CALCULATION

LEMMA 1. *Given query window $x = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ and $y = [\mathbf{y}_1, \ldots, \mathbf{y}_m]$ and the sizes of basic windows $\mathbf{B} = [B_1, B_2, \ldots, B_{n_s}]$, where B_i is the size of the i-th basic window, and $m = \sum_{i=1}^{n_s} B_i$. The exact Pearson's correlation of x and y is:*

$$Corr(x, y) = \frac{\sum_{j=1}^{n_s} B_j(\sigma_{x_j}\sigma_{y_j}c_j + \delta_{x_j}\delta_{y_j})}{\sqrt{\sum_{i=1}^{n_s} B_i({\sigma_{x_i}}^2 + \delta_{x_i}^2)}\sqrt{\sum_{i=1}^{n_s} B_i({\sigma_{y_i}}^2 + \delta_{y_i}^2)}}$$

$$\delta_{x_i} = \overline{x_i} - \frac{\sum_{k=1}^{n_s}\overline{x_k}}{n_s}, \quad \delta_{y_i} = \overline{y_i} - \frac{\sum_{k=1}^{n_s}\overline{y_k}}{n_s}$$

where, σ_{x_i} (σ_{y_i}) is the standard deviation of basic window of x_i (y_i), c_i is the correlation of basic windows x_i and y_i, $\overline{x_i}$ ($\overline{y_i}$) is the mean of basic window x_i (y_i).

- Exact pairwise correlation on arbitrary query.

$$Corr_{t+B_{n_s+1}}(x, y) = \frac{1}{C\,.\,D}\Big(T\sigma_x\sigma_y Corr_t(x, y) + B_{n_s+1}(\sigma_{x_{n_s+1}}\sigma_{y_{n_s+1}}c_{n_s+1} + \delta_{x_{n_s+1}}\delta_{y_{n_s+1}}) - B_1(\sigma_{x_1}\sigma_{y_1}c_1 + \delta_{x_1}\delta_{y_1}) - T'\alpha_x\alpha_y\Big)$$

$$C = \sqrt{T\sigma_x^2 + B_{n_s+1}({\sigma_{x_{n_s+1}}}^2 + \delta_{x_{n_s+1}}^2) - B_1(\sigma_{x_1}^2 + \delta_{x_1}^2) - T'\alpha_x^2}$$

$$D = \sqrt{T\sigma_y^2 + B_{n_s+1}({\sigma_{y_{n_s+1}}}^2 + \delta_{y_{n_s+1}}^2) - B_1(\sigma_{y_1}^2 + \delta_{y_1}^2) - T'\alpha_y^2}$$

$$\alpha_x = \frac{B_{x_{n_s+1}}\delta_{n_s+1} - B_1\delta_{x_1}}{T} \quad \text{and} \quad \alpha_y = \frac{B_{n_s+1}\delta_{y_{n_s+1}} - B_1\delta_{y_1}}{T}$$

$$\delta_{x_{n_s+1}} = \overline{x_{n_s+1}} - \overline{x_{1:n_s}} \quad \text{and} \quad \delta_{y_{n_s+1}} = \overline{y_{n_s+1}} - \overline{y_{1:n_s}}$$

- Incremental correlation calculation.

