# TECHNIQUES FOR EXPLORING CLUSTER COMPRESSED GEOSPATIAL-TEMPORAL SATELLITE DATASETS

By

John M. Ashley

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computational and Data Science

Committee:

| | |
|---|---|
| _____ | Dr. Daniel Carr, Dissertation Director |
| _____ | Dr. Kirk Borne, Committee Member |
| _____ | Dr. Amy Braverman, Committee Member |
| _____ | Dr. Jim Gentle, Committee Member |
| _____ | Dr. John Qu, Committee Member |
| _____ | Dr.Michael Summers, Department Chair |
| _____ | Dr.Timothy L. Born , Associate Dean for Academic and Student Affairs, College of Science |
| _____ | Dr. Peggy Agouris, Interim Dean, College of Science |

Date: _____

Fall Semester 2013
George Mason University
Fairfax, VA

# Techniques for Exploring Cluster Compressed Geospatial-Temporal Satellite Datasets

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

John M. Ashley
Master of Science
Michigan State University, 1991

Director: Daniel Carr, Professor
Department of Computational and Data Science

Fall Semester 2013
George Mason University
Fairfax, VA

## Dedication

To my long-suffering wife, Kathryn for all her support and understanding. Also to Mom and Dad; and my grandparents Alan, Alice, Robert, and Dorothy, who always believed.

**Acknowledgements**

There are no words to express my gratitude to Dr. Dan Carr, my advisor, who stuck with me when others would have let me drift. I'd like to acknowledge my GMU committee members, Drs. Kirk Borne, James Gentle, John Qu for pointed questions that cost me sleep but ultimately improved my research. I owe a huge debt to Dr. Amy Braveman of JPL; her expertise, research and data were foundational to this research.

I'd like to thank Dr. Estella Blaisten and Dr. Peter Becker for all their support throughout the University's various processes, and Melissa Hayes and Gail Hodges for making it possible for a remote student to still get anything done on campus. Finally, I'd like to thank Sally Evans for all her time and help with formatting this document.

# Table of Contents

## List of Tables

Table Page

## List of Figures

# List of Equations

## List of Abbreviation

Atmospheric Infrared Sounder…….AIRS
Entropy Constrained Vector Quantization……..ECVQ
Multi-Dimensional Scaling……...MDS

## Abstract

TECHNIQUES FOR THE EXPLORATION OF GEOSPATIAL-TEMPORAL DATASETS APPLIED TO CLUSTER COMPRESSED SATELLITE DATA

John M. Ashley, Ph.D.

George Mason University, 2013

Dissertation Director: Dr. Daniel Carr

NASA satellite data products are part of the recent big data explosion. An example of this are the individual physically referenced and processed footprints of data from the AIRS satellite (L2 Data Product), Each 2.3 MB data file covers a 6 minute period. Daily data volumes are 0.552GB/day and the collection of data products now spans over a decade. This research addressed NASA's L3Q Data Products. NASA has developed the L3Q Entropy Constrained Vector Quantization (ECVQ) cluster compressed dataset to provide a compact representation of the detailed data that retains much of the original multi-variate, altitudinally indexed information content summarized to a $5^o$ x $5^o$ Earth grid cell over a period of one month. The monthly summary files are-roughly 5.5MB in size, so the compression factor is about 3000 to 1. These multivariate L3Q monthly summaries differ from the NASA's L3 products which contain univariate statistics (means and standard deviations) for 1 x 1 degree earth grid cells.

In this research, I developed techniques to support hierarchical cluster analysis over multiple months of L3Q (ECVQ) cluster compressed multivariate data. I then developed new visualizations for the sets of multi-variate altitudinally indexed physical data vectors resulting from hierarchical clustering of the earth grid cells and their associated compression vectors. These techniques and visualizations allowed new, computationally feasible analysis and interaction with these datasets. The methods are potentially relevant to other ECVQ compressed multivariate data sets.

Specifically, I examined techniques to approximate the full distance matrix that is traditionally used in hierarchical clustering. I addressed the computational challenge of producing the distance matrix in a reasonable time by reducing the problem via an adapted method of cluster exemplars. These techniques enable practical hierarchical clustering of multiple months of data (granules), without losing the granule level detail. I examined the stability and performance of the method.

I developed the Palettized Automated Coloring Algorithm (PACA) to allow automated production of hierarchical global cluster set maps and additional maps to highlight the extent and changes over time of clusters. I then develop a graphic that displays, using multiple maps and colors, the evolution of hierarchical clusters over time.

I developed a custom graphic to allow visualization of large numbers of weighted geophysical data vectors. It used color, overplotting, and structural meta-data about the physical data vectors in a fashion that can be extended to other datasets. The visualization can be extended for exploratory and interactive use.

I applied a combination of these new techniques and tools to study several scenarios related to previous research. The graphics provided a step towards the goal of understanding what the grid cell cluster represented in terms of the geophysical variables.

# 1 Introduction

There is a fundamental question and three major topics that are addressed by this research. The fundamental question this research proposes to address is, can multiple time periods of geographically indexed, multi-variate data distribution summaries be used to do higher level analysis? It seems obvious that providing a summary of detailed data that is smaller but maintains more of the inherent multi-variate distributional data of raw observations would be at least as valuable as simpler multi-dimensional collections of uni-variate statistics; provided of course that such a summary is analytically tractable. This research demonstrates that at least some analytic tractability is within reach using as a vehicle the NASA AIRS L3Q Quantized data product.

## 1.1 The Data Clustering Problem

The first topic of research, then, is how can we manage and manipulate this data to allow users to focus on areas of interest, similarity, or difference? Data clustering is a technique that is commonly applied to higher-dimensional data to help answer these kinds of questions. There are a number of clustering techniques that are used on multi-dimensional data that fill a portion of that need; this research extends the use of hierarchical clustering to multiple AIRS L3Q datasets and accelerates the production of distance matrices for these distributions of data.

## 1.2 The Cluster Visualization Problem

The second topic is, how can the clustered data be visualized? Working with multiple maps, summaries, and color palettes selected to take advantage of existing associations, an automated coloring algorithm for temporal geospatial data is developed. This algorithm could easily be extended to other domains and datasets.

## 1.3 The Cluster of Physical Data Visualization Problem

The third topic is, how can we take advantage of structure and meta-data to visualize distributions of high-dimensional physical data, and the similarities and differences between multiple distributions? A specialized graphic is produced for the AIRS L3Q data; while the graphic is not directly suitable for other datasets, the approach is generalizable and extensible.

## 1.4 The Science

The research will demonstrate that the combination of tools and techniques developed allows us

- to combine multiple time-indexed, geospatially tagged multi-variate data distributions into an analytic whole via clustering in a computationally tractable fashion
- to interact with and explore such data
- to visualize the physical data underlying the clusters.

With these computational capabilities, some scientific insights into multi-period AIRS L3Q data are demonstrated.

# 2 Background

An understanding of the related work in Section 3 and the computational and visualization techniques employed in this work requires very little background material. The primary science data is described in section 2.1. A brief background in data clustering is presented in section 2.2. Section 2.3 describes a commonly used measure for defining the similarity (distance) between two distributions of multi-dimensional data. In section 2.4 visualization of multi-dimensional and geospatial data background is reviewed. Science background will be presented in context in section 7.

## 2.1 AIRS Satellite Data

The following description of the AIRS program is taken from the NASA JPL AIRS overview page.

> AIRS is a facility instrument whose goal is to support climate research and improve weather forecasting.
>
> Launched into Earth-orbit on May 4, 2002, the Atmospheric Infrared Sounder, AIRS, moves climate research and weather prediction into the 21st century. AIRS is one of six instruments on board the Aqua satellite, part of the NASA Earth Observing System. AIRS along with its partner microwave instrument, Advanced Microwave Sounding Unit (AMSU-A), represents the most advanced atmospheric sounding system ever deployed in space. Together these instruments observe the global water and energy cycles, climate variation and trends, and the response of the climate system to increased greenhouse gases.
>
> AIRS uses cutting-edge infrared technology to create 3-dimensional maps of air and surface temperature, water vapor, and cloud properties. With

2378 spectral channels, AIRS has a spectral resolution more than 100 times greater than previous IR sounders and provides more accurate information on the vertical profiles of atmospheric temperature and moisture. AIRS can also measure trace greenhouse gases such as ozone, carbon monoxide, carbon dioxide, and methane.

AIRS and AMSU-A share the Aqua satellite with the Moderate Resolution Imaging Spectroradiometer (MODIS), Clouds and the Earth's Radiant Energy System (CERES), and the Advanced Microwave Scanning Radiometer-EOS (AMSR-E). Aqua is part of NASA's "A-train", a series of high-inclination, Sun-synchronous satellites in low Earth orbit designed to make long-term global observations of the land surface, biosphere, solid Earth, atmosphere, and oceans.

Significantly more detailed descriptions of all aspects of the satellite and the sounder instruments themselves can be found at the following website:

http://airs.jpl.nasa.gov/technology/how_AIRS_works/how_AIRS_works_detail/ .

The following summary of the orbital characteristics is from:

http://airs.jpl.nasa.gov/technology/coverage/ .

AIRS coverage is pole-to-pole, and covers the globe two times a day. Because the swaths (scanning sweeps) do not overlap at low latitudes, some points near the equator are missed. However, these points are eventually scanned within 2-3 days.

Orbit: 438 miles (705.3km) polar, sun synchronous, 98.2+/- .1 degrees inclination, ascending node 1:30pm +/- 15 minutes, period 98.8 minutes

Ground Footprint: 90 per scan, 22.4 ms footprint

Swath Width:1650 km

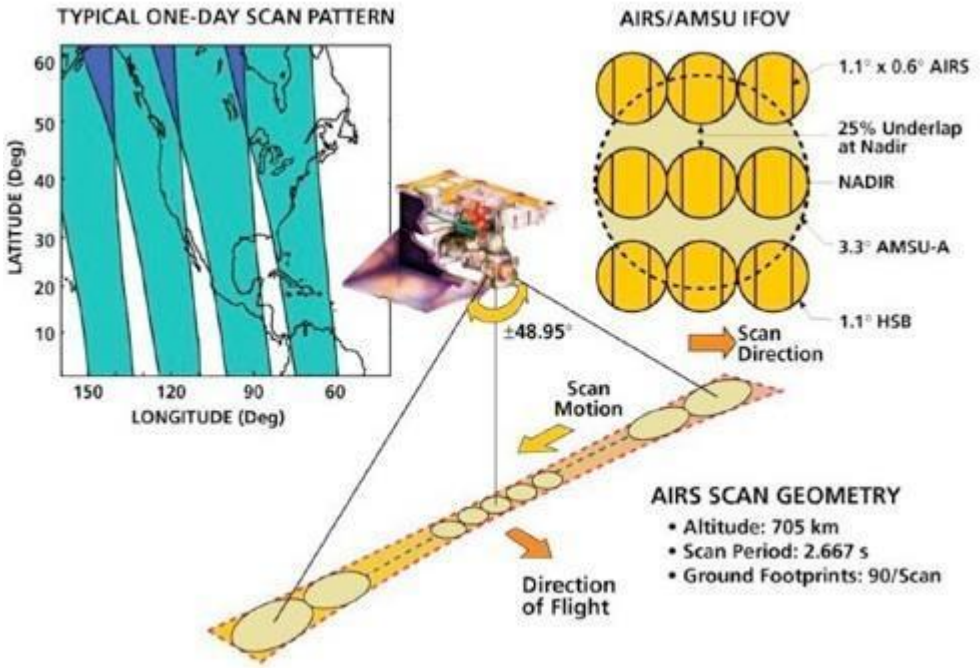



**Figure 1:AIRS Orbital Information (source: NASA JPL)**

### 2.1.1 AIRS L2 & L3 Data

AIRS L2 Data is swath data that has been algorithmically processed from the L1 spectral data (2378 spectral channels) into physical data on multiple layers of the atmosphere and captured on a "per footprint" basis. This data set contains a large number of derived fields related to temperature, water vapor, and concentration of various greenhouse gases and atmospheric dust for varying altitudes. A subset of these footprints are converted to 35 dimensional vectors for use in the L3 Quantized Summary data described below.

For more information on AIRS L2 data refer to:

http://disc.sci.gsfc.nasa.gov/AIRS/data-holdings/by-data-product/ .

L3 data is L2 data that has been summarized for various periods of time – each footprint is assigned a 5x5 or 1x1 degree cell (depending on the exact data product) and for each variable in the footprint the mean value and standard deviations are recorded along with the count of footprints in the cell. This is a very compact representation and is quite useful but it does lose any ability to accurately describe cells with distributions across a variable that are not normal – multi-modal distributions and thick or thin tails, for example, are completely lost in this product, as is any information where the value of an observation at one atmospheric pressure level are not independent of those at other pressure levels.

For more information on this product please see the following website: http://disc.sci.gsfc.nasa.gov/AIRS/data-holdings/by-data-product/ .

### 2.1.2 AIRS L3Q Data

In a number of fields, data clusters or, as they are sometime known, data signatures (prevalent in the image processing literature) are finding increased acceptance as ways to decrease the amount of data that must be processed while still attempting to capture useful and interesting detail in the original data and minimize error artifacts caused by the data reduction process. The AIRS L3Q data uses up to 100 data clusters to represent the footprint data covering a 5x5 degree grid cell. In the next section we discuss

the clustering technique used for this data summarization; here it is sufficient to point out which fields are included in the 35 dimensional data vectors that describe the clusters (cluster mean vectors).

**Table 1: Atmospheric Levels**

| Level Index | Temperature Pressure (mb) | Specific Humidity Range (mb) | Top of Cloud Cover Range (mb) | Approximate Altitude (Temp MB) |
|---|---|---|---|---|
| 11 | 150 | [TOA, 150] | | 13,500m |
| 10 | 200 | (150,200] | >= 200 | 11,800m |
| 9 | 250 | (200,250] | (200,250] | 10,400m |
| 8 | 300 | (250,300] | (250,300] | 9200m |
| 7 | 400 | (300,400] | (300,400] | 7200m |
| 6 | 500 | (400,500] | (400,500] | 5600m |
| 5 | 600 | (500,600] | (500,600] | 4200m |
| 4 | 750 | (600,750] | (600,750] | 2500m |
| 3 | 850 | (750,850] | (750,850] | 1500m |
| 2 | 925 | (850,925] | (850,925] | 750m |
| 1 | 1000 | (925,1000] | (925,1000] | ~100m |

Additional collected meta-data includes

- Scene Land Fraction

- Day Observation Fraction
- Quality Indicator Good or Better Fraction [Olsen, Braverman, Granger, Manning 2007]
-

### 2.1.3 ECVQ Data Compression

As applied to the L3Q dataset, Entropy Constrained Vector Quantization is a clustering technique (related to k-means) being used for data compression but with an additional entropy constraint that is applied as a penalty function that attempts to prevent "unnecessary" clusters from being formed. This presentation of the material is modeled very closely after [Carr, Braverman 2007]; for an alternative presentation see also [Braverman et al 2012b].

The objective function being minimized for each cell is:

**Equation 1ECVQ Objective Function**

$$L = \frac{1}{N}\sum_{n=1}^{N}\left[\| x_n - y(x_n) \|^2 + \lambda\left[-log\left(\frac{N_{y(x_n)}}{N}\right)\right]\right]$$

$x_n$ = data vector (k-dimensional data point)

$y(x_n)$= vector of centroids for the cluster to which the data vector is assigned

$N_{y(x_n)}$= Number of data vectors assigned to that cluster centroid

Note the use of the L2 norm in the equation above.

N (the number of k-means clusters) is not allowed to exceed 100 and several experimentally determined values are evaluated and the values that minimize the sum of

the grid cell objective functions over the entire data set (all grid cells) are implemented. The information theoretic factor penalizes the k-means clustering when too many clusters are used; for more details see [Chou, Lookabaugh, Gray 1989].

### 2.1.4 Principal Components (PC) Space

The 35 dimensional L3Q physical data is also processed into a principal components space. NASA has defined an 18 dimensional principal components space which accounts for about 95% of the variation in the data [Braverman et al, 2012b]. This space is used for the ECVQ compression that forms the L3Q data from the L2 data, and for the hierarchical clustering in this research.

