Querying for Feature Extraction and Visualization in Climate Modeling

C. Ryan Johnson[1], Markus Glatter[1], Wesley Kendall[1], Jian Huang[1], and Forrest Hoffman[2]

[1] University of Tennessee, Knoxville TN 37919, USA
[2] Oak Ridge National Laboratory, Oak Ridge TN 37831-6016, USA

Abstract. The ultimate goal of data visualization is to clearly portray features relevant to the problem being studied. This goal can be realized only if users can effectively communicate to the visualization software what features are of interest. To this end, we describe in this paper two query languages used by scientists to locate and visually emphasize relevant data in both space and time. These languages offer descriptive feedback and interactive refinement of query parameters, which are essential in any framework supporting queries of arbitrary complexity. We apply these languages to extract features of interest from climate model results and describe how they support rapid feature extraction from large datasets.

1 Introduction

Given the high-resolution and high-dimensionality of the datasets resulting from today's large-scale climate simulations, it is typically infeasible to visualize a dataset in its entirety. Moreover, the limits of hardware and human perception hinder real-time investigative analysis. Possible solutions to reduce the amount of visualized data may involve automatically detecting statistically unique locations [1] or using a problem solving environment supporting manual filtering of the data [2]. An alternative solution offering a balance between automation and control is to visualize or emphasize only a relevant subset of a dataset satisfying some query or hypothesis chosen by the user. To be effective, such queries must be expressed in terms of the problem domain and offer rich feedback, enabling interactive refinement of query parameters.

Modern fully-coupled general circulation models (GCMs) are frequently used to generate climate projections hundreds of years into the future. As spatial resolution and temporal output frequency increase, to better resolve and understand complex interacting phenomena, model output grows considerably. Analyzing this output through traditional means is becoming impossible. As a result, various data mining techniques, designed to extract features of interest and simplify very large time series datasets, are increasingly being applied to the climate modeling domain. Previous work by Hoffman *et al.* [3] has demonstrated the utility of such techniques by applying k-means cluster analysis to hundreds of years

G. Allen et al. (Eds.): ICCS 2009, Part II, LNCS 5545, pp. 416–425, 2009.

of climate model results. This paper describes additional techniques that show promise in extracting regional and temporal features in an automated fashion.

Herein we describe two query languages for visualizing features in any large, time-variant datasets. The first was initially developed for volume visualization in a previous work [4] to enable users to express features in terms of statistical properties of local spatio-temporal neighborhoods, while the second [5] is a temporal pattern language modeled after *regular expressions*, a powerful tool more commonly used for locating patterns in text. Both languages are tightly integrated into the visual analysis process. We apply these language frameworks to a recent climate modeling simulation and describe a mechanism making query processing scalable.

2 Neighborhood Distribution Querying

Gaining insight into high-dimensional climate data often requires investigating and testing statistical hypotheses. For instance, scientists may be curious about the differences in the interannual variability of snow coverage between two decades, whether or not precipitation and temperature are positively correlated, or if mean solar radiation is decreasing through time. Investigating these hypotheses at a global scale reveals only overall trends. Generally, it is more informative to examine local regions independently and draw conclusions from observations of smaller-scale phenomena. To this end, we describe a framework that enables statistical querying and visualization of spatio-temporal data. We develop a visual query language in which scientists can express arbitrary, statistic-based queries to determine which local regions meet a set of hypotheses.

2.1 Neighborhood-Based Querying

The core structure of our framework is the distribution of values in the local neighborhood around each data point. Scientists query for features of interest in terms of statistics derived from intervals within these distributions. Figure 1 illustrates the process of forming a query.

Neighborhood. The exact size and shape of the local region that forms the sample space can be customized by the user. By default, the neighborhood is defined spatially as the set of data points within a specified Euclidean distance of radius r. However, users can alter the shape to search for features more easily described in an anisotropic sample space. Additionally, neighborhoods may be defined across both space and time domains.