COMPLEXITY ANALYSIS

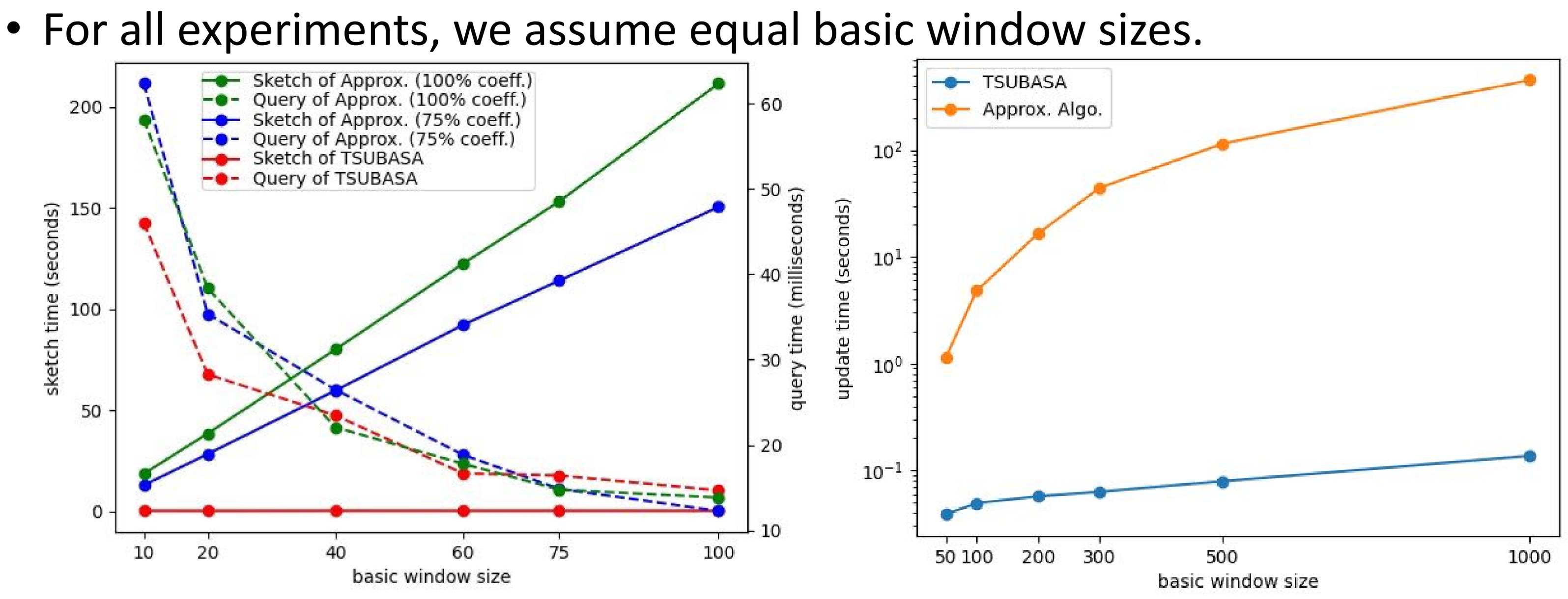
- Suppose N time-series of each time-series is in length L the basic window size of B.
- **Space complexity of TSUBASA:** $O(L\,.\,N^2/B)$
- **Space complexity of DFT-based approximation:** $O(L\,.\,N^2/B)$
- **Sketch time complexity of TSUBASA:** $O(L\,.\,N^2)$
- **Sketch time complexity of DFT-based approximation:** $O(L^2\,.\,N^2)$
- **Query time complexity of TSUBASA:** $O(N^2\,.\,L/B)$
- **Query time complexity of DFT-based approximation:** $O(N^2\,.\,L/B)$

CHALLNGES

- Exact calculation of the complete correlation matrix
- Correlation calculation on queries of arbitrary size with arbitrary start and end points
- Efficient network construction and update for historic and real-time data to achieve interactivity

