THESIS

A FRAMEWORK TO SUPPORT DISTRIBUTIONAL SIMILARITY ANALYSIS OVER ARBITRARY SPATIOTEMPORAL SCOPES AT SCALE

Submitted by

Paige Hansen

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2025

Master's Committee:

Advisor: Shrideep Pallickara
Co-Advisor: Sangmi Lee Pallickara

Mazdak Arabi

ABSTRACT

A FRAMEWORK TO SUPPORT DISTRIBUTIONAL SIMILARITY ANALYSIS OVER ARBITRARY SPATIOTEMPORAL SCOPES AT SCALE

As data volumes have grown, they present opportunities to extract insights that were previously inaccessible. Distributional similarity analysis across time and space plays a central role in identifying evolving trends and informing both model calibration and decision making. We present our methodology designed to support such longitudinal distributional analysis at scale over spatiotemporal datasets i.e., datasets comprising records that have both spatial and temporal dimensions associated with them.

Our methodology leverages a mix of statistical, algorithmic, and systems techniques to enable efficient and memory-resident similarity analysis. Rather than relying on a fixed metric, similarity thresholds are derived from the characteristics of each variable, allowing the measure to remain sensitive to intra-dataset variation. We employ the Jensen–Shannon divergence for its symmetry and boundedness and we also summarize probability density functions as compact 4-tuples that allow navigation through extents based on their degree of similarity. A refinement of this representation further allows differential scaling across dimensions to extend the scope of analysis.

The contributions of this thesis are threefold. First, we compute variable-specific thresholds that adapt similarity scoring to the distributional features of the data. Second, we introduce a novel distance-based measure that prunes the search space without compromising accuracy. Third, we demonstrate the ability to perform distributional analyses, both comprehensive and interactive, across arbitrary spatiotemporal scopes, with near real-time calculation of thresholds and similarity estimates. Our empirical benchmarks, with multivariate datasets spanning 50 years of complex, evolving climate phe-

nomena, validate these design choices and underscore the suitability of the methodology for large-scale, longitudinal datasets. Our methodology results in three orders of magnitude speedup over Apache Druid, which is a leading framework for distributional analysis at scale.

ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation (1931363,2312319), the National Institute of Food Agriculture (COL014021223, 2025-77039-45531), and an NSF/NIFA Artificial Intelligence Institutes AI-LEAF Award [2023-03616].

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The past couple of decades has seen exponential growth in the availability of data in several domains. These data continue to be available at ever increasing precision, resolution, and frequencies. A lot of this growth is driven by the proliferation of sensed data (both in situ and remote), simulations/scientific models, and applications. These data represent sensed or modeled phenomena. As such these data encapsulate a wealth of information that can be used to extract insights.

The data we consider are *spatiotemporal*. Individual data items are multivariate and are *geocoded* with coordinates representing the location or spatial extent for which the observations hold. The temporal dimension encapsulates the chronological timestamp representing when the observations were collected. Such datasets play a critical role in understanding spatiotemporally evolving phenomena such as epidemiology, environmental and ecological surveillance, climate change and adaptation, and atmospheric sciences among others.

The crux of this study is to support identification of distributional similarity at scale. In particular, the study targets looking at observational characteristics across diverse spatiotemporal scopes. The *spatiotemporal scope* comprises both a geographical extent and a temporal window. Geographical extents could be based on geohashes, quadtiles, and, in the most general case, demarcated using N-sided polygons where each vertex is represented using *<latitude, longitude>* coordinates. Distributional analyses are performed within the context of a spatiotemporal scope i.e. users specify both the extent type (e.g., administrative boundaries) and the temporal window (e.g., year, month).

Such future projective or retrospective analysis is useful for planning and decision making. In particular, this includes understanding impacts, mitigation and adaptation strategies, consequences, and potential hazards encountered at different spatial loca-

tions. For example, understanding the impacts of what happened in region **X** over a temporal window *w*, can inform decisions regarding how to cope with the same thing happening in region **Y** at a later time. Consider a future climate dataset representing the output of an ensemble of global circulation models under the 8.5 representative concentration pathways for greenhouse gas emissions. An urban planner might be interested in identifying locations in the future that will be like Phoenix in the year 2026. The planner may also be interested in the inverse, for example, in identifying locations that currently have similar temperature/humidity profiles as that of Tampa Bay in 2075.

Supporting such distributional analysis queries is difficult with traditional datastores. Datastores – be they relational, document/NoSQL, or hybrid – support queries that probe characteristics of elements within individual records/documents. These systems are not suited to support exploration of distributional characteristics across spatiotemporal scopes. Such distributional analysis queries can be useful in calibrating analysis and understanding behavior of models. A key goal of this study is to support declarative queries that abstract away details of the query specification; these include *where and when else*; *most similar*; and *k-most similar* for arbitrary spatiotemporal scopes.

1.1 Challenges

There are several challenges to enabling distributional similarity queries over multivariate datasets.

- The datasets we consider are voluminous and can thus entail significant disk and network I/O during processing. Since all processing is performed in shared clusters, the adverse impact on collocated processes can be significant.
- The data we consider can encompass vast spatial and temporal bounds. For example, in this study, we consider daily multivariate data for the continental United States at 4km resolutions for a 50-year duration.

- Spatial groupings of interest may be based on political, climatic, and/or terrestrial characteristics specified using shape files. Additionally, deterministic, hierarchical grouping based on geohashes or quadtiles can also be specified.
- Precomputing pairwise similarity scores across available (N) spatiotemporal scopes does not scale. Pairwise comparisons have a $O(N^2)$ complexity that grows quadratically as the problem size increases.
- The datasets we consider are non-stationary i.e., the means, variances, and covariances across variables within the dataset may all change over time.

1.2 Research Questions

The overarching goal of this study is to enable distributional similarity analysis at scale. Research questions that we explore as part of this study include:

RQ-1: *How can we preprocess datasets so that they are amenable to distributional analysis?* Such processing is a precursor to supporting distributional analysis at varied spatiotemporal scopes.
RQ-2: *How can we effectively identify spatiotemporal scopes that satisfy similarity queries?*
RQ-3: *How can we preserve timeliness for such queries at scale?* Given the data volumes and spatiotemporal scopes involved, ensuring timeliness as the scale increases is vital.
RQ-4: *How can we ensure that distributional analysis at arbitrary spatiotemporal scopes avoids extensive re-computations?* Repeated accesses to raw data is infeasible.

1.3 Approach Summary

Our methodology leverages a mix of statistical, algorithmic, and systems techniques to facilitate distributional similarity analysis at scale. Our framework, Archimedes [1], allows users to support distributional analysis over voluminous spatiotemporal datasets. Our methodology comprises (1) data collation, (2) selection of distributional similarity mea-

sures, (3) computing probability density functions (PDFs) for spatiotemporal scopes and encapsulating distributional information in a compact tuple set, (4) pruning the search space for queries with a novel distance-based measure, (5) leveraging a distance based data structure to ensure targeted search for similarity analysis, (6) a comprehensive search for more precise results, and (7) aggregating atomic extents to larger arbitrarily sized spatiotemporal scopes.

Profiling and contrasting distributions can be helpful in scrutinizing assumptions, formulating hypotheses, and framing analyses. To this end, we will construct an atomic spatiotemporal scope—the spatial extent and the temporal window—over which the distributional analysis would be performed and larger scopes are derived from. Our methodology spatiotemporally partitions the datasets to ensure effective data preprocessing. The data space is partitioned into spatiotemporal cubes, where the spatial dimension represents the geographical extent under consideration and the temporal window encapsulates the chronological bounds associated with the data of interest.