This data is comparable from dataset to dataset as the means and covariance matrix are set for any version of the data processing. Examination of the covariance matrices provided in multiple datasets confirms this.

## 2.2 Analytic Clustering

Hierarchical clustering techniques can be divisive or agglomerative. In this research we will consider clustering techniques that are agglomerative – they start with each individual item represented as a singleton cluster and applying a function of the distance between clusters, they merge existing clusters until eventually they have merged all the observations into one cluster that includes the entire data set.

These techniques are especially useful for data exploration as they allow navigation up and down a “tree” of clusters. It is also possible to use the tree formed to

investigate how stable the clusters are by looking at the order and distance at which each cluster was merged by the algorithm.

A disadvantage of these methods is that they normally require the entire distance matrix to be calculated. There are some techniques designed to scale that pre-cluster portions of the data using some other technique and then start the clustering with the cluster centroids of the pre-clustered data – for a good survey and summary of these techniques see [Xu, Wunsch 2009]. This research extends the technique of cluster exemplars to the AIRS L3Q data and the Wasserstein or Earth Mover's Distance.

There are a number of common functional forms for the various distance functions between clusters – see [Lance, Williams 1967] or more compactly [Xu, Wunsch 2009]. Intuitively, the distance between two clusters can be easily represented as:

- the shortest distance between members of each cluster or the largest;
- the distance between cluster centroids;
- or various weighted combinations of these distances.

The actual distance measure is not constrained to be the L2 norm.

K-Means is the most basic divisive (non-heirarchical) clustering scheme. It uses some rule (possibly random assignment) to select K points in the data space of the data set being clustered and then assigns each data point to one of the clusters. Enhancements of the algorithm use an update process to adjust the cluster points K; some use subsets of the data to perform multiple passes and create "better" initial values for the K cluster points. There are versions that attempt to automatically determine K for a given dataset

and versions (such as ECVQ) that apply information-theoretic penalty functions to determine K. It is a staple technique for data clustering and analysis.

Major advantages of K means are simplicity; it does not ever require a full distance matrix, but only the distance from each data point to the K cluster points. It is computationally tractable (of order N) and parallelizable as well.

The major drawback of K means is that the parameter K is not unique, nor is determination of an appropriate value for K trivial on large datasets.

## 2.3 Distance Metric Between Multi-dimensional Distributions

Distance between two points in multi-dimensional space (two data vectors) is a simple concept; there are commonly used norms that provide useful distance metrics. The key point for this discussion is that it is simple to conceptualize the distance between two points.

So if the grid cell values for the AIRS data are only mean values for the grid cell, each cell is a data vector and the concept of distance between any two cells is simple. But in the case considered here, the grid cell is represented by a weighted set of vectors representing clusters of data, and so we need a distance function that is meaningful for the distance between two weighted sets of vectors.

The Wasserstein metric is defined for two distributions as the minimum of the expected distance between two random variables drawn from the two distributions. This definition requires only that the distance function selected lead to a true metric. This is a

metric that can be used to define the distance between the data distributions in two grid cells in the AIRS L3Q data.

In the case of probability distributions of cluster points, there is a simple analogy that leads to what in computer science (especially in the field of image processing) is known as the Earth Mover's Distance, which is equivalent to the Wasserstein distance.

To simplify the thought process, assume that the data points are located in two dimensions, and so form a flat plain, such as a parking lot. Assume that at the points corresponding to the cluster centers for the first distribution, we pile an amount of dirt proportional to the probability of that cluster point. The problem before us is then how to transform that set of piles into a set of piles that would represent the cluster centers of the second distribution while expending the least amount of energy. Since the energy expended in moving the dirt is mass times distance, this is effectively the problem of transforming one distribution into the other.

This also turns out to be a special case of the Transport problem, for which very efficient Linear Programming algorithms exist. This understanding can be formalized as follows.

Let X be the set of k-dimensional vectors $x_i$ where $x_i$ represent the clusters in the first grid cell. Let p($x_i$) be the probability or weight associated with each vector $x_i$ . Similarly, let Y be the set of k-dimensional vectors $y_j$ where $y_j$ represent the clusters in the second grid cell, and p($y_j$) is the probability associated with $y_j$. Then, let $d_{ij}$ be the L2 norm of the distance from vector $x_i$ to $y_j$. We then want to find $p_{ij}$ that minimizes Equation 2.3-a subject to constraints on $p_{ij}$.

**Equation 2 Earth Mover's Distance**

$$D(X,Y) = min \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} p_{ij}$$

We want to find the $p_{ij}$ subject to the following constraints:

$$\sum_{i=1}^{n} p_{ij} = p(y_j) \ , \forall \ j \ \in [1, m]$$

$$\sum_{j=1}^{m} p_{ij} = p(x_i) \ , \forall \ i \ \in [1, n]$$

$$\forall \ j \ \in [1, m], \forall \ i \ \in [1, n] : 0 \leq p_{ij} \leq 1$$

There are efficient libraries and methods for solving this general transport problem; the worst acceptable case would be to set $p_{ij} = p_i * p_j$ which satisfies all the constraints but generally does not minimize the value of Equation 2 Earth Mover's Distance.

There are a number of potentially useful properties of the Earth Mover's Distance proven and presented in [Rubner, Tomasi, Greibas 2000]; [Holland, Ladner, Riskin 1996] outlines methods that allow that information to be used for fast search on ECVQ data in an image processing context where exact distances are not required.

The lower bound on the distance between two distributions is the distance between the weighted average centers of the two distributions. So if X*, Y* are the weighted average (mean) vectors of X,Y then $D(X,Y) \geq D(X^*,Y^*)$. Additionally, the triangle inequality holds; in other words $D(X,Y) \leq D(X,Q)+D(Q,Y)$.

Using these relationships in clustering this data, however, is insufficiently accurate in production of approximate distance functions to be reliable.

## 2.4 Visualization

Work in visualizing the AIRS L3Q data has been conducted by Carr and Braverman [Carr, Braverman 2007a][Carr, Braverman 2007b] and earlier in [Braverman 2002], [Braverman and Kahn 2004]. Visualizations also play a part in [Zhou, Shi 2011].

For example, data has been clustered and mapped for a single year (from [Carr, Braverman 2007b]) where multiple datasets were combined by repeated application of ECVQ on the data at a grid cell level.

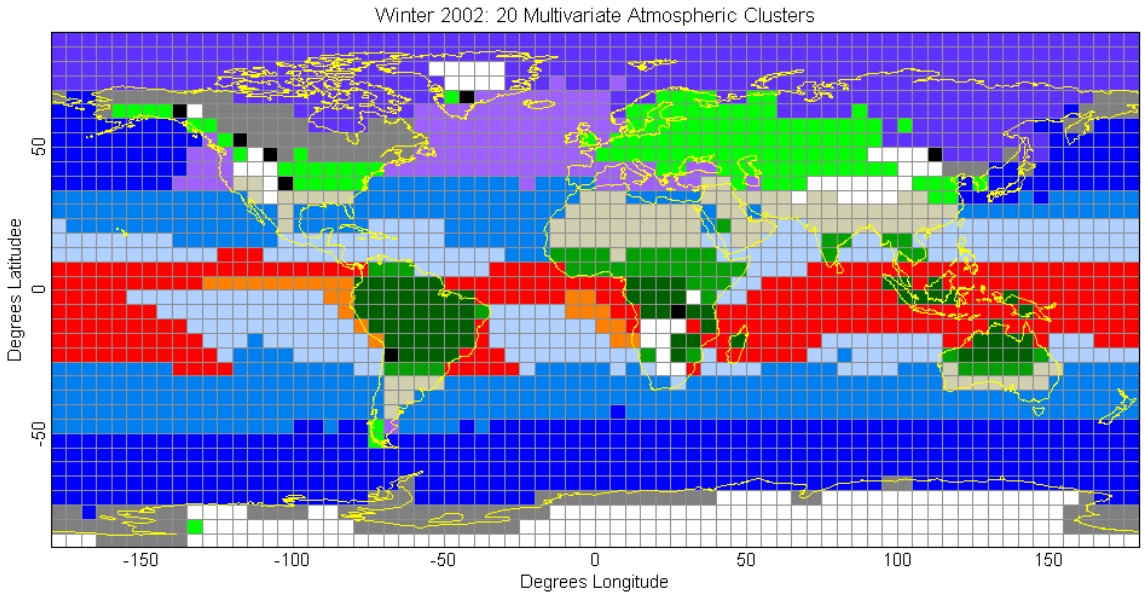



**Figure 2:From [Carr and Braverman 2007]; ECQV Clustes for one combined 3 month season.**

Figure 2, is a hand colored example showing 20 clusters, missing data, and singleton clusters. The strong banding and differentiation between clusters over land and sea was part of the inspiration for the automated coloring. More detail about the production of this graphic is in Figure 5, in Section 3 Related Work.

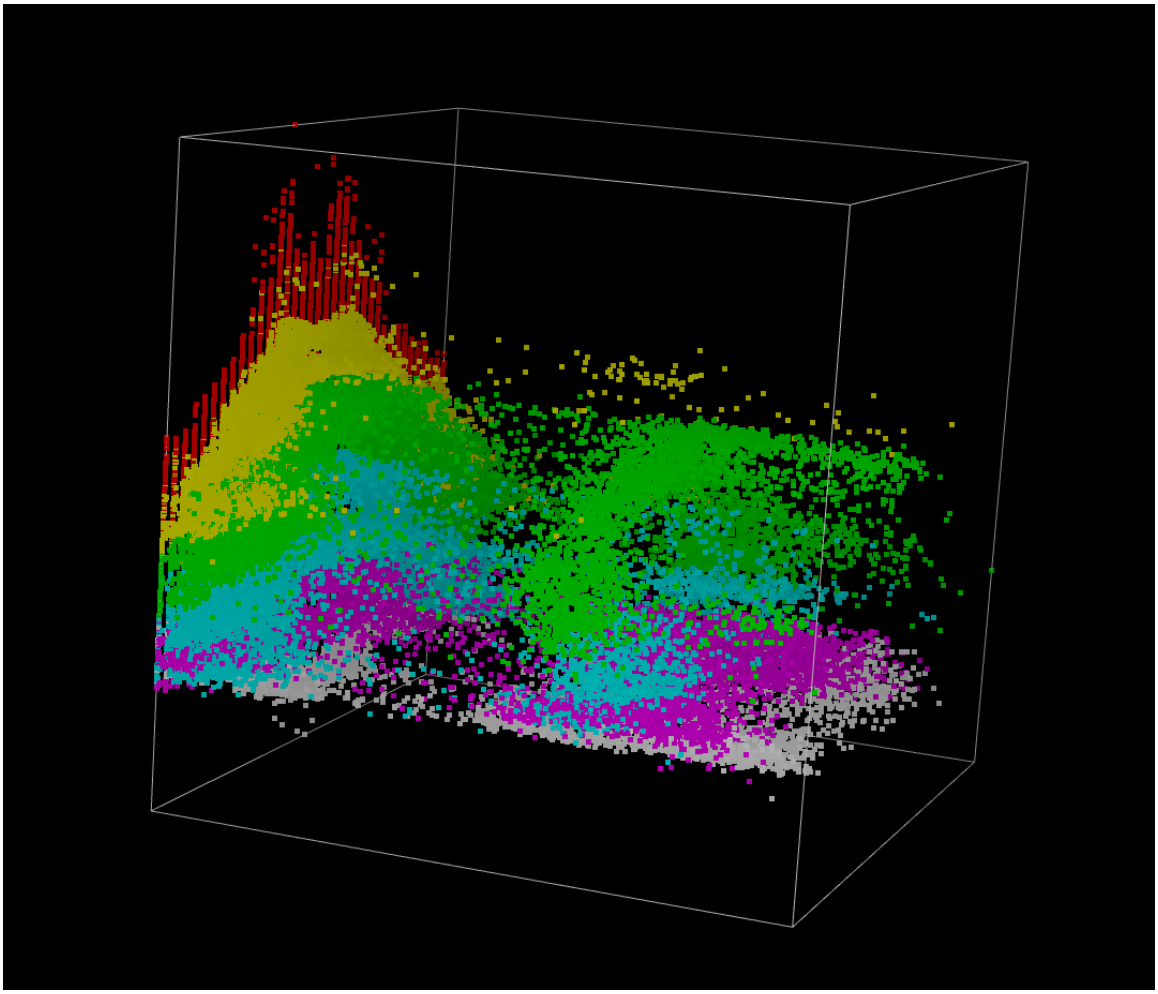


**Figure 3:CrystalVision from [Carr, Braverman 2007b].**

Figure 3 is derived from CrystalVision, a 3D visualization tool. It shows a 3-D scatterplot of Cloud Fraction, Temperature and Latitude. A 2-D scatterplot matrix brushing of global summary vector values by altitude provided the color.

Figure 4 highlights temporal dependencies of the distance data – this graphic shows the distance between November 2002 and November 2005 for each earth grid cell compressed data distribution. The distance is unitless – numerically, it is the Wasserstein

distance between distributions of weighted 18-dimensional vectors from a principle components space representation of 35 dimensional Z-scaled geophysical data vector distributions.

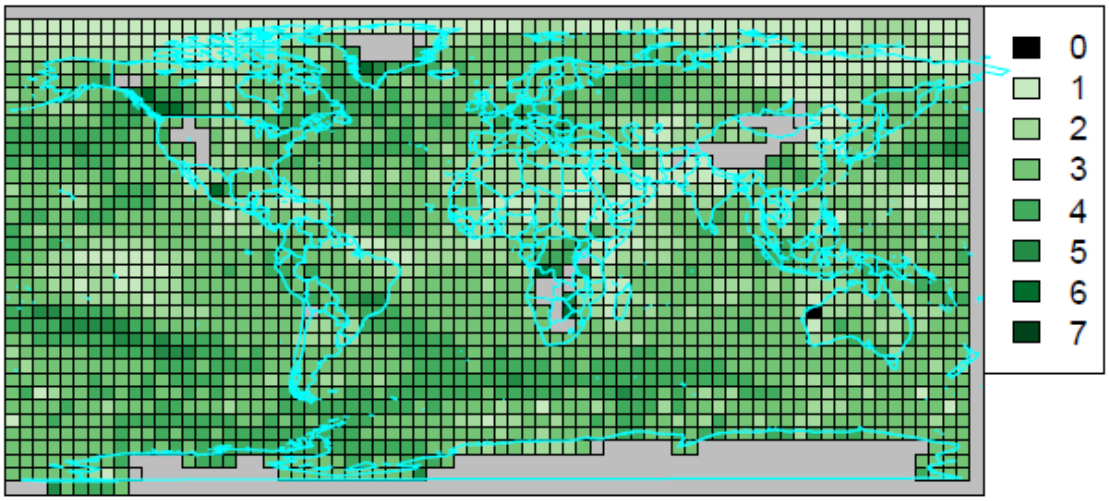



**Figure 4: Truncated Earth Mover's Distance between cells in 2002 and the same cell in 2005**

It should be noted that alternative approaches to looking at the distances between cells and extracting information from them are actively being studied. In [Braverman, Fetzer, 2006], a number of methods were examined, including hypothesis testing for determining similarity of distributions. Approaches using the first and second moments of the Mallows Distance (which is equivalent to the Wasserstein distance) are being actively investigated as well [Zhou, Shi 2011].

## 2.5 Datasets

The AIRS L3Q Monthly Version 5 Datasets are available for download from Mirador (http://mirador.gsfc.nasa.gov/) , and are formatted in HDF4 format. They occupy roughly 5MB each, and so are quite compact. There are new Version 6 datasets being produced that occupy roughly 10MB each; these files have a slightly different structure and are not strictly compatible with the Version 5 data being used in this research to date.