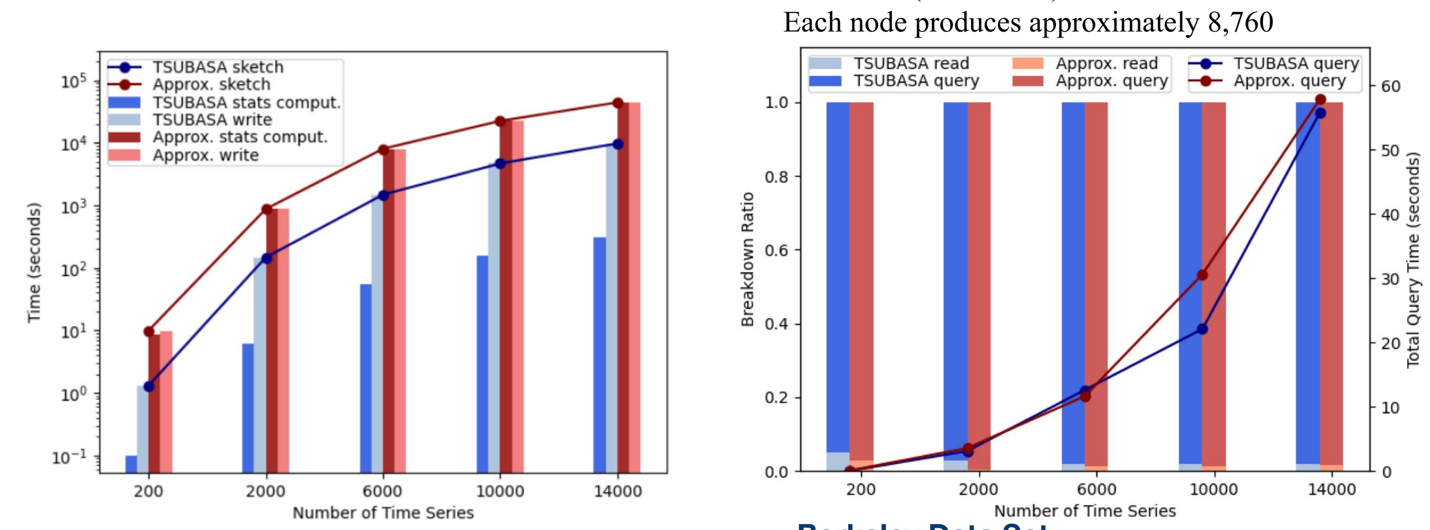
INCREMENTAL CORRELATION CALCULATION

- All algorithms are implemented using Go language. We use PostgreSQL for storing data sketches and experiments are conducted on a machine with 2 Intel ® Xeon Gold 5218 @ 2.30GHz (64 cores), 512 GB DDR4 memory, a Samsung ® SSD 983 DCT M.2 (2 TB).
- For all experiments, we assume equal basic window sizes.



In-memory TSUBASA

NCEA Data Set :
157 nodes (time-series) across the US.
Each node produces approximately 8,760

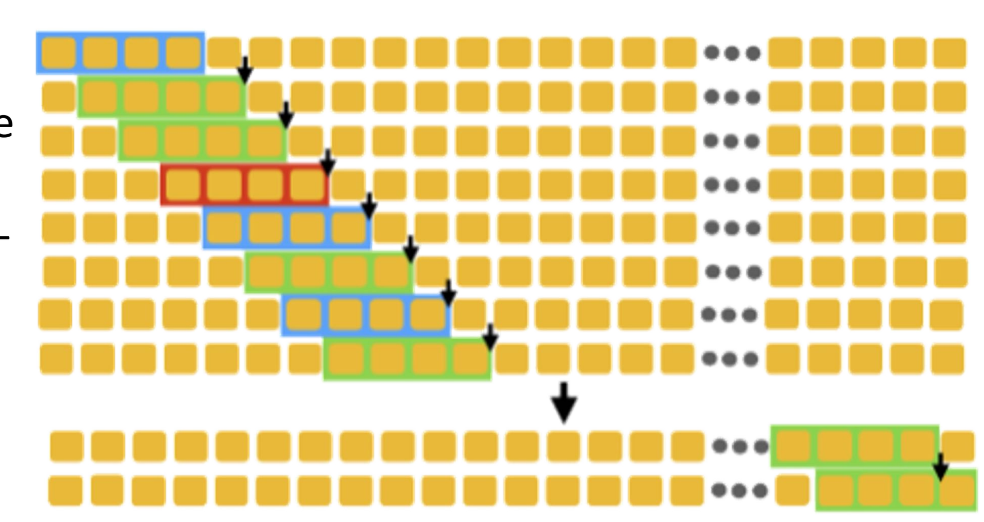


Disk-based TSUBASA

Berkeley Data Set:
18,638 nodes across the globe.
Each node has length 3,652.

SOLUTION TO SLIDING WINDOW: PREDICTION AND JUMP

The sliding window query is a common use case; for example, in neuroscience, we use a sliding window to compute the dynamic functional connectivity. We proposed an algorithm to avoid step-by-step correlation computation by predicting the future correlation based on its upper bound and lower bound.



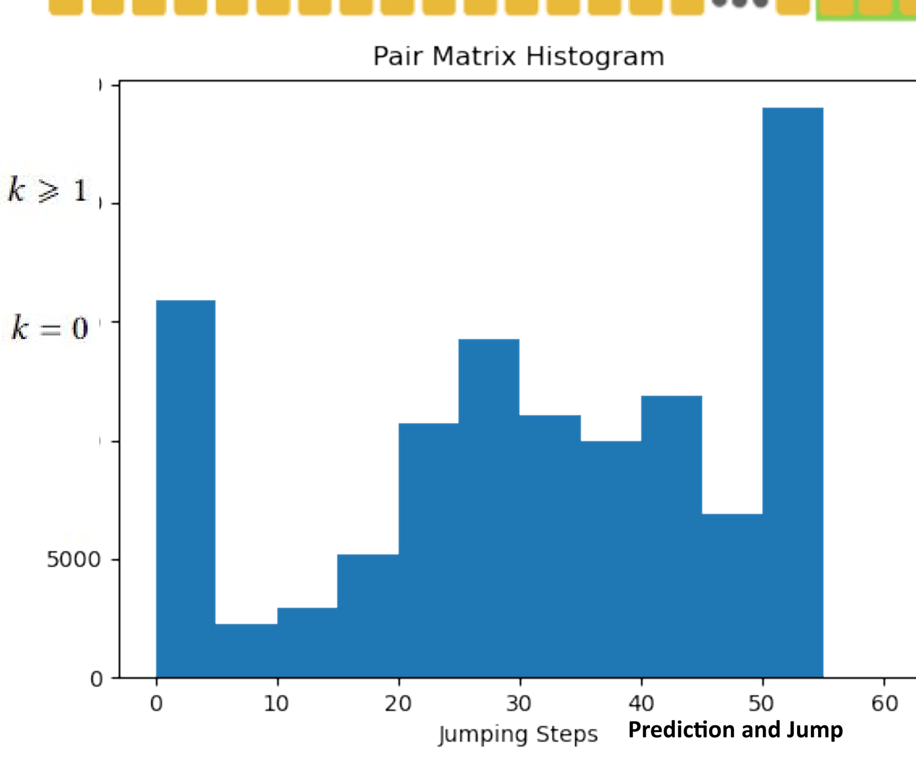
(1) $\sigma_{X_i} = \sigma_X$ ($\sigma_{Y_i} = \sigma_Y$), $1 \leqslant i \leqslant H \times n_s$
(2) $\overline{X_i} = \overline{X}$ ($\overline{Y_i} = \overline{Y}$), $1 \leqslant i \leqslant H \times n_s$