Once the data are collated, we compute PDFs associated with the variable(s) of interest for each spatiotemporal scope with respect to the chosen shapes and configured temporal step. We compute the mean, variance, skewness, and (excess) kurtosis values for each scope. Together, these four variables provide discriminatory power over a broad class of distributions.

We support two query variants for our distributional similarity analysis: a comprehensive query that is slow but finds all similar spatiotemporal scopes, and an interactive search that is probabilistic in the sense that it may miss some of the similar spatiotemporal scopes but can be an order of magnitude faster.

In our interactive search, we leverage distance in the vector space as a preliminary surrogate for similarity scores. In particular, we use distances in the vector space to prune the search space when we seek to identify spatiotemporal scopes with similar distributions. Given that each of the elements within the 4-tuples is unlikely to contribute equally

to distances that we compute, we design a novel scheme to weight each dimension differently. We leverage principal component analysis (PCA) to compute how each dimension is scaled differently. Scaling using our methodology allows us to identify/account for dimensions in the 4-tuple space that dominate distributional characteristics and variations for that particular variable.

We organize the spatiotemporal scopes, each with its own 4-tuple, in a k-d tree. Spatiotemporal scopes that are proximate to each other in the k-d tree have a high degree of similarity to each other. When querying for similar distributions, we first locate the *reference* spatiotemporal scope (e.g., maximum temperatures for Phoenix in 2026) within the k-d tree. Next, we search in proximity of this anchor spatiotemporal scope to retrieve proximate spatiotemporal scopes. Within this subset, we perform pairwise comparisons based on distributional similarity measures. We leverage the Jensen–Shannon divergence test (which, itself, is based on the Kullback-Leibler test) with the advantage that the measure is symmetric and finite. We perform these pairwise tests only for spatiotemporal scopes that are proximate, allowing us to prune the space considerably. We also leverage the k-d tree to support faster declarative queries.

We validate our ideas in the context of the voluminous Multivariate Adaptive Constructed Analogs (MACA) gridded dataset. Gridded datasets occur often in climate modeling, weather forecasts, atmospheric modeling, and other spatiotemporally evolving phenomena.

1.4 Paper Contributions

This study describes a methodology to facilitate distributional similarity analysis at scale. Our contributions include:

- Computation of variable-specific similarity thresholds for distributional analysis. This allows the similarity measure to be responsive to variations within a dataset.
- A novel distance-based measure to prune the search space for similarity analysis.

- Support for variable-specific scaling of distributional characteristics. Our distance measures account for scaling of individual dimensions in the tuple space.
- Support for streaming to ensure timely responses while ensuring that the most similar items are streamed first.
- Novel use of data structures, distance measures, and similarity scores to identify distributional similarity. Together, these allow us to prune the search space during query evaluations.
- The ability to perform distributional analysis (both comprehensive and interactive) for arbitrary spatiotemporal scopes. Crucially, our methodology enables near real-time calculation of distance thresholds and similarity estimates.

1.5 Paper Organization

The remainder of the paper is organized as follow. Chapter 2 outlines background and related work. Chapter 3 describes several key aspects of our methodology and system architecture. Chapter 4 includes a discussion of our performance benchmarks and profiling. Finally, Chapter 5 outlines our conclusions and future work.

Chapter 2

Related Works

Sketching algorithms are often used to evaluate queries issued over big data. The data sketches serve as a surrogate for the on-disk data and provide probabilistic responses to queries. The count-min sketch provides event frequencies using sublinear memory space; here, the event under consideration could either be a particular feature value or observation [2, 3]. The count-min sketch is closely related to Bloom filters, which employ hash functions over a fixed-size bit array to determine set membership. With Bloom filters, false positives are possible but false negatives are not [4]. Several sketching algorithms have been developed to determine the number of distinct (unique) elements in a multiset, such as HyperLogLog++ [5], HyperLogLog [6], LogLog [7], and Linear Counting [8].

Data may also be represented as wavelets to create high-fidelity approximations of underlying observations but require problem-specific tuning [9, 10]. Tao et al. [11] answers distinct sum and count queries over spatiotemporal data with a sketch index similar to an aRB-tree [12], where spatial indexing is implemented as an R-tree, and temporal indexing is implemented as a B-tree. This significantly reduces space requirements for answering distinct sum/count queries on spatiotemporal data. Data Cubes [13,14] provide multidimensional query and summarization functionality that are used in Online Analytical Processing (OLAP). By projecting two-dimensional relational tables to N-dimensional hypercubes, Data Cubes generalize several operators provided by relational databases. Since updates to Data Cubes are compute and data intensive, they are primarily intended for single-host offline or batch processing systems. Archimedes differs from these efforts in the following key aspects: (1) we target distributional analysis i.e. analysis in the aggregate rather than counts, cardinality, or set memberships, (2) our queries target similarity measures for distributional analysis, and (3) we support expressive query semantics in the

sense that any spatiotemporal scope can be presented as a reference scope to retrieve similarity matches.

Distributed analytics frameworks allow insights to be derived from data. Systems such as Horovod [15] also facilitate scalable, data-centric distributed orchestration of modeling workloads. Distributed data storage systems often have provisions for executing MapReduce-style computations [16, 17]. Extensions such as HaLoop [18, 19] and Twister [20] provide support for iterative MapReduce computations [21]. Spark [22] is a cluster computing framework that extends the MapReduce paradigm by supporting memory-resident datasets [23] and computations described by directed, acyclic graphs (DAGs). Our methodology preserves data locality during processing; as such, Archimedes data wrangling tasks can be expressed in the semantics required by the underlying analytics framework. The griddle effort [24] supports queries over gridded datasets by sliding a temporal window based on dyadic intervals; the crux in Archimedes is to support aggregated distributional analysis rather than point queries.

Sketching algorithms have be used to generate data *sketches* that provide compact, space-efficient surrogates for the data. These are particularly suited for unbounded data streams, and systems such as Synopsis [25], Pebbles Pebbles [26] , and Gossamer [27] capture distributional characteristics of the underlying data streams [28]; these data sketches can be inflated based on user-preferences. Our methodology does not preclude support for data sketches especially in situations where they can be inflated to generate synthetic, representative versions of the original datasets.

Finally, generative models based on Bayesian methods (e.g. Bayesian networks [29], Hidden Markov Models [30]) encode full probability distributions using a set of variables alongside their conditional dependencies. However, the crux of these methods is not so much distributional analysis (which is the focus of this study) but rather to facilitate powerful generation of synthetic, generative data that fit these distributions.

Chapter 3

Methodology

To ensure effective identification of distributional similarity at scale, our methodology includes a carefully calibrated mix of techniques encompassing data structures, algorithmic approaches for pruning the search space, and avoiding systems inefficiencies. In particular, these include: (1) effective data preprocessing which alleviates I/O shuffles and interference, (2) computation of probability density functions (PDFs) and key distributional characteristics, (3) identification of a distributional similarity measure, (4) scaling individual dimensions associated with the 4-tuples that we use to characterize distributions, (5) organizing spatiotemporal scopes while accounting for distances between them in the tuple space, (6) pruning the search space during similarity queries, (7) incorporating support for distance measures, and (8) extending atomic scopes to arbitrary scopes.

Dataset We validate our ideas in the context of a voluminous dataset that represents the outcomes of global climate circulation models under the RCP 8.5 (representative concentration pathway) based trajectory for greenhouse gas emissions. The data we use for our distributional analyses comes from the Multivariate Adaptive Constructed Analogs (MACA) Global Climate Model (GCM). The specific GCM is a gridded dataset composed of daily measurements for the continental United States, with an approximately four kilometer grid, or roughly 800,000 locations in total. We use the future climate data projections for the years 2026 to 2075, with multivariate data items available for each location for every day. In summary, our multivariate observations are available for 800,000 locations for over 18,000 days i.e. over 8 billion multivariate observations. Within our dataset, we focus our analyses on the following weather variables: maximum temperature, maximum relative humidity, precipitation, and specific humidity. We also derive a variable called respite, which is defined as the difference between the maximum and minimum temperatures, to capture the variation in temperature for a given day.