R, the statistical scripting language that this research is primarily implemented in, has in the past had limited library support for HDF5 format, but none for the older HDF4 which the AIRS data files are based on. In this work, because I frequently reference the data files, I have pre-processed the datafiles into an R friendly binary format. I have used MATLAB, which has native support for HDF4 file formats, as a pre-processor, to read in the datasets and break them into multiple files (by dataset, by variable) written in a (R compatible) binary format. This effectively builds a primitive object database in the file system, and provides a reasonable tradeoff between simplicity, comprehensibility, and performance. The time to read, process, and write 2 years worth of data (24 datasets) is 24.459 seconds, or just over 1 second per dataset, including all overheads.

## 3 Related Work

There is a paucity of analysis on the L3Q datasets available in the public literature. My research is most influenced by [Carr, Braverman 2007a], which includes:

- a brief discussion of the data,
- analysis of composed datasets via earth mover's distance
- hierarchical clustering,
- visualizations of the underlying physical data.

In [Zhou, Shi 2011], clustering is applied to the L3Q data and visualization of AIRS data. Additionally, they demonstrate experimentally that the distribution of the AIRS L3Q data precludes using certain theoretically applicable distance measures; their conclusion is that the Mallows Distance (equivalent to the Wasserstein distance and the earth mover's distance) is the most appropriate distance measure to use. They then apply multi-dimensional scaling to the physical data vectors from which they compare results using only the mean vectors. Data graphics are developed to examine clusters.

In [Braverman et al 2012b], there is the most thorough and understandable exposition of the motivation and theoretical underpinnings of the L3Q dataset to date.

A study of multiple winter datasets is performed leveraging the restartability of the ECVQ process – four winters are each individually summarized from three monthly datasets. Full intra-winter Wasserstein distance matrices are formed in the principal

components space, and then multi-dimensional scaling is applied and the results analyzed.

## 3.1 Detail: [Carr, Braverman 2007]

All visualizations in this section unless otherwise noted are excerpted from [Carr, Braverman 2007].

Winter 2002 (2002.12, 2003.01, 2003.02) data was clustered into a single dataset by application of ECVQ to the underlying monthly summaries. It was then hierarchically clustered. Manual exploration and coloring produced the following image, where white denotes areas of no data, and black denotes singleton clusters. Twenty clusters were colored.

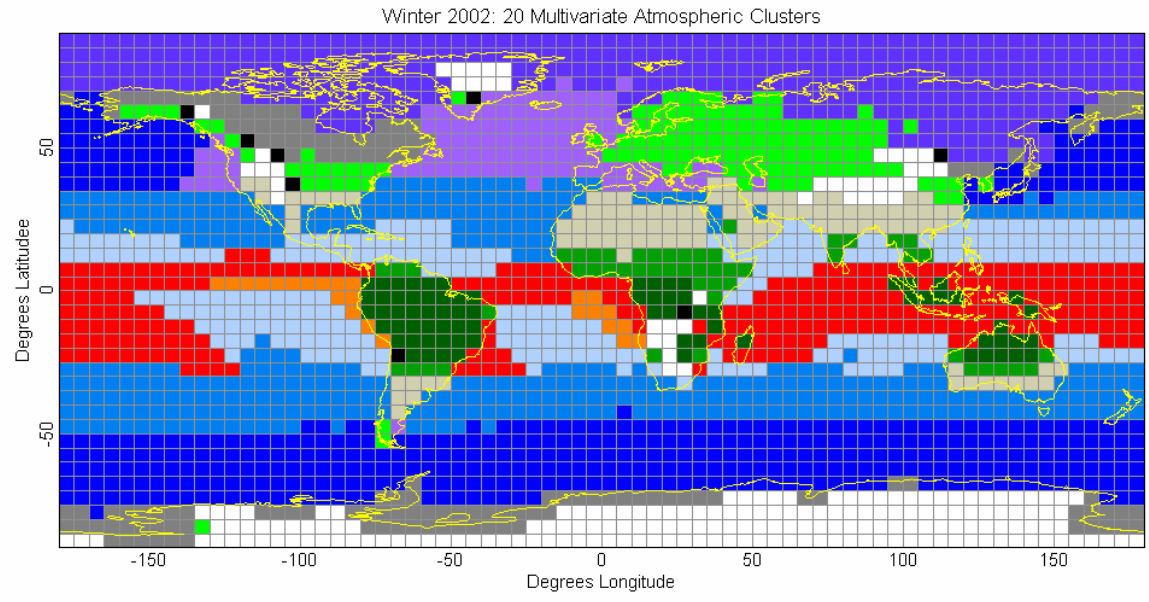



**Figure 5: Winter 2002, 20 Clusters [Carr, Braverman 2007]**

This hand-built example has some very useful features as a visualization. . The example used 20 clusters, 8 of which were singletons which were all colored black. White colors encoded the grid cells with missing values. The remaining 12 clusters were distinguished with a custom chosen color palette. Since the clusters do show a strong tendency towards including either land or ocean cells. This was interesting since only one element of the 35 geophysical parameter vectors elements directly addresses this distinction. For the grid cell clusters a palette of greens and greys was used for land clusters -- motivated by Tom Van Sant's Earth from Space poster. For the five dominant land clusters, three shades of green were associated with vegetation and two shades of gray were associated with arid clusters. This left colors dominated by red and blue forming a palette to associate with the seven ocean clusters – red, orange, three shades of blue, violet, and purple. The assignment of colors to clusters was roughly associated with sea surface temperature and latitude. Overall, colors were chosen so that they were easy to distinguish.

The approach was extended to datasets for the winters of 2003, 2004, and 2005. Below the clustering was done separately for each year. The clusters that appear over the same part of the globe don't necessary reflect the same discrete multivariate distribution of summary vectors. Dr. Carr speculates that atmosphere process will vary to year to year over and above seasonal effects and that in some cases earth surface factors are sufficiently strong that after controlling for seasonal variation groups of contiguous grid cells will tend to be in one or very few clusters year after year. The emergence of new, stable and geospatially contiguous clusters is interesting. Selection of 20 clusters in

2003 to fewer singleton clusters, but produced a small emergent cluster over China (colored pink). Additional colors were added manually to the palettes. Manually increasing the number of clusters generated to 27 and 21 clusters in 2004 and 2005 respectively recovers the same cluster. Dr. Carr speculates that if these cells are actually a distinct cluster across these three years, might air pollution effects be a strong enough factor to make this atmospherically distinct?

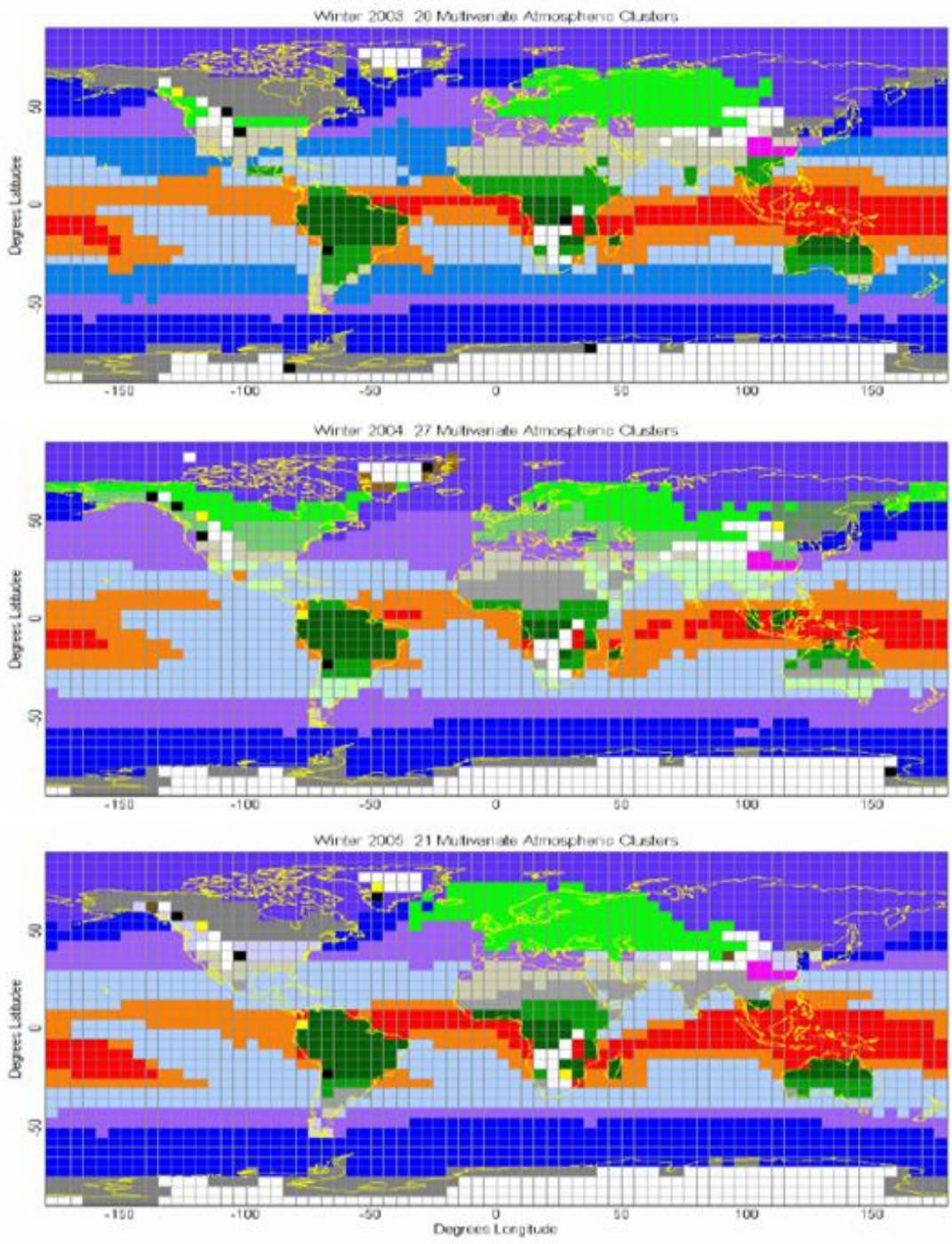



**Figure 6: Separate Views: Copied from Winter 2003,04,05 [Carr, Braverman 2007]**

The hand coloring and the similarity in the cluster shapes suggests that there are common clusters across the datasets but that may simply be an artifact of the selection of varying numbers of clusters in each winter dataset.

A number of techniques were explored to represent the multi-variate physical data. . In Figure 7, a conditioned choropleth map used a 6 x 6 set of panels to show a 6 x 6 set of earth grid cells. Each vertical rectangle in a panel represents an ECVQ cluster vector value. The rectangle width is proportional to the number of observations (footprints) in the vector weight. Each vertical rectangle is divided into 11 rectangles corresponding to 11 altitudes. Three colors were used to dynamically control cloud fraction color. Dynamic conditioning and filter views using temperature and cloud fraction are not shown. This approach does not scale to the full globe or to clusters with significantly larger number of compressed summary vectors (or to the set of all compressed vectors in a larger hierarchical cluster of earth grid cells).


**Figure 7: Conditioned Choropleth Map [Carr, Braverman 2007]**

Two and three-dimension scatterplots with linked brushing were used to study all the ECVQ cluster means and sizes for Winter 2002. This shows all the ECQV vectors in geophysical units plus latitude and longitude. Linked brushing views (here based on altitude) also include parallel coordinate plots and 3D scatterplots with motion parallel stereo.

Each 37 component vector (35 geophysical AIRS L3Q data elements plus latitude and longitude) became 11 altitude specific vectors with some replicated values to accommodate the 20 variable limitation in the software. Strengths of the method include

interactivity and a varity of statistical tools.A weakness is that the technique does not preserve the inter-altitude structure of the data (although in 3D it does maintain the intra-altitude relationships).

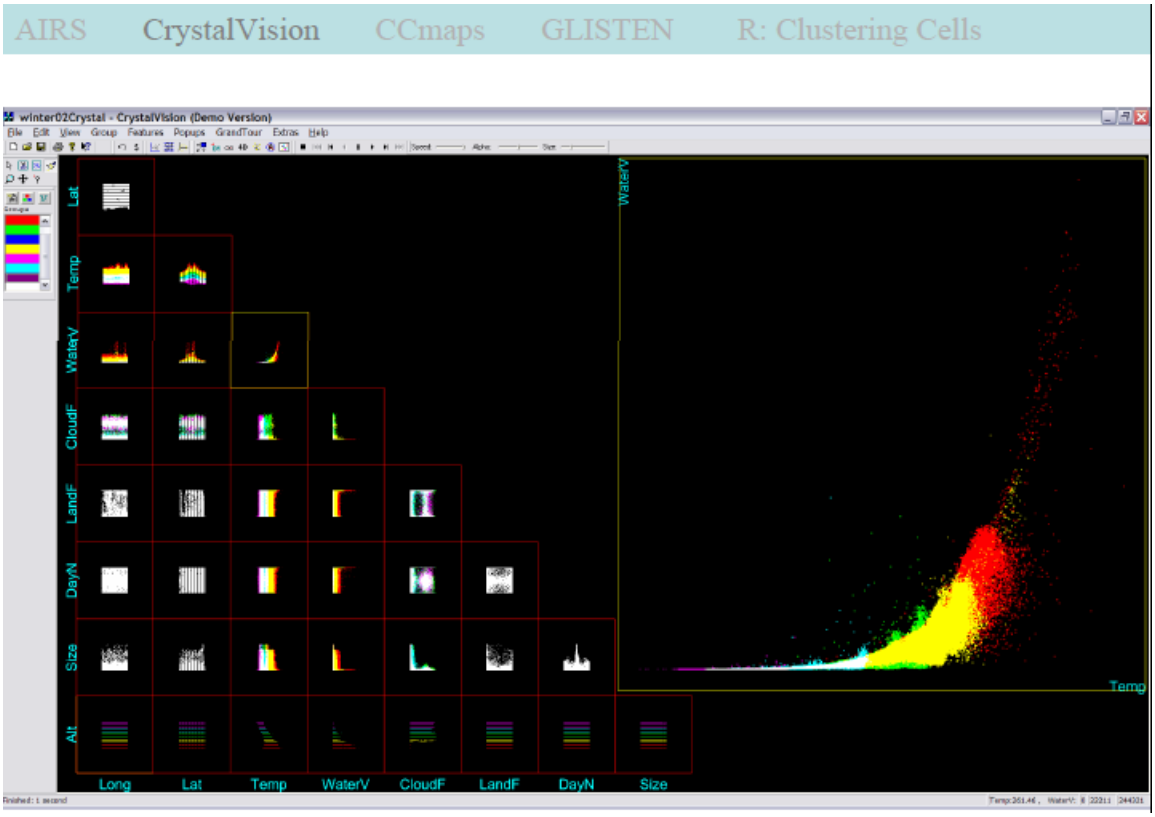



**Figure 8: CrystalVision Scatterplot Matrix [Carr, Braverman 2007]**

## 3.2 Detail: [Zhou, Shi 2011]

All images in this section are from [Zhou, Shi 2011] unless noted otherwise. The research ends up focused on the use of a multidimensional scaled version of the Mallows

distance between dimensionally reduced and Z-scaled AIRS L3Q physical data rather than the principal components vectors. The analysis bears some similarities to [Braverman et al 2012b] as it follows an earlier version of that paper.

They examine a single month of AIRS data (December 2002). They remove the indicator variables (Good, Land, Day Fractions) and then scale each remaining physical dimension to have mean zero and standard deviation one ~ N(0,1). This effectively weights each dimension identically. Clusters are produced using Mallows distance and Mean distance, with Mallows distance ending up being the authors' preferred measure as they conclude that the distribution carries additional information beyond the mean.

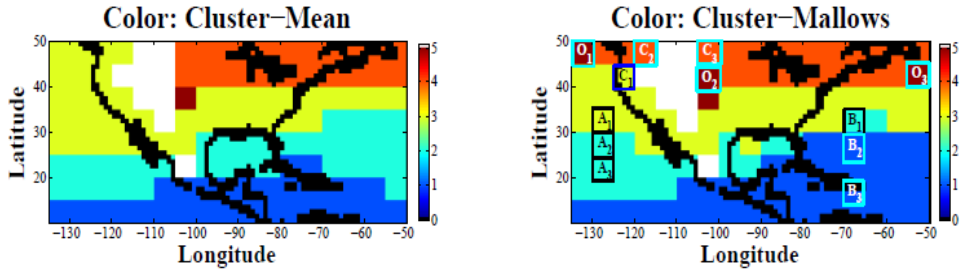



FIGURE 5. Clustering displayed in geographic maps: Three panels are colored by *Cluster-Mean* and *Cluster-Mallows*. White blocks in the maps show the locations where data are missing.