Here we define β_i^k for $C_{XY_{i:i+n_s-1}}$:

$$\beta_i^k = \begin{cases} C_{XY_{i:i+n_s-1}} + \frac{1}{n_s}\left(k - \sum_{j=1}^{k} C_{XY_{i+j-1}}\right) & k \geqslant 1 \\ \frac{\sum_{j=0}^{n_s-1} C_{XY_{i+j}}}{n_s} & k = 0 \end{cases}$$

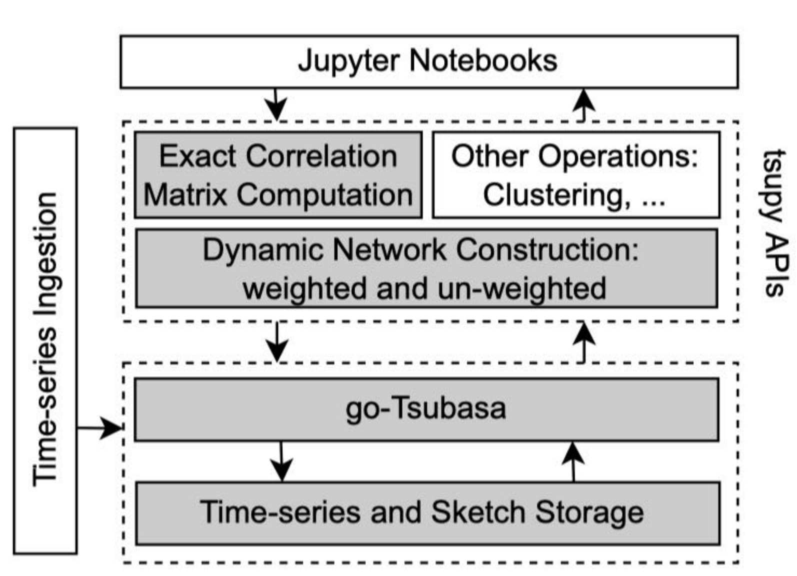
Take $C_{XY_{1:n_s}}$ as an example, we notice that:
(1): $\beta_1^0 = C_{XY_{1:n_s}}$
(2): β_1^k is a upper bound for $C_{XY_{k:n_s+k}}$

NCEA Data Set :
How many steps could we jump?
On average, 30.5376.



TSUPY: Dynamic Network Analysis Library

TSUPY is a Python library, which extends Jupyter Notebook as instrumentation for performing network construction and analysis at interactive speed. This demonstration focuses on how TSUPY enables dynamic network analysis on climate data.
https://github.com/DataIntelligenceCrew/tsupy



REFERENCES AND ACKNOWLEDGEMENT

https://github.com/DataIntelligenceCrew/tsubasa

This work was partially supported by the Goergen Institute for Data Science at the University of Rochester.

[1] S. Abe and N. Suzuki, "Scale-free network of earthquakes," EPL (Europhysics Letters) 65(4), p. 581, 2004.
[2] K. Kim, H. Joo, D. Han, S. Kim, T. Lee, and H. S. Kim, "On complex network construction of rain gauge stations considering nonlinearity of observed daily rainfall data," Water 11(8), p. 1578, 2019.
[3] A. Gozolchiani, K. Yamasaki, O. Gazit, and S. Havlin, "Pattern of climate network blinking links follows el nĩno events," EPL (Europhysics Letters) 83(2), p. 28005, 2008.
[4] Y. Zhu and D. E. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time," in VLDB, pp. 358–369, 2002
[5] A. Mueen, S. Nath, and J. Liu, "Fast approximate correlation for massive time-series data," in SIGMOD, pp. 171–182, 2010.