3.1 Building distributions for spatiotemporal scopes [RQ-1]

Our raw datasets need to be preprocessed so that they are amenable for analysis. In particular, we process the raw data through the pipeline shown in Figure 3.1 to create representations for comparing distributions and sums necessary to compute characteristics used for effectively moving through the search space.

To identify trends on meaningful spatial regions we map each point in our dataset to a larger extent. We use U.S. Census county shapes but any continuous, non-overlapping shapes, such as quadtiles, geohashes, or Census tracts could be substituted directly. We use counties because they are easily recognizable and can be mapped directly to other common datasets such as datasets produced by the U.S. Census Bureau. Because census shapes represent spatial extents that are variably sized, some counties own more individual points than others, but meet the requirements of being non-overlapping with no space in between.

Gridded datasets are typically packaged in the compressed binary netCDF format. However, most datastores cannot store netCDFs directly. Decompressing netCDF files is resource-intensive, so we developed a custom netCDF-to-CSV converter, producing smaller CSVs for collation. We map unformatted point data *<latitude, longitude>* to geospatial polygons by checking point inclusion in N-sided polygons. To minimize duplicate processing, we use Bloom Filters: space-efficient, probabilistic data structures that prevent false negatives but allow occasional false positives [4].

In the precomputation phase, we extract dataset points, associate them with their *<latitude, longitude>*, and perform spatial queries to map points to regions. Bloom Filters for each region are created and stored in-memory.

During lookup, a data point's spatial precision is reduced, and Bloom Filters are queried to identify possible regions. We refine this with full-precision lookups, confirming matches in regions. This approach ensures accurate associations while reducing processing costs. The final collated CSV size is 1.1 TB.

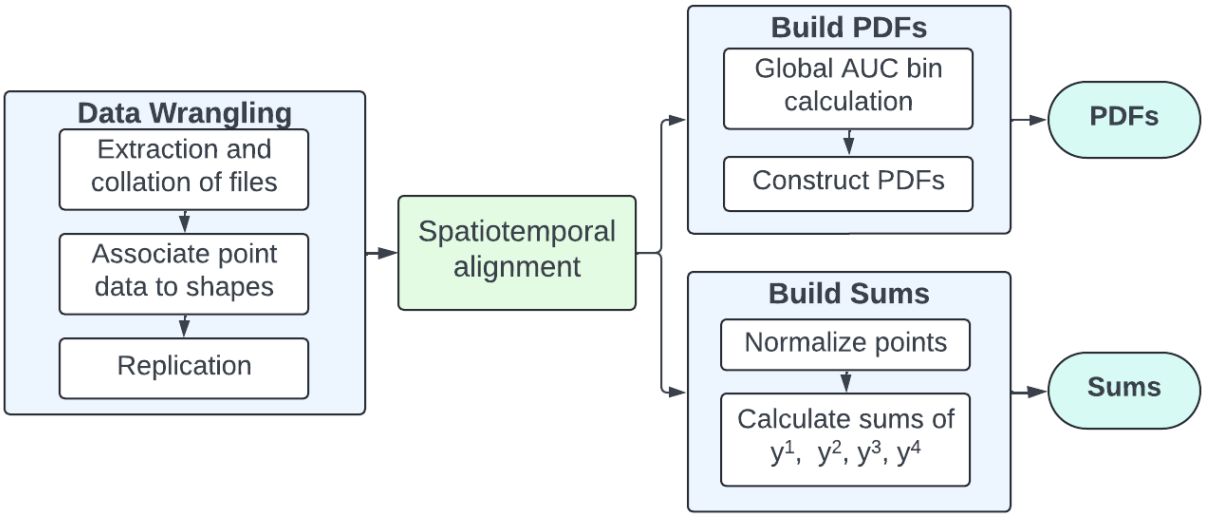

Figure 3.1: Preprocessing point data to (a) create probability density functions (PDFs) for spatiotemporal scopes and (b) generate sums of values used for deriving their distributional characteristics as 4-tuples.

Once spatial mapping is complete, we build our distributions, in the form of probability density functions (PDFs), for each county and every year in our dataset. We create one year temporal windows to avoid arbitrarily breaking up the natural seasonal cycle exhibited therein. Thus, using county by year, we have 155,200 spatiotemporal extents (as there are about 3,000 counties in the U.S.). To support comparisons between PDFs, we determine the bin boundaries of the buckets used in building the PDFs globally for each variable and use those to build the individual county-year PDFs. We construct the buckets to have approximately equal numbers of points by leveraging the area under the curve (AUC) calculation. We use two Apache Spark applications to determine the global bin boundaries and build the PDFs, and to calculate the sums of values for each of our five variables.

3.2 Computing distributional characteristics [RQ-1]

To quickly discriminate distributions that are not similar, we compute four empirical moments of the data for each county-year spatiotemporal scope: mean, variance, skewness, and kurtosis. While higher-order moments could be computed, the first four are

standard summaries that provide a precise summary of the distributional structure (historically, they were used in distributional fitting beginning with [31]) and they are readily interpreted. The mean can be interpreted as a central point or a typical value for the distribution, while variance illustrates how the data is spread about the mean. Skewness shows how asymmetrical the distribution is, while kurtosis is a measure of how heavy the tails are.

We take all values of the variable, across all county-years, and find the global minimum and maximum values. We use these values to normalize the variable to [0,1] by subtracting off the minimum and dividing by the range (maximum – minimum) shown in Figure 3.1. For every county-year spatiotemporal scope, we use the sums of values (covered in depth in §3.7) to compute the mean, variance, skewness, and kurtosis.

The distance between the distributional characteristics of our spatiotemporal scopes will serve as a shortcut to finding *neighbor* distributions. As such, we want to avoid unintended bias towards skewness or kurtosis, which are not bounded between 0 and 1 like the mean and variance derived from the same points. For example, the range of the mean values for maximum temperature was [0.4,0.8] while the range of kurtosis was [-1,5]. Therefore, we normalize each characteristic in the 4-tuples so that all have the same range of [0,1].

3.3 Similarity measures & similarity thresholds [RQ-2]

To measure the similarity of two distributions, we employ the Jensen-Shannon divergence (J-S) [32] over the Kullback-Leibler divergence (K-L) [33], Kolmogorov-Smirnov test (K-S) [34], or Earth Mover's Distance [35] because it is bounded, symmetrical and robust. The K-L score is asymmetrical and can yield infinite values when the distributions have non-overlapping support. K-S captures the maximum deviation rather than the overall similarity that the J-S effectively measures. Finally, the Earth Mover's distance is a physi-

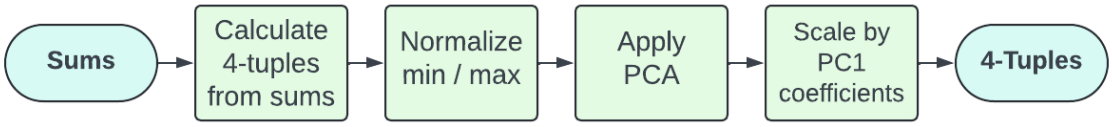

Figure 3.2: Scaling distributional characteristics (4-tuples) to facilitate reductions in the search space.

cal interpretation of the "work" needed to transform one distribution to another but can be overly sensitive to small discrepancies.