Bin Selection. With the neighborhood defined, the user selects which variables are used to define the feature to be visualized. The user may focus on a particular interval of values for each variable. We refer to the interval on which a variable is examined as a *bin*. In choosing a bin, a scientist narrows the distribution for which statistical measures are calculated to a relevant subset of the data. For example, an investigation of liquid water may involve only temperatures above freezing.

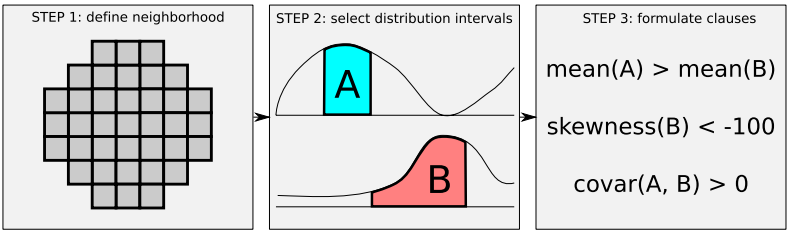


Fig. 1. An illustration of the statistical querying process. (a) Scientists first choose the spatio-temporal neighborhood serving as the sample space. (b) Next, the sample space is refined to intervals of the variables and their distributions relevant to the problem. (c) Lastly, scientists describe features of interest using inequalities relating the intervals' statistical primitives.

Querying by Predicate Clauses. With the sample space and distributions in place, queries are now expressed as a series of inequalities relating properties of distribution intervals to target criteria. The properties currently supported include relative frequency, mean, variance and standard deviation, skewness, and covariance. As an example predicate clause, let us consider the feature of rapid surface temperature change in the spatial domain. We select a circular neighborhood of radius 3, a bin encompassing all possible surface temperatures, and a predicate clause stating that the standard deviation must be greater than 3. We apply this query to the July 2000 timestep of the IPCC land-model simulation, with the result shown in Figure 2. Large elevational gradients are the primary feature emphasized by this query because these are the neighborhoods with large temperature deviations.

In the proceeding example, the target criterion is constant: standard deviation must be greater than 3.0 for a neighborhood to match. However, we can also allow target criteria to be expressed dynamically in terms of other interval properties. To query for locations where the mean percentage of a land grid square covered in snow one decade is half that of another decade, we select a neighborhood

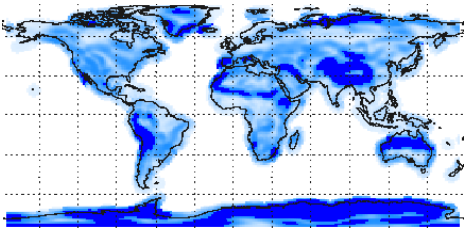


Fig. 2. A query for surface temperatures in circular neighborhoods of radius 3 in July 2000 having a standard deviation greater than 3°K. The dark locations fully match the query, while the lighter shade's opacity reflects how nearly a non-matching neighborhood came to matching.

spanning ten years, establish two intervals from 0% to 100% on snow coverage for each decade, and a clause requiring `mean(bin 1) < 0.5 * mean(bin 0)`.

Additionally, features typically involve compound criteria, and our query language readily handles multiple predicate clauses. Neighborhoods may match only a subset of the query's clauses, and in order to visualize partial matches, we record a *predicate signature*, which is a bitfield with bit i set to 1 if the criterion of clause i is fully met in a neighborhood. A neighborhood's predicate signature is used to determine its color when visualized. Each clause may also be assigned a weight reflecting its degree of importance in defining a feature.