**Figure 9: North America Dec 2002 Clusters from [Zhou, Shi 2011]**

In Figure 9, above, the Mallows distance identifies additional outliers and has slightly different cluster memberships. The blocks labeled A are used by the authors to examine physical variation using the graphics below. In each block, there are 32

columns; the first 11 correspond to temperature, then the next block corresponds to specific humidity, and the last to cloud fraction. The top row is the mean data for the entire cell. Each row below that is an individual data vector from a cell, whose height is proportional to the relative weight. Color is used to encode the scaled value of the physical data value, clamping the Z score to the range [-3,3].

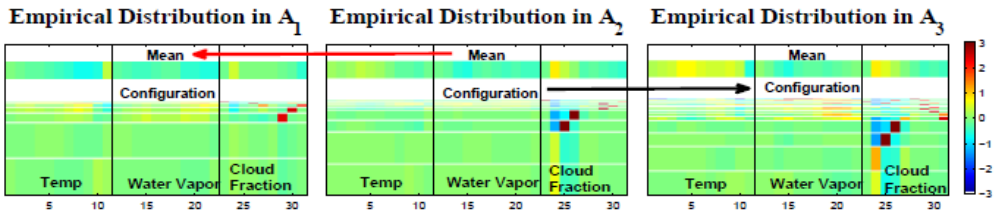



**Figure 10: Physical Data Visualization; excerpt from Figure 7 [Zhou, Shi 2011]**

## 3.3 Details: [Braverman et al 2012b]

All figures in this section are taken from [Braverman et al 2012b] unless otherwise noted. The paper starts with a recap of the AIRS satellite data, and outlines the scale of the challenge in dealing with the processed footprints in the AIRS L2 data – about 240 granules of each of about 5MB per day – 1350 data vectors per file.

The methodology of converting L2 to L3 data is reviewed, and then the underpinnings of the production of quantized compressed summaries (L3Q) data is described. As an example of the use the data can be put to, seasonal summaries are

produced by repeated application of ECVQ to December, January, and February data starting in 2002 and covering 2003, 2004, and 2005.

Usefully, the JPL team has provided timing data in their Table 1, to which I will refer later. While not detailed in the table, they also mention that the time to compute the Wasserstein distance using the GNU library they employed is roughly 1 second per cell pair.

Table 1. Compression statistics for 12 months covering the winters of 2002, 2003, 2004, and 2005

| | Raw | 5-day | Monthly | Seasonal |
|---|---|---|---|---|
| Total file size | 32.78[1] GB | 3.10 GB | 148.47 MB | 40.96 MB |
| Average PC-scale summary distortion | 0 | 0.83[2] | 7.59[3] | 9.42[4] |
| As a proportion of total variation | 0 | 0.06[5] | 0.26[5] | 0.19[5] |
| Computation time (hours) | NA | 10.29[6] | 7.2[7] | 2.4[8] |

NOTE: [1]378 KB/day × 240 granules/day × 361 days.
[2]Average grid cell distortion over all nonempty grid cells for each 5-day period. Seventy-two 5-day values are then equally weighted.
[3]Average grid cell distortion over all nonempty grid cells for each month. Twelve monthly values are then equally weighted.
[4]Average grid cell distortion over all nonempty grid cells for each month. Three monthly means averaged to form a seasonal mean; then these values equally weighted.
[5]Total variation is the weighted average of squared distances between cluster representatives and grid cell means, plus grid cell distortion.
[6]16,210 seconds required to process 7 years of data on 8, dual core 2.2 GHz AMD Opteron processors with individual executions on a single core. This equates to 37,051 seconds for 1 year on one processor.
[7]Calculated as 12 × 36 minutes to process 1 month on one Mac 3.2 GHz processor.
[8]Calculated as 4 × 36 minutes to process one season on one Mac 3.2 GHz processor.

**Figure 11: Timings from [Braverman et al 2012b]**

The JPL team then applies multi-dimensional scaling to the four overall distance matrices they computed in the principal components space to produce corresponding 6 dimensional vectors. They then study the first two dimensions from the multi-dimensional scaling, MDS1 and MDS2, using a variety of graphics.

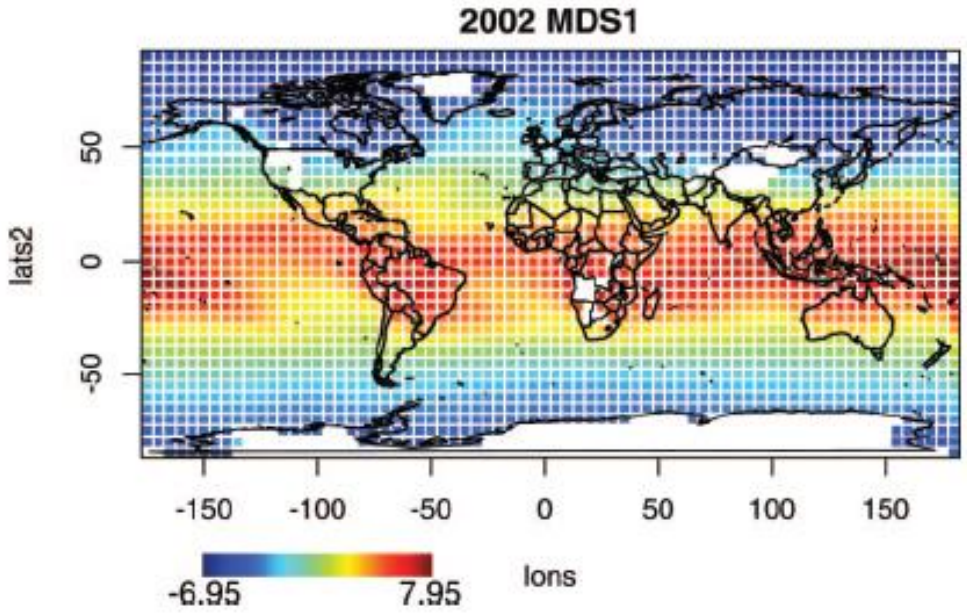



Figure 7. MDS1 coordinate values displayed in geographic space. 2002 shown, other years are visually indistinguishable from 2002. White pixels correspond to grid cells with no AIRS data present.

**Figure 12: MDS1 Coordinate Values from [Braverman et al 2012b]**

Note again the general structure of these values off the coast of South America and Africa. At a high level, this structure is similar to that found by [Zhou, Shi 2011] from MDS scaling of the raw physical data and in [Carr, Braverman 2007] from hierarchically clustering the principal components data distances directly. MDS1 is strongly correlated with latitude and surface temperature (which relationship has an obvious physical coupling).

The JPL team also finds structure in the MDS2 dimension although in 2004 they have to reverse the sign of the MDS2 dimension to maintain the apparent structure of the

pattern between the years. Given that the sign of the dimension is effectively arbitrary in multi-dimensional scaling, this isn't unexpected – but it is another manual intervention. Between $30^{o}$ N and $30^{o}$ S MDS2 appears to be fairly strongly correlated to an atmospheric measure called vertical velocity.

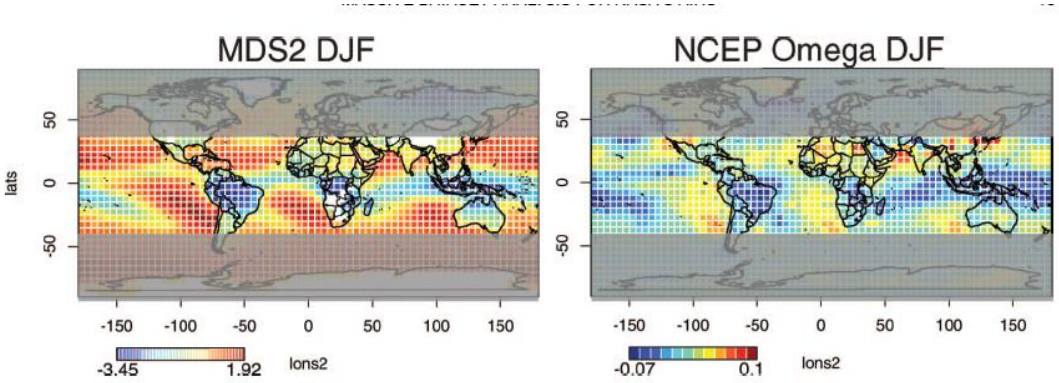



Figure 10. Comparison of average (negative) MDS2 over winters 2002–2005 (left) for the region between 30°S and 30°N with the NCEP reanalysis average omega for December, January, and February, 1968–1996 (right). White pixels correspond to grid cells with no AIRS data present.

**Figure 13: MDS2 & Vertical Velocity [Braverman et al 2012b]**

Omega is the Lagrangian derivative of pressure with respect to time at the 500hPa pressure level, and is a measure of vertical velocity. Additional analysis by the JPL team highlights this more fully in the paper. Of interest to me here especially is the general shape of the clusters, the scientific and mathematical underpinnings, and the fact that all these mappings were produced and colored via an iterative manual process.

# 4 Creating Globally Hierarchically Clustered Sets

There are a number of challenges related to building clusters of multi-dimensional data distributions over time. NASA scientists have an approach that allows clustering of large volumes of Level 2 (processed satellite scan footprint) data into L3Q data (cluster compressed summary data). The details of this ECVQ approach are outlined in Section 2 Background; specifics of the implementation are in [Braverman et al 2012b]. Repeated application of the ECVQ method allows multiple datasets to be combined into a single dataset. This research, however, looks at combining multiple datasets into a consistent whole while maintaining the lower level summaries using hierarchical clustering as a vehicle.

The first challenge, then, is finding a suitable distance metric to express the similarity or difference of two empirical distributions. The Wasserstein distance (which involves solving a linear programming problem) is computable, and has all the needed properties for a proper metric. Proper metrics allow many other analytical techniques which rely on a distance metric to be used. These include agglomerative and divisive clustering methods and self-organizing maps. Agglomerative clustering, for example, was shown in [Carr, Braverman 2007].

The second problem is that while the distance function would be well defined in the physical measures, the various physical data is not independent between variables or altitudes, nor are the scales of the measurements of equal informational value. NASA

scientists, in the construction of the AIRS L3Q dataset, have applied Principal Components Analysis to the 35 measured and derived physical variables and reduced them to 18 dimensions in a PC space. This reduces the computational complexity of the underlying distance calculation and also helps with the scaling of the variables which are in different units.

The third problem is one of scale. Each monthly AIRS L3Q dataset could have as many as 2,592 grid cells, although the average is closer to 2200 due to issues with data calculation over ice, snow and high altitude regions. If $N_m$ is the number of months of data to be studied, then calculation of the distance matrix requires the solution of approximately $\frac{(2200N_m)^2}{2}$ linear programming problems. Experimentally, a reasonably capable CPU core today can calculate approximately 82 distances per second.

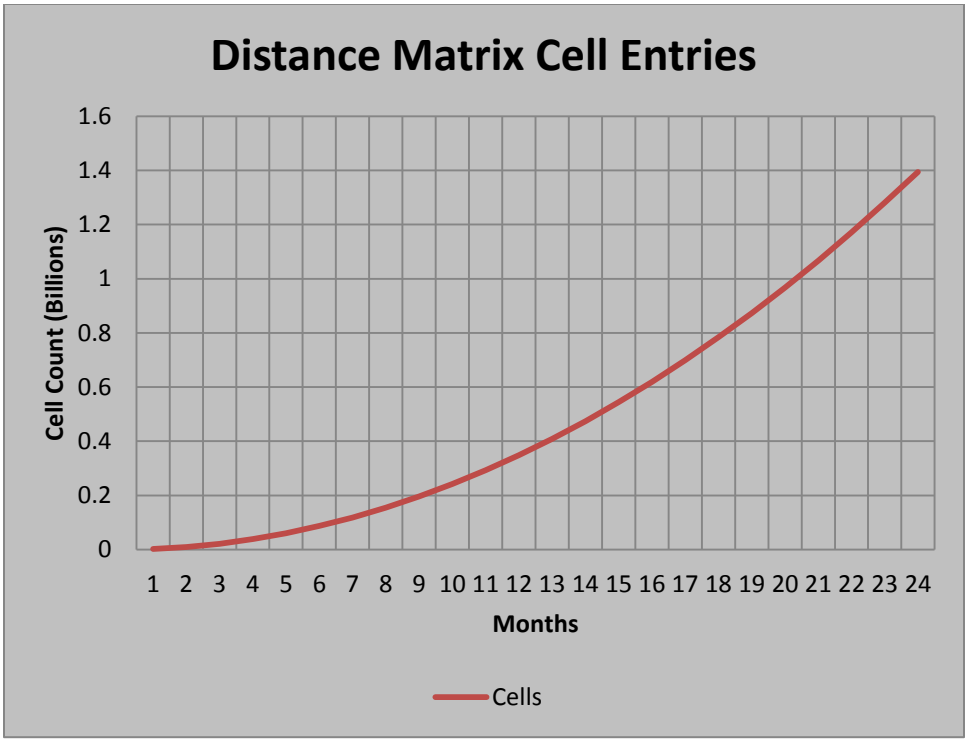



**Figure 14: Size of Full Distance Matrix by Month**

The chart above shows the number of entries in the distance matrix, which grows dramatically over time. Another way to look at this would be to compute the number of CPU cores required to populate the distance matrix in 60 minutes.

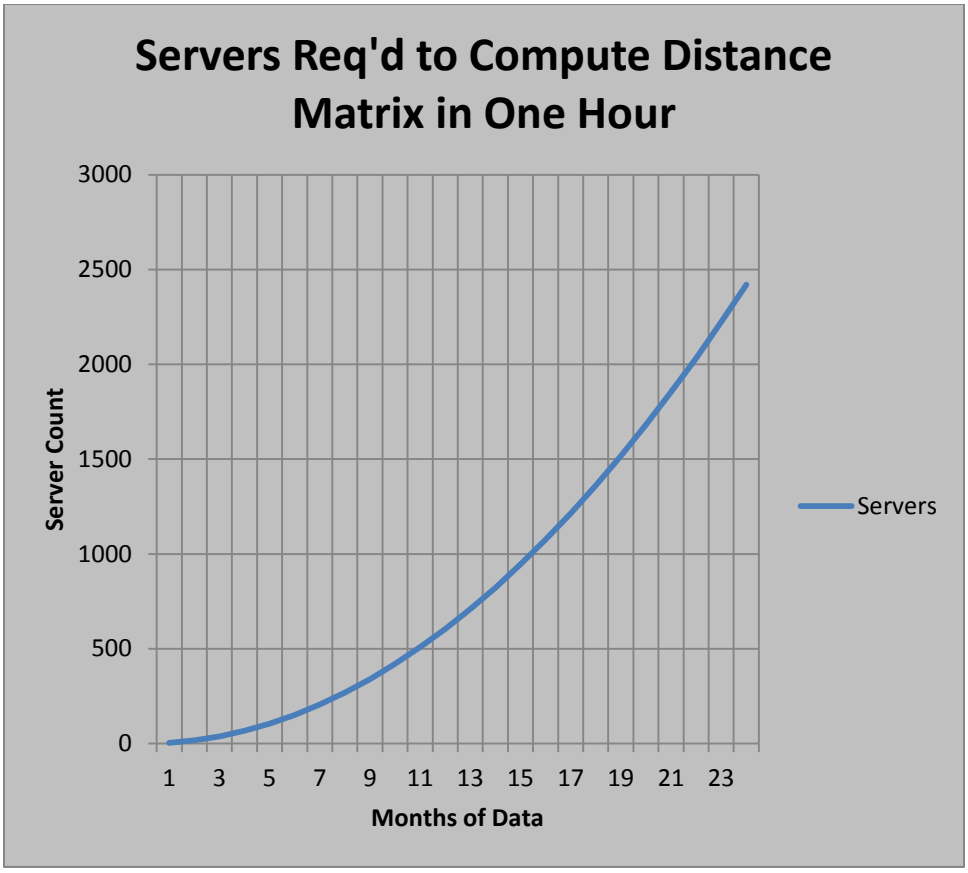



**Figure 15: Servers to compute full distance matrix in 1 hour**

This rapidly becomes cost prohibitive.