To compare distributions, we define a threshold to indicate similarity in the context of each of our variables. We assume that, in general, a distribution for a given spatiotemporal extent, a county-year, will be similar to the same county in the following year. Using this assumption, we calculate the J-S value for consecutive years for all counties. We calculate the upper bound for each variable independently. To allow for anomalies, we use the mean of these values as our upper bound similarity threshold. In domains where consecutive temporal windows cannot be assumed to be generally similar, domain specific knowledge could be leveraged to produce example pairs of acceptably similar distributions.

3.4 Scaling aspects of distributional characteristics [RQ-2]

Our goal is to situate likely-to-be-similar extents close to each other in our 4-tuple space defined by the distributional characteristics' 4-tuple made up of mean, variance, skewness, and kurtosis. Towards that end, it follows that the importance of each to discriminate will be different. Additionally, every variable in our dataset will demand varying levels of importance be assigned to each characteristic.

To avoid an expensive empirical survey to scale the 4-tuples, we leverage Principal Component Analysis (PCA) [36]. PCA is a dimension reduction technique that represents datasets using orthogonal principal components that are linear combinations of the original variables. The first principal component (PC1) that is generated using PCA explains

Table 3.1: Explained variance of the first principal component (PC1) by variable.

Variable	Explained Variance
Max Temperature	0.75
Max Relative Humidity	0.83
Precipitation	0.56
Respite	0.60
Specific Humidity	0.78

the most variance in the dataset. Specifically, we utilize coefficients, also referred to as loadings or weights, from PC1 to transform the data. Each dimension is multiplied by its corresponding PC1 coefficient, effectively weighting the variables according to their contribution to the principal component that explains the most variance in the data. The actual explained variance of PC1 is shown in Table 3.1. This transformation reorients the data in a way that emphasizes the most significant patterns of variation, while preserving the original dimensionality of the dataset. This pipeline is depicted in Figure 3.2.

Using the same assumption that we used for determining a similarity threshold (§3.3), that is, a given location's distribution in one year will be similar to the same location in the following year, we determine the radius around a reference point that will be searched. We calculate the Euclidean distance in our scaled 4-dimensional space between each location's consecutive year pairs and use the average for our query radius.

3.5 Organizing spatiotemporal scopes [RQ-2]

In order to use distance as a surrogate for pruning the search space at scale, we need a data structure that can efficiently navigate our 4-dimensional space to find extents that are nearby, and thus have a stronger likelihood of having a similar distribution. We employ k-d trees because of their compact size. The structure can be memory resident, even at scale, and could be duplicated across multiple servers to increase throughput. Notably, proximity searches are built into the data structure. K-d trees, one per variable, are constructed from the normalized and scaled 4-tuples. A k-d tree splits the data at

each axis along the median, alternating axes at each level of the tree, allowing for efficient Euclidean distance searches.

Because the 4-tuples are not clustered, but rather exist along some continuous spectrum not entirely defined by our 4-tuple, a Ball Tree or k-means-based clustering would not perform well. In the case of k-means clustering, we were unable to obtain even a moderately good silhouette score, indicating that our 4-tuples cannot be well separated into distinct clusters. We experimented with using k-means to create clusters to prune the search space for maximum temperature with $k = \sqrt{N}$ [37], or $\sqrt{155200} = 394$. Searching within a cluster yielded only 3.2% of the the total truly similar distributions being resident in the cluster.

3.6 **Pruning search space using distances** [RQ-2, RQ-3]

In interactive search, we use distance measures as a preliminary surrogate to prune our search space. We leverage the k-d tree to do a coarse grain elimination of extents that are farther away in the 4-tuple space. Only the extents that are inside of our search radius undergo a fine grain analysis using the J-S test to determine whether they are truly similar. Using the search radius we determine with consecutive years (Section 3.4), we are able to greatly reduce the number of PDFs we apply the J-S to. Because this is done at query time, a user could increase the distance they want to search within, possibly increasing the number of total similar extents that fall within the radius, but the most similar (that is, those with the lowest J-S) are always found very close. Increasing the radius also introduces more extents to be searched which become less likely to be similar as the distance increases.

3.7 **Variably sized extents using atomic extents** [RQ-4]

By maintaining the sums of values of the atomic unit, Archimedes can support aggregating those units to larger scopes. For our future climate dataset, the atomic unit is a

county for a single year. We can support near real time multi-year aggregation of both the PDFs and the 4-tuples such that a sliding window query over county 5-year extents is possible by simply using the new extents in place of the atomic units. Though we ground this discussion in expanding the temporal scope, it is also possible to increase the spatial scope in the same way. For example, counties can be aggregated to perform similarity analysis at the state level, as counties are entirely contained within their respective states. The method to aggregate scopes is not dependent on the atomic units containing the same number of points.

The PDFs are aggregated using a weighted average of the contributing distributions based on the number of points mapped to the spatial extent and the number of days in year j, denoted n_j. The weighted average is necessary because of leap years, the extra day magnified by the number of points to a county because counties are irregular and variably shaped.

In order to derive the 4-tuples needed for interactive search for the new temporal extents, we maintain atomic unit sums $S_{jp} = \sum_{k=1}^{n_j} y_{jk}^p$ for each year j and power $p = 1,2,3,4$, where y_{jk} is a measurement on day $k = 1,2,\ldots,n_j$ of year j. We can then use these atomic unit sums instead of the individual measurements to compute the (mean, variance, skewness, kurtosis) 4-tuple for a window of J consecutive years. This 4-tuple in turn relies on the sample mean and the sums of deviations to the second, third, and fourth powers.

The **sample mean** for a window size of J is

$$\bar{y}_{\bullet\bullet} = \frac{1}{\sum_{j=1}^{J} n_j} \sum_{j=1}^{J} \sum_{k=1}^{n_j} y_{jk} = \frac{1}{\sum_{j=1}^{J} n_j} \sum_{j=1}^{J} S_{j1}.$$

Then, using the sample mean, n_j, S_{j1}, and S_{j2}, we can compute the **sum of squared deviations** needed for sample variance as

$$\sum_{j=1}^{J} \sum_{k=1}^{n_j} \left(y_{jk} - \bar{y}_{\bullet\bullet}\right)^2 = \sum_{j=1}^{J} S_{j2} - 2 \sum_{j=1}^{J} S_{j1} \bar{y}_{\bullet\bullet} + \sum_{j=1}^{J} n_j \bar{y}_{\bullet\bullet}^2.$$

Similarly, the **sum of cubed deviations** needed for skewness is computed as

$$\sum_{j=1}^{J}\sum_{k=1}^{n_j}\left(y_{jk}-\bar{y}_{\bullet\bullet}\right)^3$$

$$=\sum_{j=1}^{J}S_{j3}-3\sum_{j=1}^{J}S_{j2}\bar{y}_{\bullet\bullet}+3\sum_{j=1}^{J}S_{j1}\bar{y}_{\bullet\bullet}^2-\sum_{j=1}^{J}n_j\bar{y}_{\bullet\bullet}^3.$$

Finally, the **sum of deviations to the 4th power** needed for kurtosis is computed as

$$\sum_{j=1}^{J}\sum_{k=1}^{n_j}\left(y_{jk}-\bar{y}_{\bullet\bullet}\right)^4=\sum_{j=1}^{J}S_{j4}-4\sum_{j=1}^{J}S_{j3}\bar{y}_{\bullet\bullet}$$

$$+6\sum_{j=1}^{J}S_{j2}\bar{y}_{\bullet\bullet}^2-4\sum_{j=1}^{J}S_{j1}\bar{y}_{\bullet\bullet}^3+\sum_{j=1}^{J}n_j\bar{y}_{\bullet\bullet}^4.$$

Therefore, we can prepare for a sliding window query by aggregating our PDFs using a weighted average and by deriving our 4-tuples using the one-year sums S_{jp} instead of recomputing the mean, variance, skewness, and kurtosis from the measurements directly. We scale our new 4-tuples with their new PC1 coefficients, as in Figure 3.2. Though the time to build these aggregated extents is only near real time, the time to compute a specific window size is amortized over the time the window size is used and the computed PDFs and 4-tuples are space efficient and can be easily stored for future use.