To avoid a misleading boundary between matching and non-matching neighborhoods, each neighborhood is also assigned a score in $[0, 1]$ indicating how closely the criteria of the clauses are met. The score for an individual clause is assigned according to an exponential decay function of a neighborhood's distance from the target criterion and a dropoff parameter that controls how quickly the score drops with distance, as detailed in Equation (2). A neighborhood's total score is defined in Equation (3) as a weighted sum of the clause scores.

$$\Delta = \begin{cases} 0 & \text{if criterion is met} \\ |lhs - rhs| & \text{otherwise} \end{cases} \tag{1}$$

$$score_i = \exp\left(\frac{\Delta^2 \times \ln(.5)}{dropoff_i^2}\right) \tag{2}$$

$$score = \sum weight_i \times score_i \tag{3}$$

2.2 Query Visualization

A query is visualized by coloring each location with its neighborhood's predicate signature, which represents the combination of clauses that are fully met. Scientists can interactively customize the colormap indexed by these predicate signatures or make a particular combination completely transparent. The neighborhood's score is used to increase the neighborhood's transparency further. By default, low-scoring neighborhoods are assigned a transparency close to 0, while high-scoring neighborhoods are more opaque.

Figure 3 is an example visualization of a query with multiple clauses. The query investigates the relationship between mean temperature increase and mean precipitation between the first and last decades of the IPCC dataset. Each color represents a different combination of the three criteria being met, though only six of the eight possible interactions actually occur.

2.3 Implementation

To formulate distribution queries for spatio-temporal neighborhoods, we have built a graphical tool that guides the user in selecting neighborhood sizes, bin ranges, and criteria. Changes to query parameters instantaneously update the visualization for immediate and interactive feedback. Using the mouse, users can hover over each pixel and see exactly which clauses are fully met without having

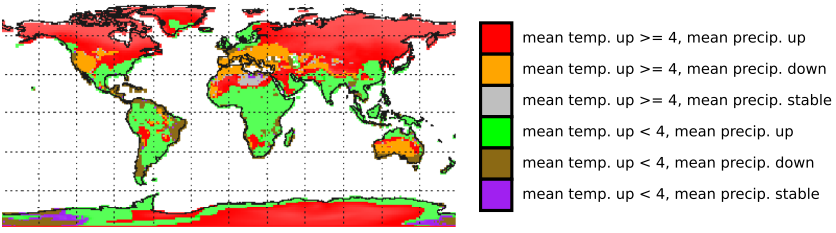


Fig. 3. An investigation of mean temperature and precipitation between decades 2000–2009 and 2090–2099. Six possible interactions are observed in the simulation: mean temperature may or may not go up 4°K or more, and mean precipitation may increase, decrease, or stay constant. Each possible combination of events yields a unique predicate signature and color.

to decipher the colormap. A separate pane allows each individual clause to be studied in isolation from the entire query. The visually-composed neighborhood, bin, and clause information can be saved in an XML format and reloaded and modified as needed.

```
for each neighborhood h
   for each neighbor n
      for each variable v to be binned
         if n.v in bin
            add n.v to bin
   calculate bin statistics
   for each clause c
      if c.criteria met
         c.score = 1
         set bit in h.predicate_signature
      else
         calculate c.score according to dropoff
      h.score = h.score + c.weight * c.score
```

Fig. 4. The algorithm for evaluating a neighborhood distribution query. The output of this algorithm is a set of predicate signatures and scores for each neighborhood in the dataset.

3 Temporal Querying

Using query-based visualization, users can sift through very large and highly complex multivariate data. Though queries are often guided by human-domain knowledge, query-based visualization commonly involves trial and error. Unfortunately, this approach typically does not scale as datasets grow exponentially large. In this work, we describe a query language for accelerating discovery of temporal connections between multivariate patterns of interest in climate modeling simulation data.

3.1 Textual Pattern Matching

Motivated by the elegance and power of regular expressions and globbing in text string searching, we have developed a text-based search language for concisely specifying temporal patterns. In our system, a user hypothesis can be loosely defined. In traditional regular expressions, wildcards are used to support inexact matching. For example, `file*.pdf` is an expression that matches any files named with prefix `file` and extension `pdf`. Our system accepts a similar kind of qualitative query. We use wildcards to provide a powerful way of representing the existence of temporal events.