Storing the distance matrix for all prior months would be an option; in this case with sufficient storage space, the calculation of the distance matrix elements that need to be added becomes linear (effectively only requiring filling the bottom row of blocks of the distance matrix). By the tenth year, however, this requires $0.5(120 * 2200)^2$ values. If stored as IEEE double precision floating point numbers, this requires approximately 278 GB of storage. Working with this amount of data in system memory is not possible

on many machines today. This would require disk based storage which would have significant performance impacts. The data is growing at a rate proportional to $O((2200N_m)^2)$ which will grow faster than compute power.

Given that the problem of computing the distance matrix itself is growing with $O((2200N_m)^2)$, it would be reasonable to target a technique that drastically reduces the constant portion (2200) of the problem scaling and allows some level of user control – letting them trade time for accuracy, while still delivering reasonable values for both. This could involve using faster approximations in the distance calculation, a further reduction in the dimensionality of the distributions, or some other means of reducing the volume of computation and storage required.

At this point, it is logical to ask, based on the prior work, why it isn't sufficient to simply restart the ECVQ process to combine any set of datasets for study? Certainly, from Figure 11: Timings from [Braverman et al 2012b], it seems that for 12 months of data, instead of requiring 100 servers for an hour, some small multiple of 36 minutes on a Mac will suffice to do the compression followed by a couple of hours to do the final distance matrix.

This repeated application of ECVQ is a powerful tool, and if the desired time granule of interest can be reduced to a single unit (a month, a season, a year), then the technique appears to deliver excellent results. It does create a new atomic granularity in the data, however. If a researcher is looking for climate patterns in multiple months rather than seasons, this might prove to be unacceptable. Or, if the research wanted to identify similar patterns in Northern and Southern hemisphere winters, the six months required

would form a single dataset; this would confound the summer and winter patterns in each hemisphere.

In the event that multiple seasons were to be clustered together, the techniques developed here for accelerating the computation and managing the clustering and visualization of the AIRS L3Q data would still work with minor code modifications.

An alternative approach to multi-year clustering would be to compute the overall ECVQ compressed summary for all the data, over some chosen basis in terms of geospatial and temporal distribution of cells. All "raw" cells could have their distances to these "superclusters" calculated, or bookkeeping in the production process could record which grid cells contributed what portion to each "superclusters". Maps of these membership or affinity statistics might provide further interesting insights. This is left as a future research topic.

Another, potentially very fast alternative, would be to calculate the distance matrices and clusters on a dataset by dataset basis, once -- similar to the pre-processing this research uses – and then use the Jaccard distance between clusters to combine clusters across datasets.

**Equation 3: Jaccard Distance**

$$D_{Jaccard}(A,B) = 1 - \frac{A \cap B}{A \cup B}$$

The Jaccard distance can be quickly calculated with two accumulators and a final division; extremely fast relative to any of the other options presented here. This technique has an implicit assumption built in, however – that distinct clusters that cover the same earth grid cell in multiple datasets are, in fact, the same cluster. This is in some sense akin

to the seasonality effect that can result from repeated application of the ECVQ technique. Quantification of this effect is left as a future research topic.

## 4.1 Approximating the Wasserstein Distance

Since calculation of the Wasserstein Distance is the performance bottleneck in my analysis of the cluster compressed AIRS data, it is natural to ask if that computation can be accelerated in any way, including perhaps by approximation.

As indicated in section 2.3, [Rubner, Tomasi, Greibas 2000] showed that there is a lower bound on the distance between two distributions X and Y with mean X* and Y* -- $D(X,Y) \geq D(X^*,Y^*)$. If there is no need for a complete distance matrix (for example, in a serial aggregation of clusters that will terminate at a distance cutoff, or for image search) this could be a useful screen prior to calculation of the actual Wasserstein distance. If the full distance matrix is required, or the lower bound is insufficiently discriminatory then this doesn't help.

[Barbour, Xia 2006] showed a likely upper bound on the metric distance between two distributions is derived based on Poisson approximations of the distributions. The quality of the result is related to the quality of the fit of the Poisson distribution to the underlying data. [Horowitz, Karandikar 1994] showed an upper bound on the square of the Wasserstein distance assuming that the sets X and Y were drawn from i.i.d. distributions. These results are interesting but would require significant further refinement to apply to smaller empirically distributed datasets such as this.

[Shirdhonkar, Jacobs 2008] proposed a wavelet transformation that allows estimation of the Earth Mover's Distance to within a set of error bounds. Experimentally, these bounds are shown to range from a factor of 4 to a factor of 10. This error range still allows the technique to be useful in the domain of image search but, in its current form, renders it ill-suited for approximating the needed distances in this research.

[Kreitmeier 2011] showed the optimal compression error in an entropy sense is expressible in terms of the Wasserstein distance between the original and the compressed distribution. While interesting, this is not a useful result for this research.

Recall our expression of the problem.

$$D(X,Y) = min \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} p_{ij}$$

$$\sum_{i=1}^{n} p_{ij} = p(y_j) \ , \forall \ j \ \in [1, m]$$

$$\sum_{j=1}^{m} p_{ij} = p(x_i) \ , \forall \ i \ \in [1, n]$$

$$\forall \ j \ \in [1, m], \forall \ i \ \in [1, n] : 0 \leq p_{ij} \leq 1$$

Does knowledge of a minimum intra-set distance between members within earth grid cells X and Y give any practical insight into the distances $d_{ij}$? If we calculated some strategically chosen set of inter-grid cell vector L2 Norm distances $d_{ij}$ we could estimate upper and lower bounds on $d_{ij}$; however calculation of $d_{ij}$ is highly parallelizable and relatively quick, and so we can have the actual result in similar time to what it would take to get upper and lower bounds on the value.

Is there performance to be gained in the calculation of the $d_{ij}$ themselves, by approximating $d_{ij}$ by $d_{ij}$* which ignores some number of dimensions of the vector means from the distance calculation? In theory, this would have a relatively small impact, as the square root of the sum of the squares of differences is more and more dominated by the execution of the square root function as the number of summand terms decreases. This will also introduce a bias into the overall calculation, underestimating the true distance. It is possible that caching and other architectural effects could provide an unexpected performance boost for this technique, and so a simulation was performed. Starting from a 18D vector, vectors are removed from the distance calculation and performance and accuracy of the resultant distances relative to the full distance are computed. From Figure 16: L2 Norm Calculation Times, it is apparent that the time to do the calculation is relatively insensitive to the number of dimensions used.

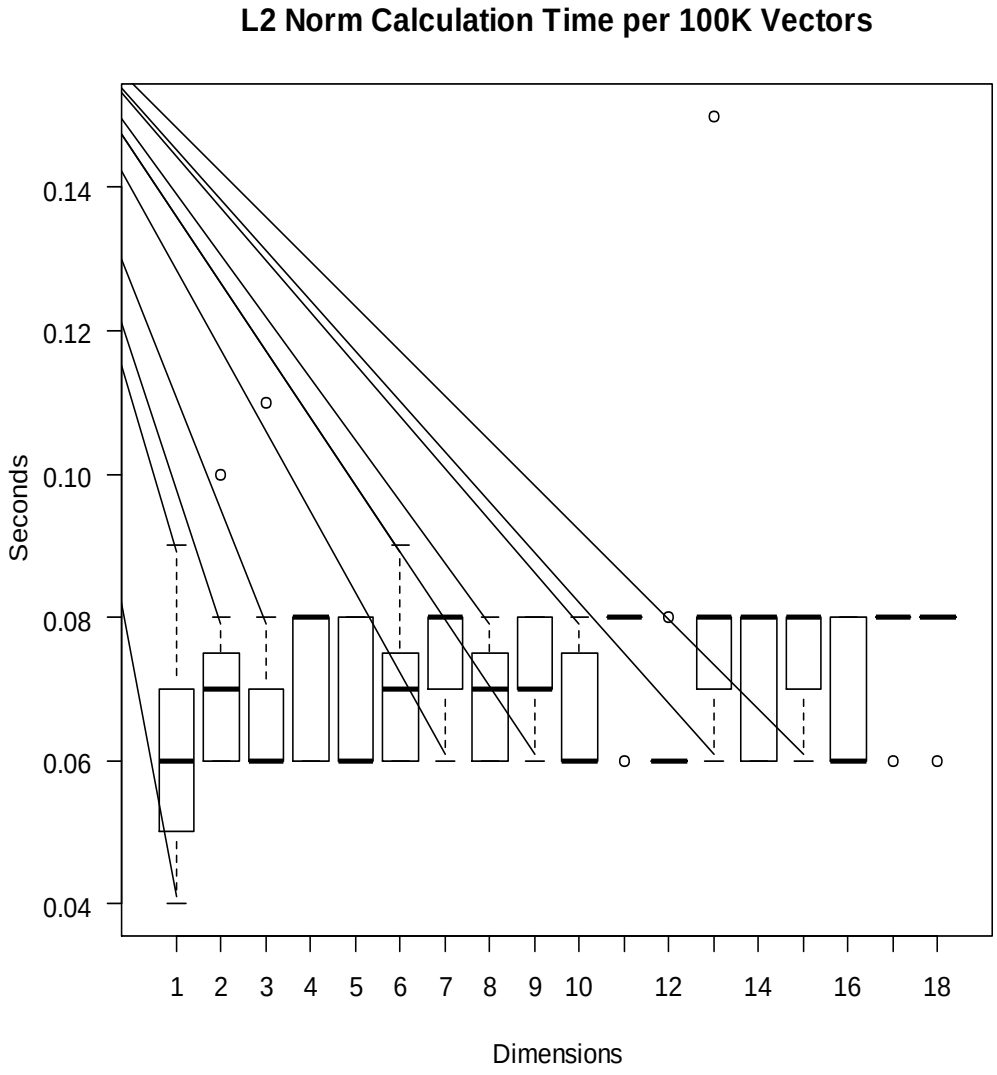



**Figure 16: L2 Norm Calculation Times**

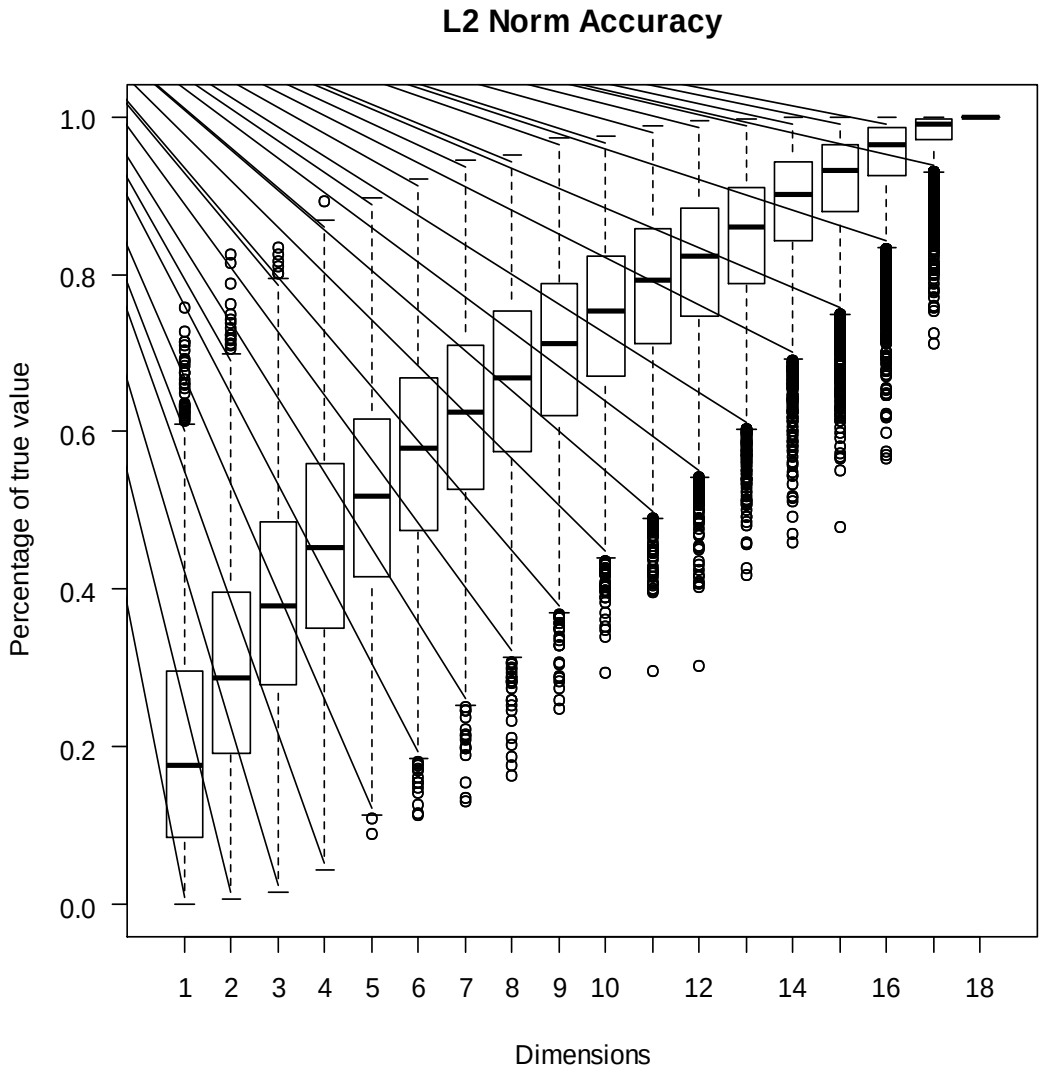



**Figure 17: L2 Norm Accuracy**

Figure 17 above shows the fractional percent of the true value that reducing the dimensionality due to reducing the number of vector components used in computation of the L2 norm.

Could we gain performance in the calculation of the summand terms $d_{ij}p_{ij}$ by ordering X and Y w.r.t. the probability weights and dropping some of the vectors with small weights? This would deliver an estimate of D(X,Y) that would have some bias, and would reduce the size of the systems that needed a linear solve, which will impact total compute times.

Implementation would require pre-computed sorts, but that price could be paid once in the pre-processing of the data, when the complete intra-month distance matrix is computed. The question then is, does this have enough performance impact enough of the time to be worth introducing an approximation with a likely bias into the mix?

In Figure 18, it is apparent that a consistently high bias is introduced via weight censoring. This computation was run on a sample of random data By removing vectors and reweighting to enable the linear programming routines to work, probability weight has to "move further" on average. Performance-wise, Figure 19 shows that for larger numbers of vectors, there is some benefit to lightly trimming the tail of the distribution.

Examination of the distributional data in Figure 20 reveals a relatively fat and flat tail beyond about 37 compression vector summaries. The performance graph in Figure 19

certainly suggests that there could be some small (factor 2-5) overall speedup available using this technique.

It seems more likely that rather than censoring based on weights, which is by definition discarding some of the data in the production of the approximation, that using weighted K-means to further compress the number of vector representatives in the earth grid cell compressed data clusters to form a 20-cluster or 30-cluster summary directly to represent the earth grid cell for distance calculations might speed up the entire tail of the performance curve with potentially less impact to accuracy. This study is an interesting area for future research. The benefit of censoring seems small relative to the potential bias being introduced, and so this research did not pursue it further.