When the distributions of single years are combined to build a larger temporal window, a side effect is that the resultant distribution is more generalized, which in turn will change the rules of discriminating between them. Therefore, after rebuilding the PDFs and 4-tuples in the context of the new temporal window, we need to redefine similar; how that similarity is represented in both the PDFs using the J-S and the 4-tuples using Euclidean distance. We will define the similarity threshold to be the average J-S of consecutive sliding windows. This means there will be overlap in the measurements represented in each of the temporal neighbor distributions, causing both the similarity threshold and

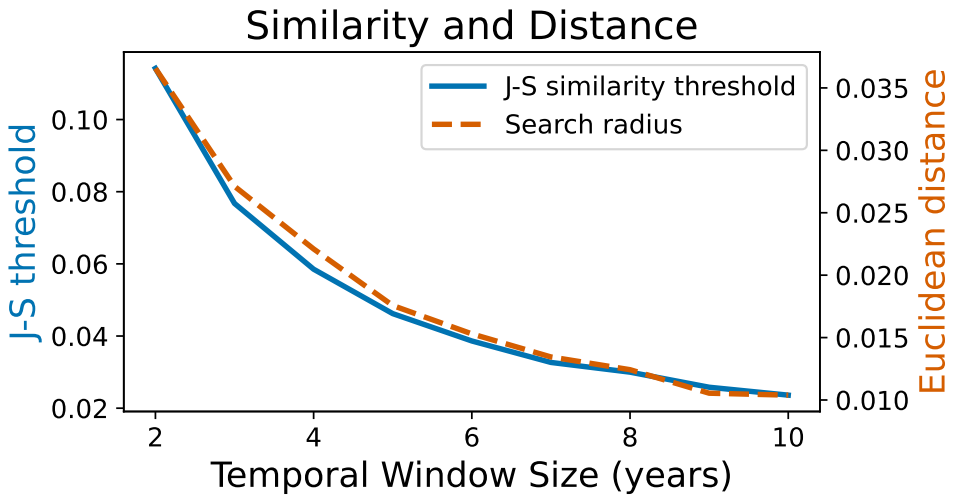

Figure 3.3: The relationship between the similarity threshold and search radius as the temporal window size increases. As the distributions become more general, the threshold for similarity is reduced.

search radius to be more strict in the context where our distributions are more general while remaining closely related, shown in Figure 3.3.

3.8 Support for declarative queries [RQ-3]

Our methodology supports several types of declarative queries over arbitrary spatiotemporal extents, including finding distributionally similar spatiotemporal extents across all variables and a top-k query.

Using interactive search, we can quickly find a set of spatiotemporal extents that are similar to the specified location and time. Shown in Figure 3.4, we query the k-d tree for neighbors that are inside of the search radius in the scaled 4-dimensional space. The Jensen-Shannon divergence is then computed for each of these neighbors. Using the similarity threshold, we determine if the neighbors are similar. At query time a user could, depending on their use case, tighten or relax both the similarity threshold and the search radius.

Because the most similar extents are found nearest to the inquiry point and we are searching the neighbors in order of nearness, we are able to begin streaming results

while more similar extents are found and are likely to find the most similar early in the search.

The similarity query can be extended to find extents which are similar in more than one variable by intersecting each set of results. Because the k-d tree is space-efficient, each variable's tree could be queried in parallel, even on only one machine.

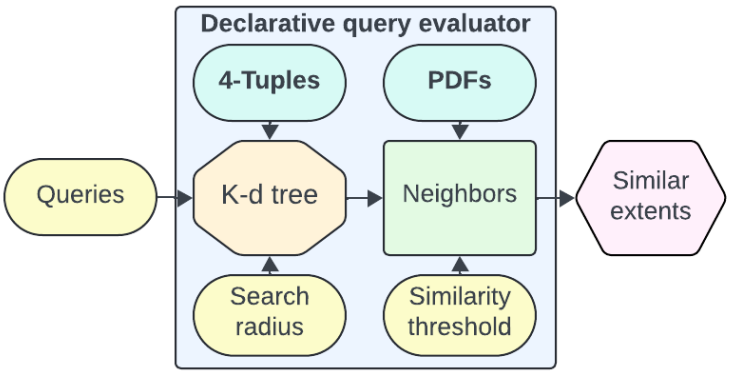

Figure 3.4: Interactive search pipeline

Another variation on the similarity query is a top-k query. The same process of querying the k-d tree for neighbors within a radius is performed, but only the most similar extents found are retained. Except in the case where N is exceptionally large, the results from a top-k query done with interactive search are identical to those from a comprehensive approach (which requires looking at every extent).

Chapter 4

Benchmarks

Our benchmarks profile the following aspects of our methodology:

1. How effective are our methods in constructing distributions over specified spatiotemporal scopes?

2. How fast are our methods in computing variable-specific similarity thresholds? Recall we identify per-variable similarity thresholds based on the data characteristics.

3. How effective is our approach in scaling individual dimensions in the 4-tuple space? Do they provide discriminatory power in pruning the search space?

4. How effective are distance measures as a surrogate for similarity to prune the search space?

5. Are we able to preserve interactivity in our queries? What are the trade-offs involved?

6. How effectively can we construct variable sized temporal windows during distributional analysis?

7. How does Archimedes' performance compare with a powerful, open-source tool (i.e. Apache Druid)?

To ensure representative results, we report our performance over multiple test points. The set of test points per variable are probabilistically distant from each other, and are therefore both random and dispersed in the 4-tuple space. This is important because in the search space, there will be more densely populated areas, causing fluctuations of work done.

4.1 Experimental Setup

We performed our experiments using HPE ProLiant DL60 Gen9 machines with the Intel Xeon 6-core, 2.40GHz CPU. The machines run AlmaLinux 8.9 with kernel version 4.18.0. The Spark cluster we use runs on 30 of these machines, with an average of 64 GB RAM per machine. The HDFS cluster uses 50 machines, with 2 PB of storage.

4.2 Time to construct distributions for spatio-temporal scopes

We use Apache Spark to construct both the sums of atomic extents needed for building the distributional characteristic 4-tuples and probability density functions (PDFs). The data is stored in HDFS partitioned by county to leverage data locality during computations. In order to build comparable PDFs, we first calculate the global bin boundaries for each variable. To compute bins for all 5 variables takes 31.6 minutes. Building PDFs for all spatiotemporal scopes for all 5 variables takes 119.5 minutes. Computing the sums of values for all variables and scopes takes 27.7 minutes.

4.3 Time to compute 4-tuples, similarity threshold, and distance radius

After the large precomputations of PDFs and sums, we compute the remaining necessary components to support our queries, reported as the average over 10 tests. To build the 4-tuples using the sums takes 0.86s. To normalize and scale the 4-tuples takes 0.38s. To derive the similarity threshold and search radius takes 7.00s and 1.52s respectively. In total, this processing step takes 9.77s.

4.4 Effectiveness of using PCA and scaling

We strengthen the correlation between J-S similarity and distance by reorienting the 4-tuples along their first principal component. We validate this using 20 of the random,

Table 4.1: Scaling with PC1 improves the correlation between similarity and distance.