To application scientists, this method of vaguely specifying temporal patterns to visualize is extremely useful. Domain knowledge is often expressed in a qualitative manner, and scientists can be hard-pressed to define exact data queries to extract meaningful subsets of the data. Using our temporal regular expression language, qualitative patterns containing wildcard characters can be entered and expanded to a set of discrete data queries that are extracted from the dataset. For a scientist, this offers a more natural way of entering qualitative domain knowledge and avoids a potentially lengthy search process guided only by trial and error.

3.2 Pattern Matching and Syntax

For querying, we employ range queries, a widely used method for selecting subsets of multivariate data. Even though range queries are usually discrete, our system accepts quantitative queries that match and support a user's unclear or imprecise understanding of data. Thus, we allow a user to issue "fuzzy queries" in order to explore a data set he is not highly familiar with. Range queries may contain wildcard characters such as `*` and `?` that are expanded to generate actual range queries, much like the UNIX function `glob()` expands wildcards in `file??.pdf` and regular expressions expand patterns such as `file.*\.pdf`.

However, our language has a few non-traditional elements. The first one is `T`, the temporal mark. A search of `[4]*T[5]*` means that we are looking for patterns where an attribute is valued at 4 for zero or more timesteps and then changes to value 5. The temporal mark is the time of this event's occurrence, and in this case it is chosen to be the instant of the first timestep of the value 5. Our parser extracts the `T` from the expression and then generates all discrete patterns of interest. The location of `T` is recorded so as to indicate the precise time of the event's occurrence in further visualizations or data explorations.

The following list of examples demonstrates accepted wildcards, special characters, and valid data ranges:

- `[-1e15 - 1025.7]` – data ranges for a single timestep. This range contains a from-value and a to-value. `[74.2]` has both values being identical (same as `[74.2 - 74.2]`).
- `[0.3 - 10e9]*` – a data range applied to zero or more sequential timesteps.
- `?`, `?*` – wildcard ranges. The first represents the entire range of data values for a single timestep. The second represents the entire range of data values for zero or more timesteps.

- `T` – the (optional) temporal mark. It marks the time index at which the event that is subject of the query occurs.
- `[1 - 5]*[8]` – a query without a temporal mark is also valid. This query addresses all items for which values in all timesteps are between 1 and 5, and a value of 8 in the last timestep.

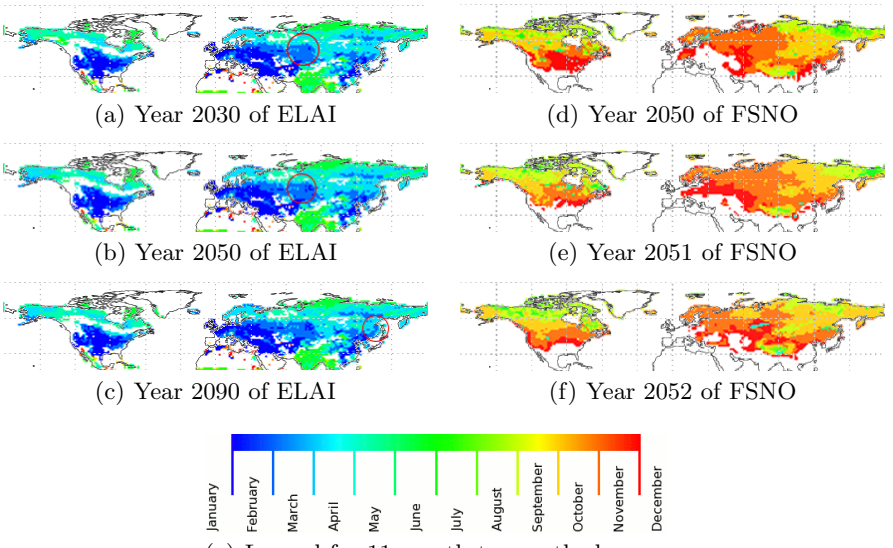
3.3 Querying Global Climate Model Results

Using the results from a global climate model, we establish "events" that are defined by one or more variables changing over time at different spatial locations. We mark the time of such events at each spatial location with the temporal mark `T`, and color individual pixels according to the extracted times. As we are considering temporal change between consecutive months, color indicates the point in time in between the months in which the event occurred, *e.g.*, given the colormap in Figure 5(g), a blue pixel depicts that the event happened in the transition from January to February.