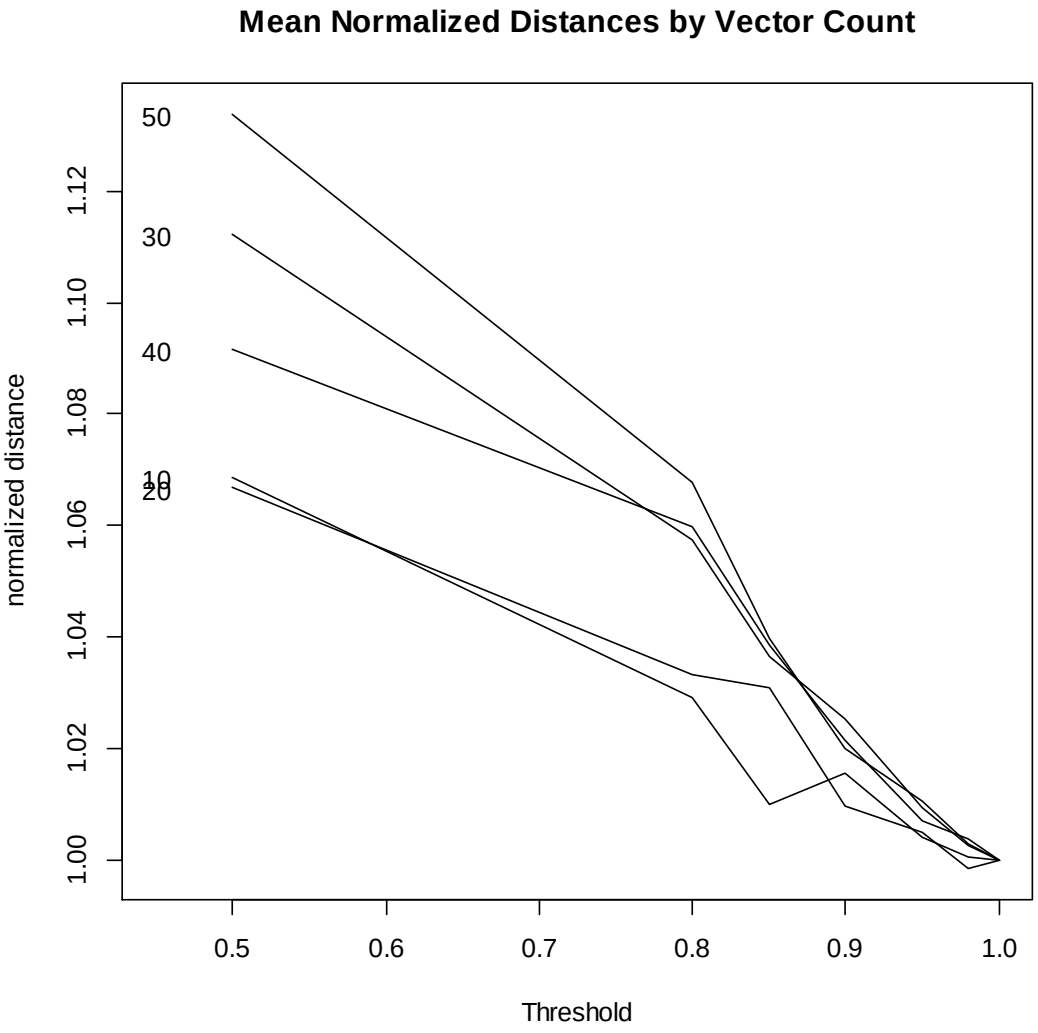



**Figure 18: Impact of threshold censoring on distance metric**

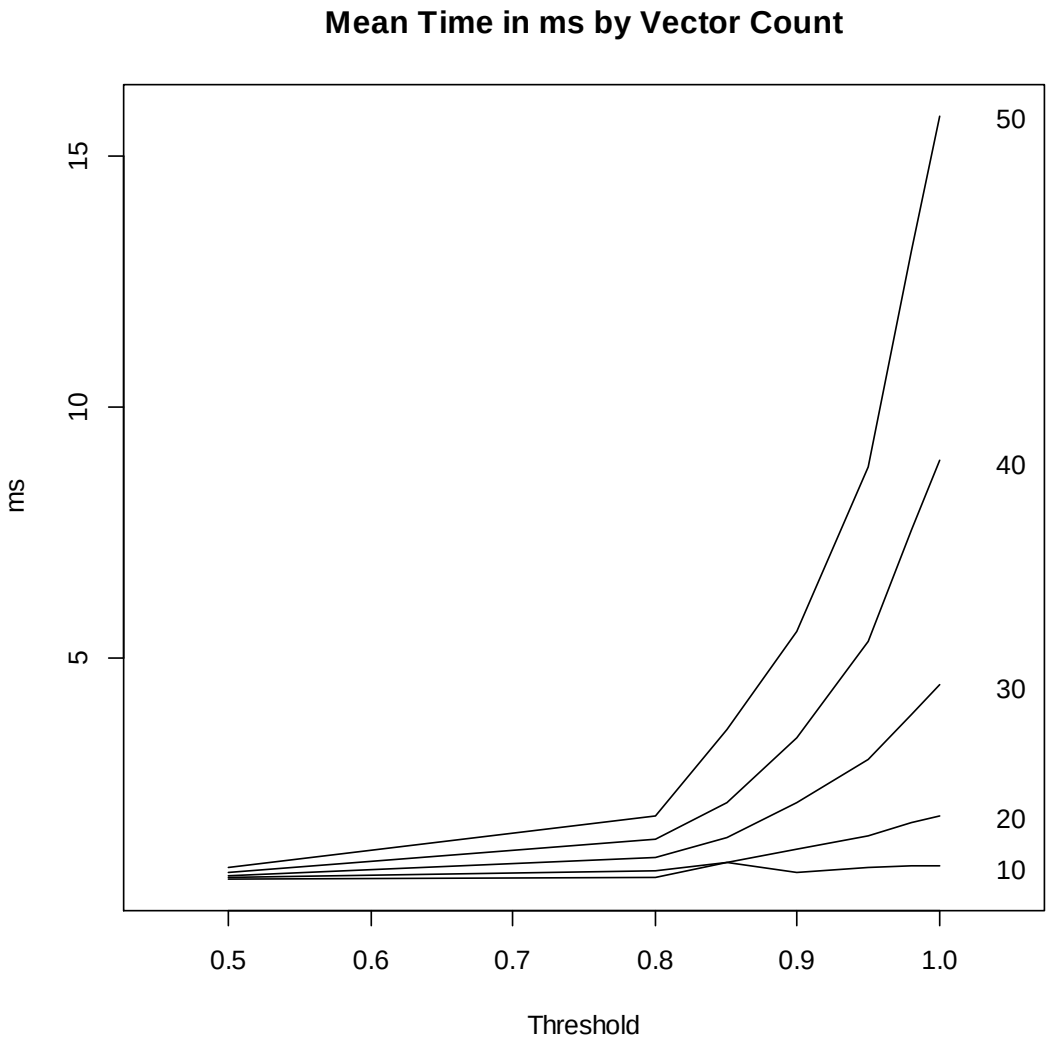



**Figure 19:Performance inpact of weight censoring**

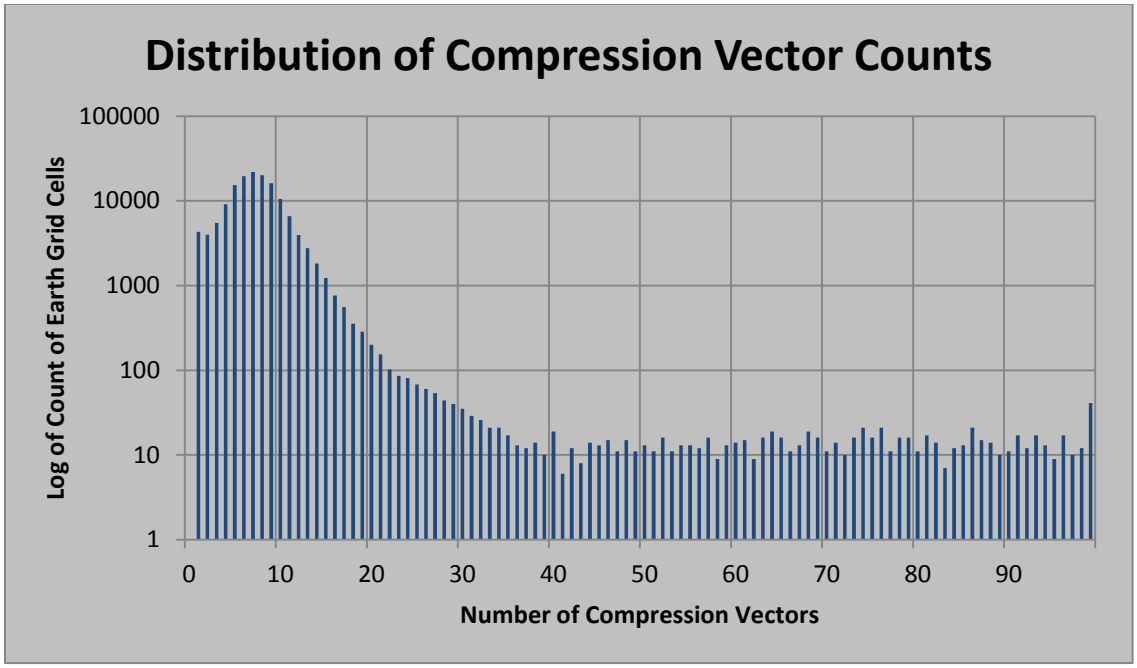



**Figure 20: Distribution of Compression Vectors**

Computationally, there is another question that should be asked – can this calculation be done in single precision rather than double precision? On most computer architectures today this will double the effective throughput on both compute constrained and bandwidth constrained parts of the problem. Where it will not offer any improvement is in activities like pointer chasing, index calculations, and instruction decode & execution. By analogy with Amdahl's Law, the maximum performance gain on offer is less than a factor of two. This would, however, still be attractive. Because of the lack of an optimized single precision linear transport solver library, this is an open topic for further research, and is mentioned here only for completeness.

## 4.2 Other scale reductions

One simple way to reduce the size of the problem would be to replace many earth grid cells with some smaller number of representative cells – in the case of this research, a singleton cell. This would not change the big-O order of the problem – the number of cells in the distance matrix is still proportional to $N_m$, the number of earth grid cells with data in a given dataset -- but could drastically reduce the scale of the constant in front of $N_m$, with commensurate computational benefits. A number of techniques for creating representatives are documented in the literature, and their application to this problem is relatively straightforward, with a few caveats.

The first issue is, of course, choosing which cells will be replaced with a smaller set of cells (or singleton cell). A logical choice is to use an existing clustering algorithm to choose clusters of similar cells to be removed as a set. Another would be to aggregate grid cells on a pre-determined spatial and temporal grid. An example of the first technique would be to use agglomerative clustering, and cut the tree at a level that gives a targeted number of clusters or distance between clusters. An example of the second approach would be to aggregate neighboring 5x5 degree spatial cells into 20x20 degree cells, or to directly to aggregate multiple datasets across the same 5x5 degree spatial cells via restarting ECVQ.

Once a set of cells is selected, it remains to either select or construct a representative cell. If we choose to construct a representative cell, we must decide how many clusters it would contain, their weights, and the data vectors. One possible technique for this would be to apply ECQV again, at the cluster level. We could also

attempt to solve an optimization problem, on some measure of distance or entropy, to find a "center" of the cluster.

I have chosen to select an exemplar earth grid cell from the existing intra-month cluster members. The details of this technique are described in the following sections.

### 4.2.1 Calculation of cluster representatives

Given that I wish to select exemplars from each intra-month cluster to represent that cluster at the next level of analysis, a method of choosing one or more grid cells from the cluster must be determined. A straightforward criterion is to select the grid cell with the minimum total distance to all the other grid cells in the cluster. As the distance measure is a metric, this ensures that the selected grid cell will be "closest" to the distributional center of the cluster. Alternative approaches to selecting a representative earth grid cell could be devised, but this is computational tractable and fits well with my intended use of the representative.

### 4.2.2 Adjustment to the local block distance matrix entries

A complication arises when using exemplars to restart clustering across multiple datasets. Consider the set of points in the figure below.

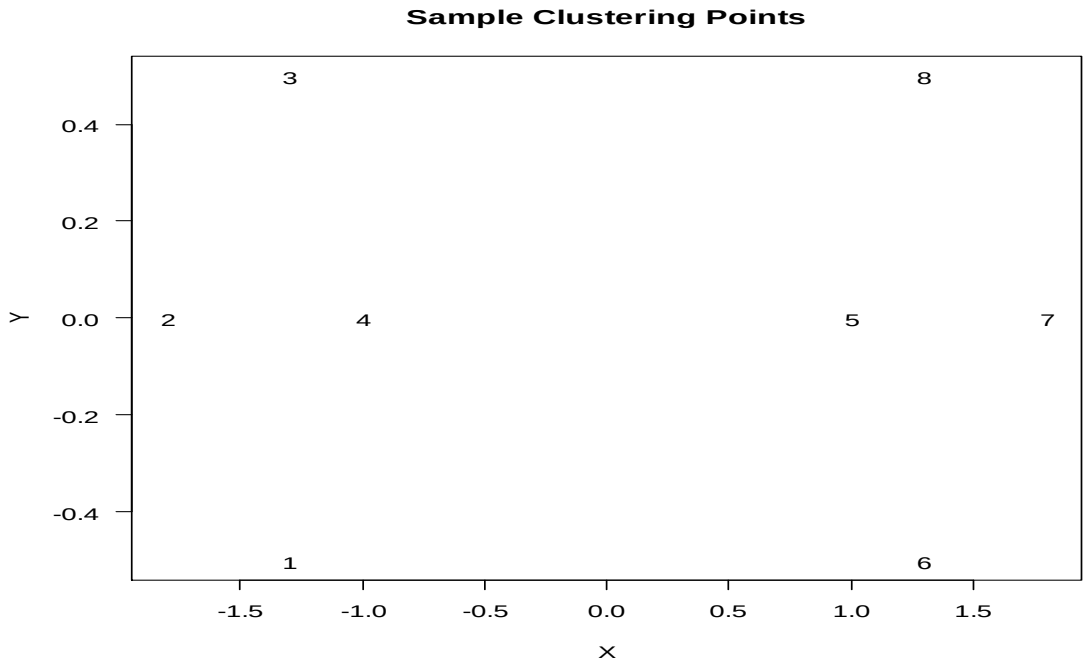



**Figure 21 Sample Points in the X,Y plane**

The points 4 and 5 would be the representatives of two clusters, 1-2-3-4 and 5-6-7-8. Using Nearest Neighbor connections ("single" clustering method in R) these two clusters are joined by 4-5, at an inter-cluster distance of 2. This is show in the clusterting dendrogram below.