	None	PC1	% Increase
Max Temperature	0.83	0.89	7.2
Max Relative Humidity	0.89	0.89	0.0
Precipitation	0.37	0.47	**27.0**
Respite	0.77	0.85	10.4
Specific Humidity	0.91	0.93	2.2

evenly dispersed test extents. Correlation, shown in Table 4.1, is calculated between the distance and J-S values for all extents, when compared to each test extent. The explained variance contributed by PC1 for each variable is shown in Table 3.1.

4.5 Effectiveness of using distance measures for pruning

By using distance between scaled 4-tuples as a surrogate for similarity, we are able to either reduce the number of extents we search through, increase the total number of similar extents we find, or increase the ratio of similar extents we find compared to the total number of extents within the search radius. These gains are summarized in Table 4.3.

We visualize the relationship between the distance and J-S in Figure 4.2 and show where our thresholds lay: We prune many extents based on distance at the cost of missing some extents with J-S values at the higher end of our acceptable similarity range. Though we miss those extents, we do find the *most* similar, as illustrated in Figure 4.1, which shows the percentage of the top N% results found by comprehensive search that were also found by interactive search. Generally, interactive search finds nearly all of the most similar extents. **In the top 5% of the true most similar extents, more than 87% are found by interactive search**. This was validated using 20 of the random, evenly dispersed test extents.

Table 4.2 shows a high level summary of the effectiveness of our optimizations to the 4-tuple space. The two variables, precipitation and respite, which saw the most dramatic

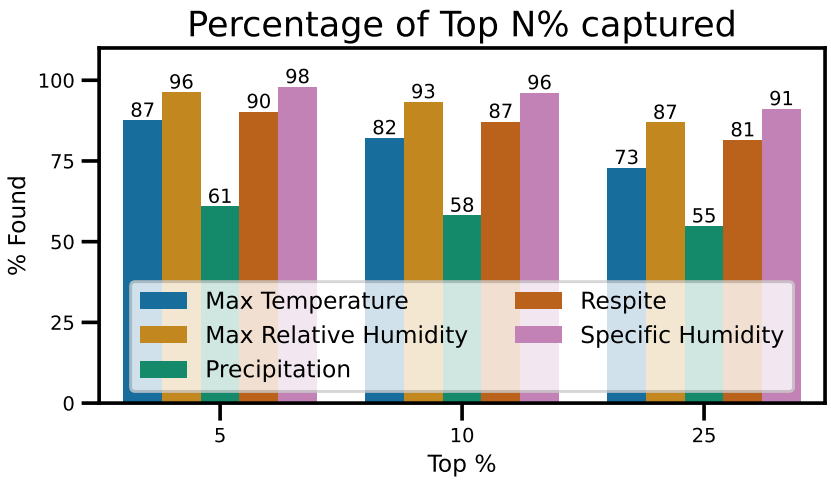

Figure 4.1: The average percentage of extents found by interactive search in the Top N% of the comprehensive search using 20 test points for each variable. Nearly all of the most similar extents are captured within the search radius.

Table 4.2: Summarized impacts of scaling the 4-tuple search space. With scaling, the search space becomes more efficient.

Variable	Net improvement of search space	Improved effectiveness of reducing work
Max Temperature	0.85%	7.0%
Max Relative Humidity	0.32%	0.0%
Precipitation	**4.50%**	7.9%
Respite	**5.64%**	**25.4%**
Specific Humidity	1.02%	5.3%

improvements were also the ones which had the lowest correlation between distance and similarity before any scaling.

In Table 4.3, the percent searched is the number of extents within the search radius. Percent found is calculated by counting the number of similar extents within the radius out of the total number of similar extents that exist. Hit rate is calculated as a ratio of the number of extents that were searched within the radius that were similar.

Precipitation trails all other variables with respect to nearly all measures. The correlation in Table 4.1 shows that precipitation has the lowest correlation between distance in the 4-tuple space and the J-S divergence. When scaled using the first principal component, it gains the most of any variable, 0.10. *The values in precipitation are extremely*

Table 4.3: Impacts on work done and search quality by scaling 4-tuples by PC1. All variables saw an improvement in either reducing the work done or the quality of the search.

Variable	Scale	% Searched	% Found	Hit Rate
Max Temperature	None	9.26	53.03	0.57
	PC1	**8.25**	50.11	**0.61**
Max Relative Humidity	None	19.82	87.85	0.26
	PC1	19.73	**88.51**	0.26
Precipitation	None	27.17	50.42	0.38
	PC1	**24.66**	49.77	**0.41**
Respite	None	13.25	71.32	0.59
	PC1	**10.65**	72.31	**0.74**
Specific Humidity	None	10.48	66.96	0.38
	PC1	10.58	**72.16**	0.40

skewed with over 60% of measurements being 0. Because so many values are the exact same, is is harder to discriminate between them using only the 4-tuple. These issues propagate throughout all of our benchmarks.

Max relative humidity also requires a high number of extents to be searched within the radius. *More than 60% of relative humidity's values are in the top 25% of the range.* This skewed distribution of values implies that many of the extents are similar to each other. This is supported by max relative humidity finding the highest number of the total matches within the proximity radius (Table 4.3).

4.6 Contrasting comprehensive and interactive search

4.6.1 Finding similar extents

Using 20 of our random, evenly dispersed test extents, we compare the time it takes to perform the J-S similarity test on every extent (comprehensive search) versus only looking within the search radius (interactive search). **Using interactive search, we are able to return a set of similar distributions up to an order of magnitude quicker than the comprehensive approach.** Table 4.4 shows that in the case of maximum temperature, we can gather results 11x faster than the searching the full set of extents.

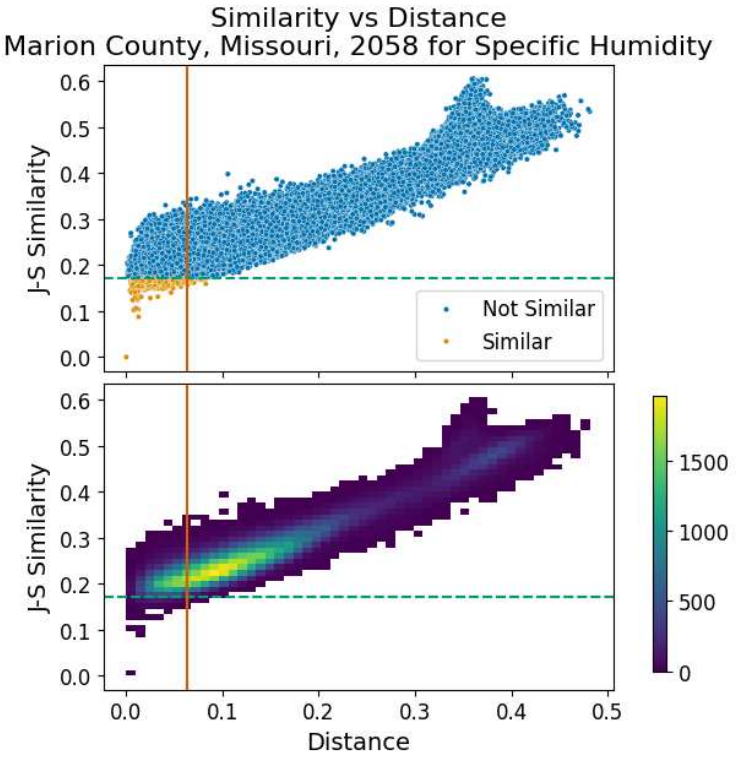

Figure 4.2: Visual aids for how the proximity relates to similarity for one extent and all other extents. The vertical line shows the search radius and the horizontal line shows the similarity upper threshold. (Top) Our goal is to maximize the similar (orange) points in the lower-left quadrant while minimizing points in the top-left and bottom-right quadrants. (Bottom) Areas in the 4-tuple space are densely packed, requiring more work when not pruned.