In Figures 5(a) through 5(c), we display the temporal change of the variable ELAI (Exposed one-sided leaf area index in units of m^2 of leaf area per m^2 of ground area) over the course of one year. We query for data locations that experience a positive relative change of more than 40% after a period of zero or more timesteps in which the relative change is low (between −40% and +40%). The purpose of this query is to find the beginning of the growing season in the Northern Hemisphere. We mark the first timestep (month) in which the threshold of change is exceeded and color the pixels accordingly. We chose 40% as the threshold for spring green-up because smaller changes in leaf area index do not represent a temporal feature of interest.

We can immediately see how the event of marked temporal change in ELAI follows a certain path as the year progresses. As we expect, it begins in the southern parts of North and Central America and generally advances north month by month. On the Eurasian landmass, a similar progression to the north is visible, as is a progression from central Europe towards central Russia and Siberia. We also notice that the differences between years are minor and almost imperceptible. The temporal change of the variable ELAI over the course of each year appears to be very stable.

Figures 5(d)-(f) display the results of a query designed to identify the point in time when the first large snowfall between May and December occurs. The query then considers only data locations with snow cover (variable FSNO, representing the fraction of ground covered by snow) is either reduced or increased by no more than 7%. When we encounter the first temporal change of the snow fraction larger than 7%, we consider it the first large snowfall and mark its timestep (month). We chose 7% since it gives us a good threshold that makes our query resistant to minor changes of the snow fraction which would generate a false and potentially early temporal mark. Again, we can clearly make out the underlying pattern. The snow cover first grows larger by more than 7% in northern Canada and Siberia, as well as the Himalayas, and then progresses into the warmer regions to the south and, in the case of the Eurasian landmass, to the west. One can also recognize the Rocky Mountains in the Western U.S. as an area of early snowfall.



(a) Year 2030 of ELAI (d) Year 2050 of FSNO

(b) Year 2050 of ELAI (e) Year 2051 of FSNO

(c) Year 2090 of ELAI (f) Year 2052 of FSNO

(g) Legend for 11 month-to-month changes

Fig. 5. (a-c). A query for significant change in ELAI in the northern hemisphere. Locations are colored according to their temporal mark `T`, which indicates the time at which the change occurs. The colormap is shown in (g). Areas highlighted in red circles indicate locations where the temporal marker shifts most between decades. (d-f) A query for the time of the first large snowfall in the northern hemisphere.

3.4 Query Performance

Our system for evaluating queries is implemented using scalable data servers described in Section 4. All timing results have been collected using an AMD 2.6 GHz Opteron cluster connected by InfiniBand DDR 4X network and the climate data set from section 3.3. Measured performance metrics of our system are highly dependent on the data and the query. With most tests that we have run using 20 compute nodes as data servers, the query performance is acceptable for interactive use.

4 Scalable Querying

With the ever-increasing size of scientific simulation data, fast querying requires a scalable solution to managing and extracting features of the data. Previous methods have demonstrated scalable querying of large multivariate data [6], but I/O and preprocessing times have not been considered. However, for *in situ* querying of simulation data—in which the data is examined as it is created—these I/O and preprocessing times play a crucial role in the user experience. We describe a method of leveraging the power of modern supercomputing resources and parallel I/O to make this data-intensive query processing as fast as possible.

Table 1. Performance results on 20 compute servers. Running time is the wall clock time between query invocation and receipt of all matched locations.