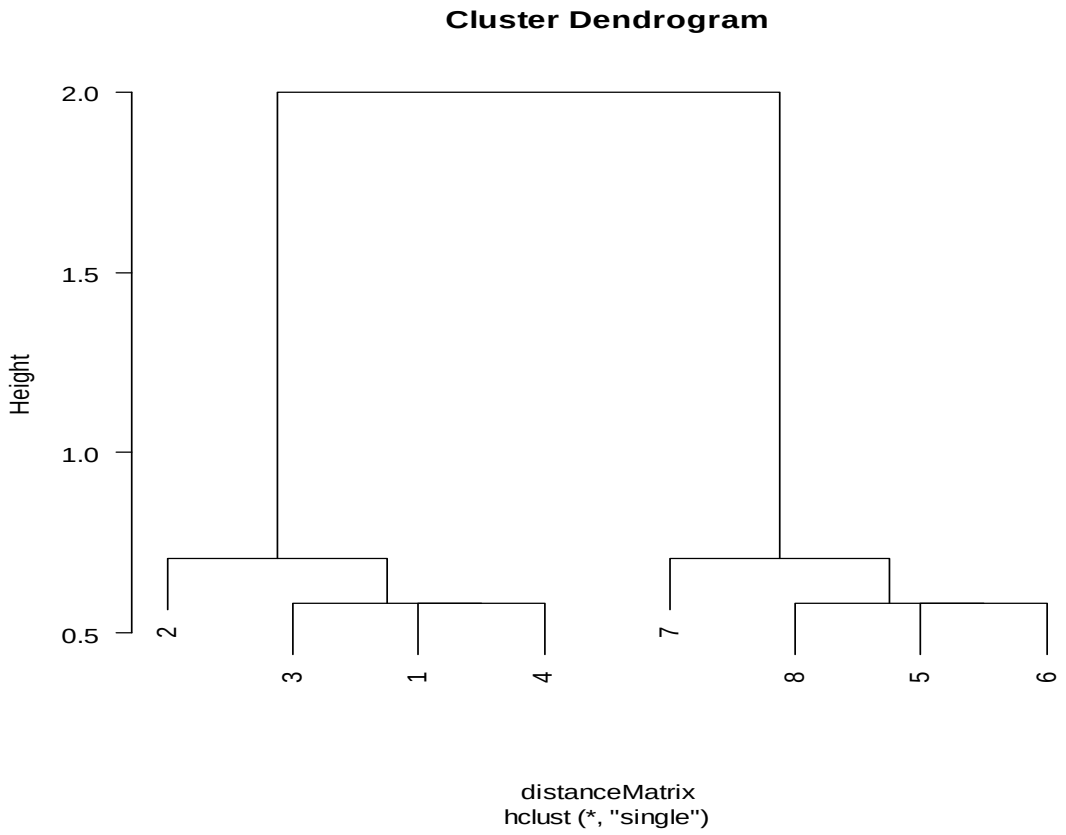



**Figure 22 Nearest Neighbor Hierarchical Clustering**

If we use a clustering technique that uses a more complicated formula for the distance between two clusters (any form of weighted average of the distance between all the points in the clusters, for instance the R "complete" clustering), then the distance between clusters 1-2-3-4 and 5-6-7-8 will be greater than 2. Indeed, in this example that distance is 3.6 .

The dendrogram below illustrates this.

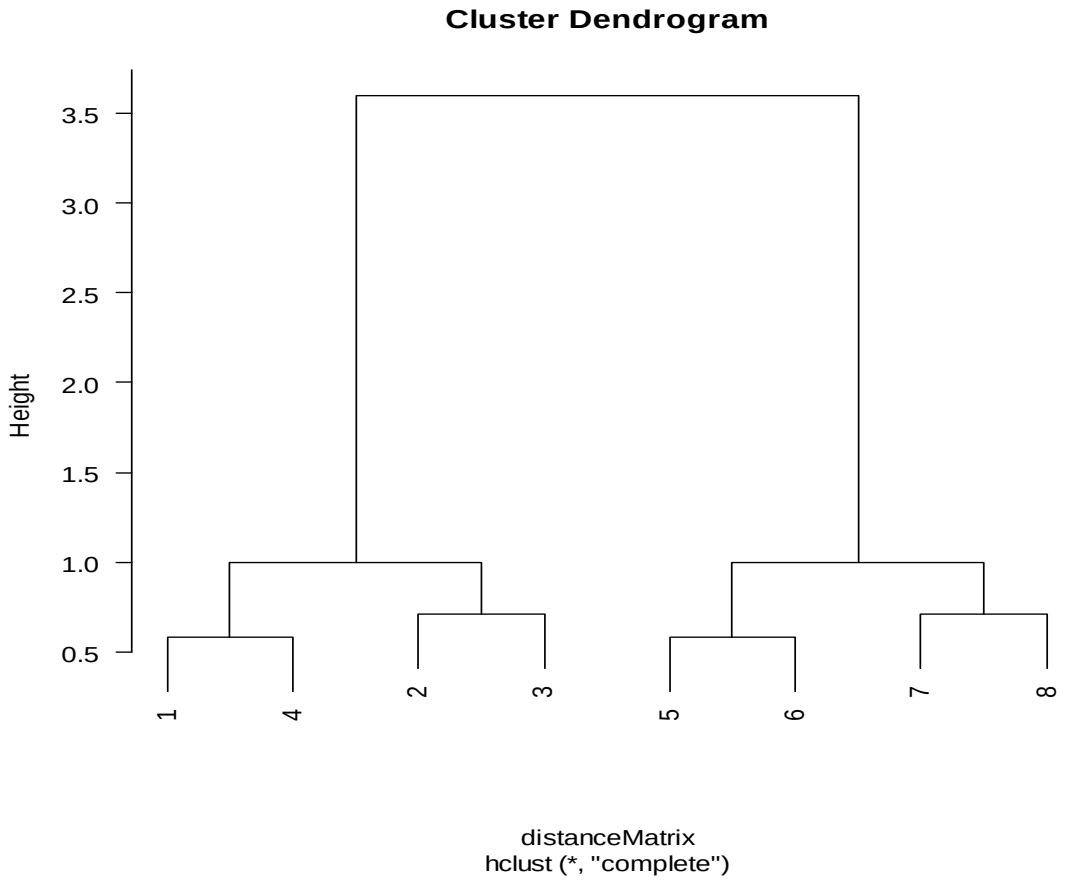



**Figure 23 Complete Hierarchal Cluster**

This means that if we enter points 4,5 into the list of cluster representatives the distance, taken from the local distance matrix in this case, will be recorded as 2. But the clusters they represent do not join until a distance threshold of 3.6. Without an adjustment, once the clustering is restarted, a potentially large number of early merges will be intra-dataset, as there will be many representatives that are closer than their respective hierarchical cluster merge distances. My approach forces these potentially early intra-month cluster merges to be delayed in favor of inter-month merges. I do that

by adjusting the distance matrix to the threshold value used to determine the intra-month cutset.

### 4.2.3 Overall impact and performance of the method

Timings with ever increasing numbers of clusters reveal something about the impact the method has on overall performance of the technique.

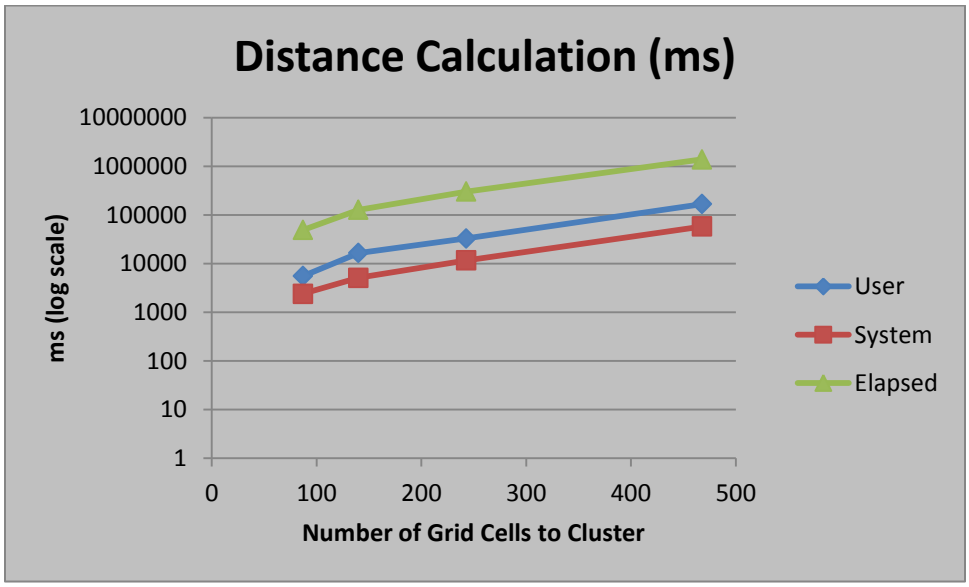



**Figure 24: Distance Matrix Calculation Times**

As this is a linear trend on a log scale, the growth in compute time required clearly demonstrates that attacking raw data (about 2200 clusters per time period) would rapidly become prohibitive. Elapsed time (the one of most concern for interactive users) shows a practical rate of 82 distance calculations per second. At that rate, calculation for

a single month's dataset (a granule) is roughly (2200*2200/2)/82 seconds, or about 8 hours. This would grow proportionate to the square of the number of datasets to be analyzed. Clearly, managing the number of grid cell representatives used is the most direct way to manage the time consumed by the analysis.

Figure 25 illustrates the tradeoff between the cutoff values of D for the hierarchical clustering of sets of earth grid cells, the number of representatives, and the performance and speedup for a study consisting of 3 monthly datasets. The numerical scale is common but the units of each line are different.

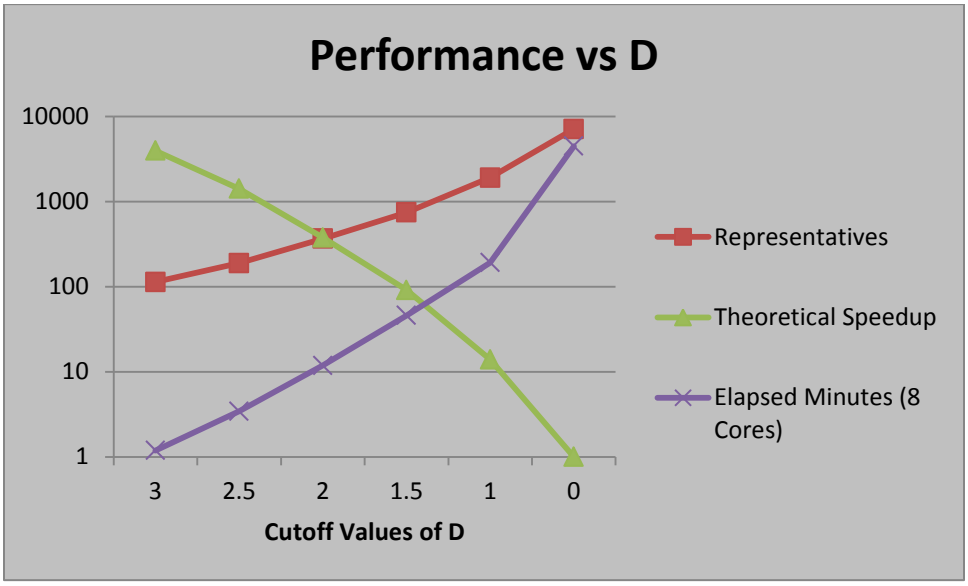



**Figure 25: Performance tradeoffs by hierarchy cutoff value**

### 4.2.4 Impact to clustering (cluster stability)

There are a number of ways to examine the stability of the clusterings produced by this method. One would be to look at the stability of the set of cluster representatives selected.

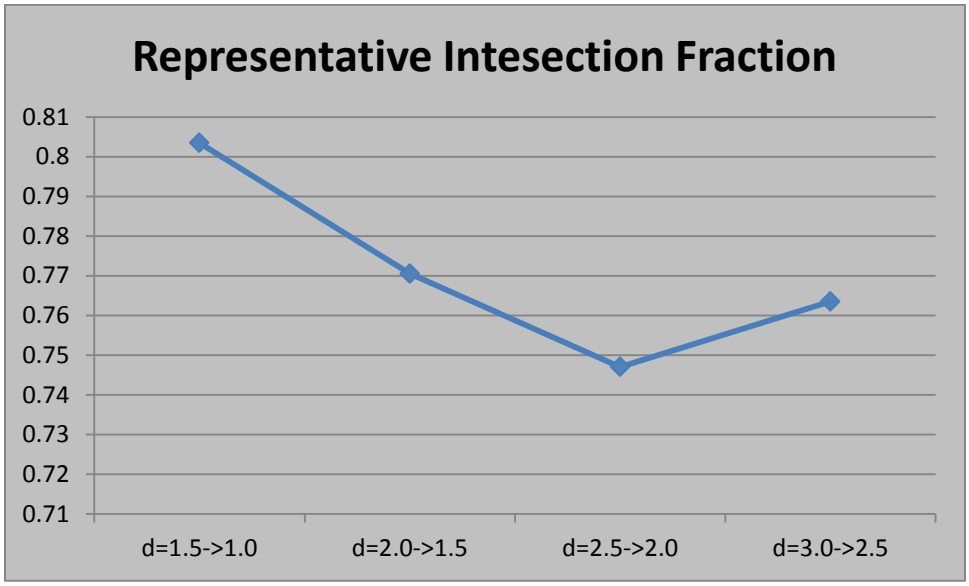



**Figure 26: Cluster Representative Stability**

This chart shows the fraction of the cluster representatives at the first hierarchical cutoff level that are still representatives in the second set. If all the cluster representatives were the same, then the resultant clustering would also be the same. Having the sets of representatives be different, however, is no guarantee that the final clustering will be different.

Visually, we can line the maps up in pairs for a time period and attempt to compare earth grid cell clustering outcomes

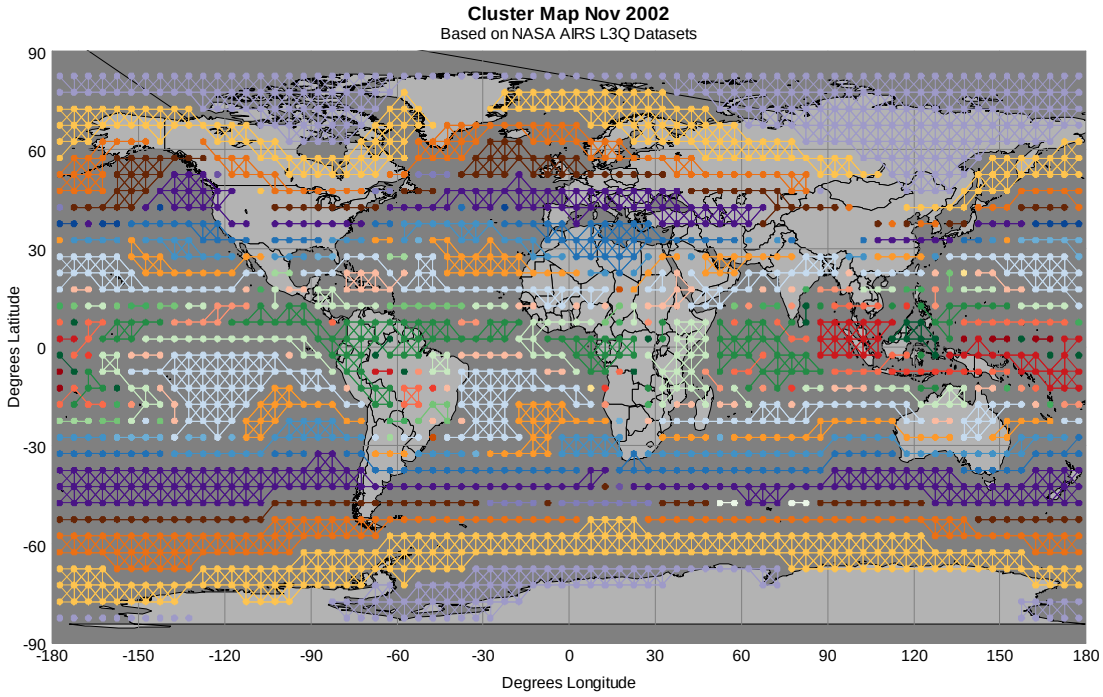



**Figure 27: Clustering with D cutoff = 2.0**

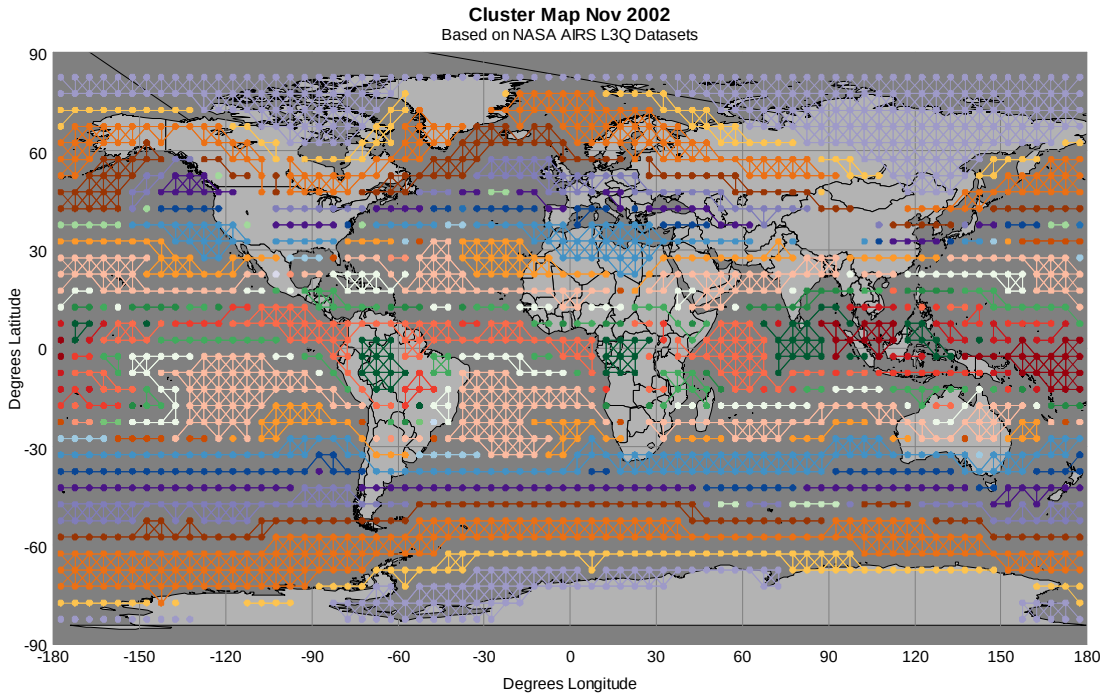



**Figure 28: Clustering with Cutoff Threshold = 1.5**

A number of interesting changes are apparent, although at a broad level the patterns are similar. First, the cluster over the sea and some of the land of the US Pacific Northwest has changed its positioning in the coloring. Also, two of the cells on the southern edge (one on land, one at sea) have left the cluster. The large cluster off the west coast of South America has split in two, while two clusters over the headwaters of the Amazon basin have merged into one.