Precipitation and max relative humidity saw less of a speedup than the other variables. This is a direct consequence of the Percent Searched in Table 4.3, where the total percentage of searched extents was high, 24.66% and 19.73% respectively.

To find similar distributions in more than one variable, the similar extents for each variable can be found in parallel because they are entirely independent of one another. The time to find the intersection is the time of the longest running variable plus the time to intersect the results. The intersection can be optimized to begin with the shortest variable set.

A single J-S computation takes between 40 and 57 microseconds to compute (averaged over 3,104,000 tests). A single nearest neighbors k-d tree query takes between 32 and 40 milliseconds (averaged over 200 test points and 50 tests). Though the k-d tree

Table 4.4: Average time to find similar extents using comprehensive versus interactive search. Interactive search reduces the search time for every variable.

	Comprehensive (s)	Interactive (s)	Speedup
Max Temperature	7.74	0.70	**11.1**
Max Relative Humidity	7.85	1.60	4.9
Precipitation	6.96	1.87	3.7
Respite	7.71	0.92	**8.4**
Specific Humidity	7.37	0.81	**9.1**

query takes 3 orders of magnitude longer than the J-S test, it is only performed once per interactive query and reduces the number of J-S tests by thousands.

4.6.2 Top-k query

In order to find the top-k using comprehensive search, all samples must be evaluated. In our interactive approach, if the N is not excessively large, the **top-k most similar extents will be found within our proximity radius**. This makes interactive well suited to this type of query. Table 4.5 shows the speed and quality of the first similar extent found using either comprehensive or interactive search.

Additionally, in the context of streaming results, interactive search has the desirable property of identifying a more relevant *first* result up to 5.9x faster, as soon as 23ms. For example, the average J-S of the extent that is found first using comprehensive when searching maximum temperature is 0.20, but the upper bound threshold for similarity on that variable is 0.22. The first extent found using interactive search was 0.14. Both our approaches yield identical result sets when doing a complete top-k search, indicating that, for an k of 50, all 50 are found within the radius, shown in Table 4.6.

4.7 Building variably sized temporal windows

We build the extents needed for a sliding window query for window sizes of 2-10 for the maximum temperature variable, and present the times averaged over 10 tests. Because we have a set number of atomic temporal scopes, in our case 50 years, as the window

Table 4.5: Average time to find the first similar extent. The interactive search can find a similar extent faster and improves the quality of the extent found, shown by the lower average J-S score.

	Comprehensive		Interactive		Speedup
Variable	Time to first	Avg J-S	Time to first	Avg J-S	
Max Temperature	0.122s	0.20	0.036s	**0.14**	3.3
Max Relative Humidity	0.135s	0.15	0.034s	**0.12**	3.8
Precipitation	0.051s	0.10	0.038s	0.10	1.3
Respite	0.456s	0.12	0.077s	**0.08**	**5.9**
Specific Humidity	0.086s	0.16	0.023s	**0.12**	3.7

Table 4.6: Time to return the Top-k results for comprehensive vs. interactive queries, using a k of 50. Interactive search provides the same results as comprehensive several times faster.

	Comprehensive		Interactive		Speedup
Variable	Total time	Avg J-S	Total time	Avg J-S	
Max Temperature	7.86s	0.13	0.69s	0.13	**11.4**
Max Relative Humidity	7.26s	0.09	1.47s	0.09	4.9
Precipitation	6.14s	0.07	1.72s	0.07	3.6
Respite	7.60s	0.07	0.96s	0.07	**7.9**
Specific Humidity	7.15s	0.09	0.81s	0.09	**8.8**

size increases the number of extents in the range decreases. As illustrated in Figure 4.3, the overhead for creating a bigger window is offset by the reduced number of possible windows, slightly reducing the time to process the new larger extents.

4.8 Comparing to Apache Druid

Apache Druid is a distributed, column-oriented, real-time OLAP datastore optimized with roaring bitmaps for low-latency queries [38] [39]. We selected Druid because it supports querying for the distribution of a subset of points from a large dataset. The distributions are created using a data sketch of the data subset.

In order to support distributional similarity analysis, the distributions must, first, be accurate enough to discriminate between a similar and not similar distribution and, secondly, be timely enough to be useful. Using the Apache Datasketches Druid extension, we first create a data source containing precomputed sketches of every county-year for max temperature. The sketches are configurable in size, and we present our results using

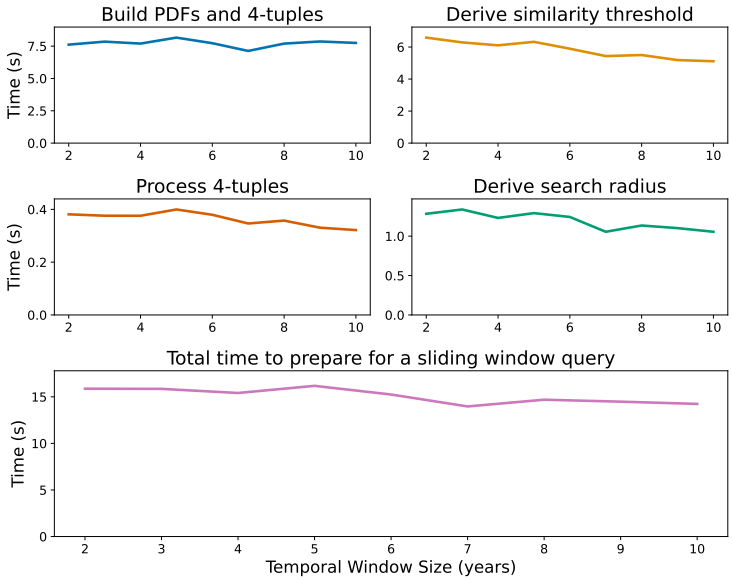

Figure 4.3: The time to build the aggregated PDFs, derive the 4-tuples, and determine new similarity threshold and search radius for various windows sizes. The added overhead for combining more years is offset by the reduced number of temporal extents being built.

Table 4.7: Column-normalized confusion matrix comparing Druid distribution estimations against true distributions. Druid incorrectly identifies many extents as being similar, diluting results.

	Druid's similar	Druid's not similar
Actual similar	20.82%	<0.01%
Actual not similar	**79.18%**	99.99%

the largest possible data sketch. We test similarity with 20 test reference extents against all other extents. Using Druid, the comparison task takes 26 minutes to complete per reference point, compared to Archimedes' 7.74 seconds.

Druid identified only 1–3 of over 15,000 truly similar results. Adjusting the similarity threshold to align with Druid's distributions increased J-S scores by 145% on average and resulted in a 99.99% detection rate but had a high false positive rate—4 in 5 matches were false positives (Table 4.7). Ultimately, Druid's distribution estimates lacked sufficient accuracy for reliable similarity discrimination, even with tailored thresholds.

Chapter 5

Conclusions

In this study we described our methodology to perform distributional similarity analysis at scale.

RQ-1: Staging datasets while accounting for spatial characteristics allows us to ensure data locality during calculation of the PDFs and the 4-tuples that we use to characterize them. Data from different spatial extents are located on different machines; this allows load balancing of the datasets. Because we leverage shape files to represent spatial extents, alongside inclusion or exclusion of data items based on geocoded point data, it is applicable to other spatial partitioning schemes such as those based on quadtiles and geohashes.