Query	# Locations matched	Running time (secs)
`[0-10][0]?[0-1e10]*`	3	3.168
`[0-99][0]?[0-1e10]*`	3	26.522
`[40-60]??[0-1e10]*`	342	6.067
`[50]??*?*`	3,615,888	7.454
`[70]?[-5-1e10]*[5--1e10]*`	6,584	7.485
`[0-20]??[0-1e10]*?*`	3,593,696	159.97
`[80]?[-100-1e10]*[100--1e10]*[-100-1e10]*`	16,994,091	248.87
`[80-82]?[-100-1e10]*[100--1e10]*[-100-1e10]*`	50,565,859	757
`[0-99]?[-100-1e10]*[100--1e10]*[-100-1e10]*`	$\approx 1,685,529,000$	$\approx 72,500$

With the present ability to do calculations at one petaflop and beyond, many applications are bound by I/O speeds, and users of high performance systems realize that the memory access rate often determines the performance of their applications [7]. To reduce I/O bottlenecks, we employ existing parallel I/O libraries such as Parallel netCDF [8] for high-performance access to scientific datasets. Scientists create a configuration file of all the variables of interest from an arbitrary number of files. The configuration file is passed on to a routine that encapsulates the task of loading the data from disk using all available processors. This routine achieves maximum bandwidth by dividing the loading of data across processors and performing collective I/O when possible.

After being read from disk, the data must be distributed to keep the query processing load-balanced. In previous work [6], we have shown a method that offers near-optimal load-balancing among servers. Both query languages described above operate on value ranges of the data, and accordingly, processing a query can be accelerated by first sorting the data. The entire dataset is sorted in Hilbert space and distributed to the servers. To provide scalable sorting, we use an algorithm that keeps the data distributed among all the processors throughout the entire sort. This algorithm performs a global merge of the data and then swaps the data in a round-robin fashion. With the data quickly narrowed down to only relevant intervals, the rest of the query criteria can be evaluated in a distributed manner.

5 Conclusion

We have described two query languages for extracting features from very large scientific datasets, and have demonstrated their utility by applying them to extract and visualize features of interest from climate model output. Such tools are becoming increasingly important as simulations generate higher spatial and temporal resolution datasets that necessitate use of novel techniques for analysis. Neighborhood-based querying is useful for extracting spatial patterns from large datasets based on the statistical parameters of a region's frequency distribution. Temporal querying is useful for extracting patterns of change from very large

time-variant datasets. High performance I/O and parallel data processing are technologies that enable interactive query evaluation.

References

1. Jänicke, H., Wiebel, A., Scheuermann, G., Kollmann, W.: Multifield visualization using local statistical complexity. IEEE Transactions on Visualization and Computer Graphics 13, 1384–1391 (2007)
2. Kehrer, J., Ladstädter, F., Muigg, P., Doleisch, H., Steiner, A., Hauser, H.: Hypothesis generation in climate research with interactive visual data exploration. IEEE Transactions on Visualization and Computer Graphics 14(6), 1579–1586 (2008)
3. Hoffman, F.M., Hargrove, W.W., Erickson, D.J., Oglesby, R.J.: Using clustered climate regimes to analyze and compare predictions from fully coupled general circulation models. Earth Interactions 9(10), 1–27 (2005)
4. Johnson, C.R., Huang, J.: Distribution driven visualization of volume data. IEEE Transactions on Visualization and Computer Graphics (2009) (to appear)
5. Glatter, M., Huang, J., Ahern, S., Daniel, J., Lu, A.: Visualizing temporal patterns in large multivariate data using textual pattern matching. IEEE Transactions on Visualization and Computer Graphics 14(6), 1467–1474 (2008)
6. Glatter, M., Mollenhour, C., Huang, J., Gao, J.: Scalable data servers for large multivariate volume visualization. IEEE Transactions on Visualization and Computer Graphics 12(5), 1291–1298 (2006)
7. Ross, R., Peterka, T., Shen, H.-W., Ma, K.-L., Yu, H., Moreland, K.: Visualization and parallel I/O at extreme scale. Journal of Physics (Conference Series) 125 (July 2008)
8. Li, J., Liao, W.K., Choudhary, A., Ross, R., Thakur, R., Gropp, W., Latham, R., Siegel, A., Gallagher, B., Zingale, M.: Parallel netCDF: A high-performance scientific I/O interface. In: Proceedings of Supercomputing (November 2003)