RQ-2: Rather than specify a fixed similarity measure per variable, our methodology bases similarity scores on the data characteristics. This allows the similarity score to have different thresholds for different variables. Using the Jensen-Shannon divergence provides two key benefits: symmetry and bounded measures. Summarizing the PDFs as a 4-tuple allows memory residency and underpins the ability to scale. A refinement of our methodology scales these dimensions differently. In particular, the use of PCA allows the scaling of each dimension to be based on their contribution to variability in the dataset. Scaling improves the effectiveness of the reduced search space by up to 25% (Table 4.3).

RQ-3: Using scaled distances in the 4-tuple space as a preliminary measure of similarity allows us to prune the search space for queries. In fact, as our benchmarks substantiate, the distance measure is strongly correlated with the Jensen-Shannon divergence (Table 4.1). Crucially, this allows us to reduce the number of pairwise comparisons that need to be performed (Table 4.3). The k-d tree organizes data based on distances. Organizing the 4-tuples using the k-d tree simplifies lookups based on distance measures. The pruned search space for queries reduces the number of pairwise similarity comparisons

that need to be performed ensuring timeliness in the evaluations, up to 11x faster (Table 4.4) while maintaining high quality results (Figure 4.1).

RQ-4: By computing the sums of values for an atomic extent, we ensure flexibility to aggregate both the distributions and 4-tuples to arbitrary scopes in near real time (Figure 4.3). Our methodology allows the spatial and/or temporal bounds associated with the spatiotemporal scopes to be configurable. Avoiding the costly need to return to the original data supports user exploration and analysis at scale without making assumptions about their use case.

In future work, we will explore combining distributional analysis with rapid visualizations and summarization using choropleth maps that are well suited for rendering spatial variation of data. This will allow users to track how these similarity scores evolve over time. Another avenue for our future work is to support measures of dissimilarity. For example, identification of the antipode for a spatiotemporal scope at scale based on user-specified measures.

Bibliography

[1] Paige Hansen, Nathan Orwick, Kassidy Barram, Pierce Smith, Jay Breidt, Sangmi Lee Pallickara, and Shrideep Pallickara. Archimedes: A framework to support distributional similarity analysis over arbitrary spatiotemporal scopes at scale. In *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 215–225, 2025.

[2] Graham Cormode. Count-Min Sketch. In LING LIU and M. TAMER ÖZSU, editors, *Encyclopedia of Database Systems*, pages 511–516. Springer US, Boston, MA, 2009.

[3] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[4] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970.

[5] Stefan Heule, Marc Nunkesser, and Alexander Hall. HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 683–692, Genoa Italy, March 2013. ACM.

[6] Philippe Flajolet, Eric Fusy, Olivier Gandouet, and Frederic Meunier. HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics*

& Theoretical Computer Science, DMTCS Proceedings vol. AH, 2007 Conference on Analysis of Algorithms (AofA 07)(Proceedings), January 2007.

[7] Marianne Durand and Philippe Flajolet. Loglog Counting of Large Cardinalities. In Gerhard Goos, Juris Hartmanis, Jan Van Leeuwen, Giuseppe Di Battista, and Uri Zwick, editors, *Algorithms - ESA 2003*, volume 2832, pages 605–617. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[8] Kyu-Young Whang, Brad T. Vander-Zanden, and Howard M. Taylor. A linear-time probabilistic counting algorithm for database applications. *ACM Trans. Database Syst.*, 15(2):208–229, June 1990.

[9] Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Found. Trends databases*, 4(1–3):1–294, 2011.

[10] Shahriar Yousefi, Ilona Weinreich, and Dominik Reinarz. Wavelet-based prediction of oil prices. *Chaos, Solitons & Fractals*, 25(2):265–275, 2005.

[11] Yufei Tao, George Kollios, Jeffrey Considine, Feifei Li, and Dimitris Papadias. Spatio-temporal aggregation using sketches. In *Proceedings. 20th International Conference on Data Engineering*, pages 214–225. IEEE, 2004.

[12] Dimitris Papadias, Yufei Tao, P. Kanis, and Jun Zhang. Indexing spatio-temporal data warehouses. In *Proceedings 18th international conference on data engineering*, pages 166–175. IEEE, 2002.

[13] J. Gray, A. Bosworth, A. Lyaman, and H. Pirahesh. Data cube: a relational aggregation operator generalizing GROUP-BY, CROSS-TAB, and SUB-TOTALS. In *Proceedings of the Twelfth International Conference on Data Engineering*, pages 152–159, February 1996.

[14] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing data cubes efficiently. *SIGMOD Record.*, 25(2):205–216, 1996.

[15] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow, 2018.

[16] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, jan 2008.

[17] Tom White. *Hadoop: The definitive guide*. O'Reilly, 2012.

[18] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. Haloop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1–2):285–296, Sep 2010.

[19] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D. Ernst. The haloop approach to large-scale iterative data analysis. *The VLDB Journal*, 21(2):169–190, Mar 2012.

[20] Jaliya Ekanayake, Hui Li, Bingjing Zhang, Thilina Gunarathne, Seung-Hee Bae, Judy Qiu, and Geoffrey Fox. Twister: a runtime for iterative mapreduce. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, Jun 2010.

[21] Haejoon Lee, Minseo Kang, Sun-Bum Youn, Jae-Gil Lee, and YongChul Kwon. An experimental comparison of iterative mapreduce frameworks. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, page 2089–2094, Oct 2016.

[22] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, page 10, USA, 2010. USENIX Association.

[23] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, page 2, USA, 2012. USENIX Association.

[24] Pierce Smith, Sangmi Lee Pallickara, and Shrideep Pallickara. Griddle: Effective query support over voluminous gridded spatial datasets. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 792–799. IEEE, 2022.

[25] Thilina Buddhika, Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara. Synopsis: A distributed sketch over voluminous spatiotemporal observational streams. *IEEE Transactions on Knowledge and Data Engineering*, 29(11):2552–2566, 2017.

[26] Thilina Buddhika, Sangmi Lee Pallickara, and Shrideep Pallickara. Pebbles: Leveraging sketches for processing voluminous, high velocity data streams. *IEEE Transactions on Parallel and Distributed Systems*, 32(8):2005–2020, 2021.

[27] Thilina Buddhika, Matthew Malensek, Shrideep Pallickara, and Sangmi Lee Pallickara. Living on the edge: Data transmission, storage, and analytics in continuous sensing environments. *ACM Transactions on Internet of Things*, 2(3):1–31, 2021.

[28] Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, and Harshawardhan Gadgil. Building messaging substrates for web and grid applications. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 363(1833):1757–1773, 2005.

[29] Irad Ben-Gal. *Bayesian Networks*. John Wiley & Sons, Ltd, 2008.

[30] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[31] Karl Pearson. X. contributions to the mathematical theory of evolution.-ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London.(A.)*, (186):343–414, 1895.

[32] María Luisa Menéndez, JA Pardo, L Pardo, and MC Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

[33] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[34] Yadolah Dodge. Kolmogorov–smirnov test. In *The Concise Encyclopedia of Statistics*, pages 283–287. Springer New York, New York, NY, 2008.

[35] Yossi Rubner and Carlo Tomasi. The earth mover's distance. In *Perceptual Metrics for Image Database Navigation*, pages 13–28. Springer US, Boston, MA, 2001.

[36] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[37] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Elsevier/Morgan Kaufmann, 2012.

[38] Fangjin Yang, Eric Tschetter, Xavier Léauté, Nelson Ray, Gian Merlino, and Deep Ganguli. Druid: a real-time analytical data store. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 157–168, New York, NY, USA, June 2014. Association for Computing Machinery.

[39] José Correia, Maribel Yasmina Santos, Carlos Costa, and Carina Andrade. Fast Online Analytical Processing for Big Data Warehousing. In *2018 International Conference on Intelligent Systems (IS)*, pages 435–442, September 2018.