We can also visualize the difference between two clusterings with a heatmap, generated by looking at which cluster each earth grid cell occupies under each clustering. A clustering comparison of the same 3-month sample dataset clustered at representative

cutoffs of d=1.0 and d=1.5, with 60 heirarchical clusters displayed, is below. The heatmap has entries where given clusters have earth grid cells in common under the two different clusterings. The actual cluster number is an artifact of the analysis although the strong diagonal nature of the plot shows that the general ordering of cluster formation is similar. If a cluster was completely stable across under the pair of clusterings, then there would be only one non-zero entry in a row-column set. Multiple entries indicate different cluster memberships under different representative clustering scenarios.

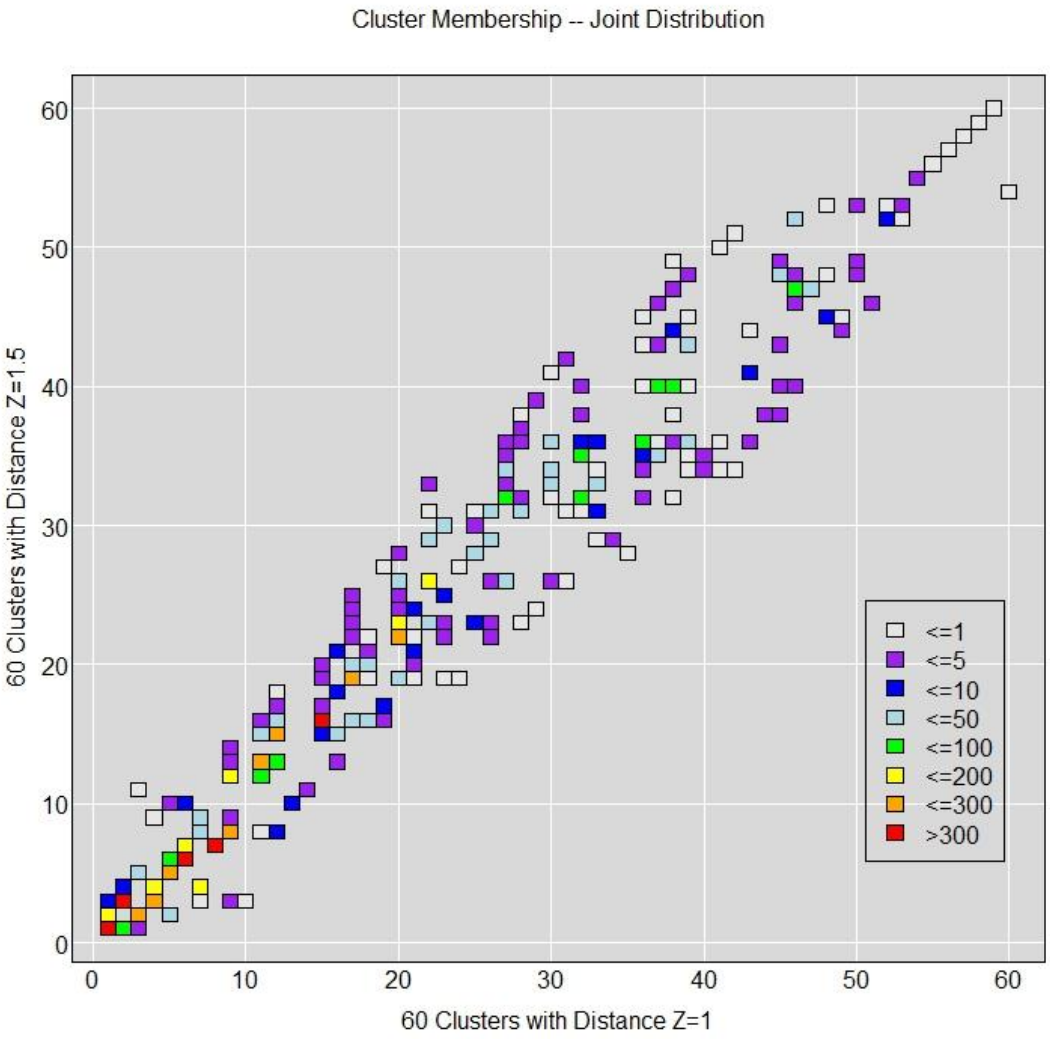



**Figure 29: Cluster Stability Analysis, Joint Distribution**

I conclude that the clusters are not completely stable under the technique as it stands; both numerical and visual evidence confirms this. The general pattern of clusters is however close, and the geophysical data visualization tools highlighted in Section 6

provides evidence that the clusters produced by the method of representatives are supported by the data. Additional studies of the stability of the clustering results based on the representatives chosen would be an interesting topic for future research.

## 5. Displaying Multi-Dataset Clusters

Given that I have clusters of earth grid cells from multiple months, the next step is to attempt to visualize those data clusters to help guide further analysis of the data. To that end I developed a number of techniques and graphics.

Because of the complexity and quantity of the data being explored, techniques to automate coloring of sets of maps covering multiple time periods are discussed in Section 5.3 Palettized Automated Coloring Algorithm. Stability of the resultant colorings in the face of small data changes is discussed in Section 5.3.5 Coloring Stability.

Issues with focusing attention and change blindness led to the development of several focusing graphics, displaying one or a small number of clusters over multiple maps. This also allows the display and analysis of more clusters than my initial palette supports for a single map; it is also more cognitively accessible. The final techniques developed for small multiples of clusters are in Section 5.4.5 Multi-set Hierarchical Earth Grid Clusters. The ultimate product of the single cluster extents over time maps are discussed in Section 5.4.6 Single Set Hierarchical Earth Grid Clusters over Time.

Following the existing literature, the demonstrated maps prior to Section 5.6 Other Datasets cover only a single season. My techniques do allow more than three months to be included in a study, which I demonstrate in Section 5.6.

Finally, in Section 5.7 Computational Complexity of the Method, I review the time required for the different phases involved in computing the initial clustering and the production of initial cluster maps.

## 5.1 Sets of maps over time

Existing studies of the AIRS L3Q data predominantly focus on either a single data granule or pre-merge existing data granules into a single dataset which can then be geospatially visualized. In this research, the data from each existing data granule is preserved and presented for visualization, resulting in a natural "map per dataset". In such a situation, showing cluster membership across time (separate images) presents challenges. Here we use consistency of color to indicate cluster membership – the same color represents the same cluster across time.

## 5.2 Map coloring strategies

Commonly, cluster membership of spatial areas on a map is indicated by color. Random assignment of colors to clusters would, by definition carry minimal informational value beyond distinguishing clusters. Some method to assign colors is needed if we want to add additional information.

Assignment of colors to clusters based on a value in the data (i.e. mean cluster temperature at sea level) against a scale of chosen colors is extremely common. While this is simple, and quick, it generally requires a graduated scale to make any sort of sense – which either limits the colors to a very small number or makes it challenging to discriminate between clusters which are near the same data values. This method does, however, have the advantage of visually presenting some information about the underlying clusters.

Assignment by some meta-data associated to the data (i.e. number of cluster members or average latitude) against a scale of chosen colors is also simple. Unless cluster size is particularly scientifically interesting, however, this will likely add no comprehensibility to the map.

Hand coloring (direct manual assignment of colors to clusters) can produce maps that highlight certain areas of interest, provide good ability to discriminate between neighboring clusters, and can be related to the science of interest. This requires potentially multiple iterations and manual intervention to produce each map, which is time that could have been spent examining the clusters for scientific insight.

This research leverages existing expectations from topographic relief maps with a meta-data driven approach to automatically color cluster mappings using palettes that have proven discriminatory power. While perhaps not generating the same ease of insight that a well colored, manually generated set of maps would, by using similar heuristics to those chosen by topographical mapmakers the results appear to be reasonably useful. This could be considered a meta-data driven coloring, except that in this case distinct clusters with the same meta-data value on the coloring scale are not colored together.

## 5.3 Palettized Automated Coloring Algorithm

The Palettized Automated Coloring Algorithm (PACA) was first described in [Carr, Ashley 2011] as the Initial Coloring Algorithm. The algorithm relies on:

- multiple pre-chosen palettes of ordered colors,

- a meta-data driven function that maps cluster members to preferred palettes and preferred positions within those palettes,
- an aggregation method for those preference mappings, and
- an assignment procedure for mapping cluster aggregate preferences to colors.

While the algorithm can be applied to any data driven investigation of clusters where more than one single pre-chosen palette would be useful, this research implements only the specialized case needed to provide automated coloring of the generated AIRS L3Q clusters.

### 5.3.1 Palettes

In this research, two palettes were selected – Land and Sea. These two palettes were defined for two reasons:

- climate science dynamics are different over large areas of water versus land and
- these choices let us take advantage of likely prior user associations of colors.

The second factor leverages the common topographical convention of using graduated blue colors for water depth, and greens and browns to color for some combination of height and aridity.

A given cell has a preference for being colored using a color from the Land palette derived from the percentage of land data in the 5x5 grid cell. Cells near the poles that didn't have data were arbitrarily assigned a preference for Land. While these preferences are stable over (non-geologic) time, the algorithm itself does not impose any requirement that the preference function not change over time.

### 5.3.2 Palette Colors

Multiple Colorbrewer monochromatic color sets were combined to produce the two palettes. Sets of colors from Colorbrewer [Brewer 2002] are chosen to be easily distinguishable and take into account common colorblindnesses and printability. They have been developed to give equal perceptual steps between colors.

The Sea palette uses reds, blues and purples to form a palette running from dark red to light red, then dark blue to light blue, then dark purple to light purple. This follows a familiar red->warm, blue->cold color mapping found in many environments (bathroom taps, for example) and also the blue->water mapping found in many topographic maps that include surface water features.

The Land palette uses browns and greens to form a palette running from light brown to dark brown, then light green to dark green. This will leverage a very common theme in topographic maps, using browns to denote arid land areas and greens to denote more fertile ones.

Below, in Figure 30: Sea Palette, the SST data underlying the palette preference function is displayed in the palette colors – this data is external to the AIRS L3Q dataset. The dataset used to form the basis for the palette preference function is derived from NOAA Optimum Interpolation Sea Surface Temperature V2 data products. NOAA_OI_SST_V2 data was provided by the NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, from their Web site at http://www.esrl.noaa.gov/psd/.

In Figure 31: Land Palette, the Normalized Difference Vegetation Index (NDVI) data is colored using the land palette colors – this dataset is also external to the AIRS

L3Q dataset. It is derived from Moderate Resolution Imaging Spectroradiometer data on NASA's Terra satellite. MODIS data is available at www.landcover.org.

The only restriction from the method is that there be some way to map an earth grid cell to a preference for each palette and for a position on that palette. Note that some sea regions with islands and coastal regions will have preferences on both palettes. The methodology does not require that an individual cluster member have only a single preference. Mathematically, the preference function returns a palette preference vector, not a scalar.

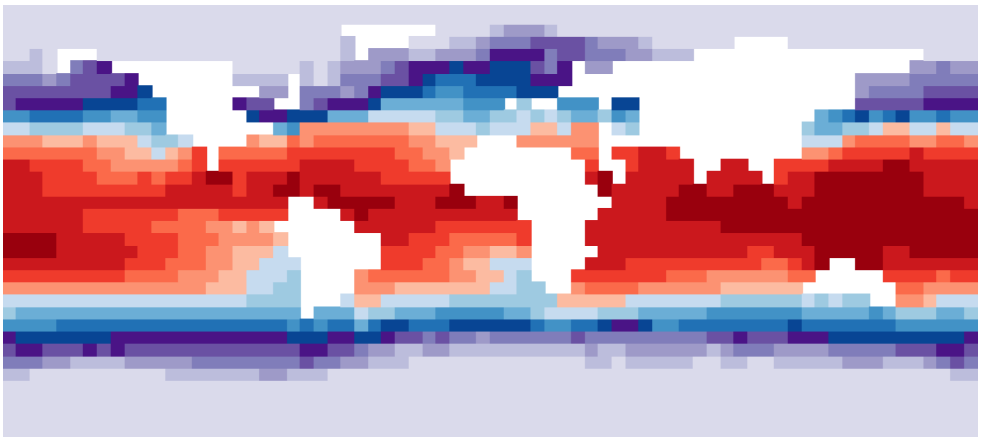


**Figure 30: Sea Palette**

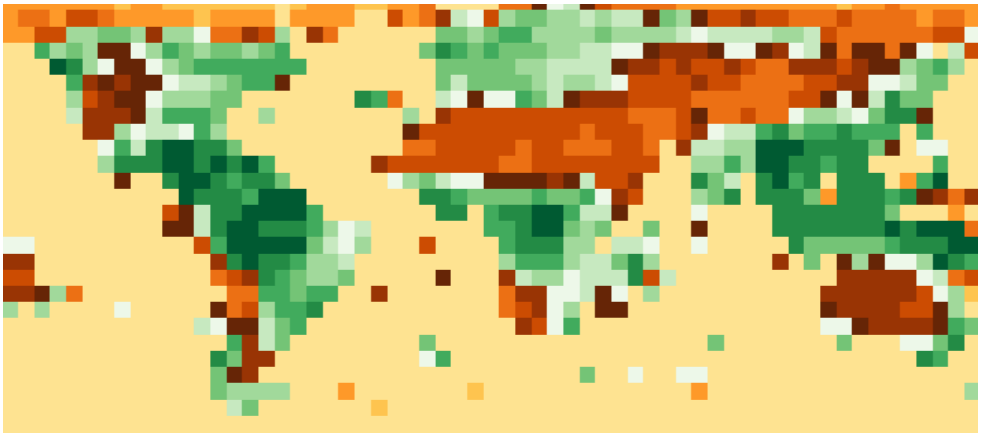


**Figure 31: Land Palette**

### 5.3.3 Palette Position Preference

The palette position preference function uses additional meta-data derived from geospatial cell location. Specifically, palette preference position for the Sea palette is derived from mean sea surface temperatures. Palette preference positions for the Land palette are derived from NDVI greenness index data. While in the research code the function used is not time varying (it is based on a particular month of SST and NDVI data) there is no reason this cannot be time varying. Nor, obviously, does the function need to be the same for different palettes – it is not in this case. It merely must be calculable for any cluster member and generate a preference for a ranking on each palette.

Palette position preference is aggregated by taking the mean palette position preference of all the cluster members and rank ordering them. This rank ordering is then used with the ranked colors on the palettes to assign colors to clusters.

### 5.3.4 Pseudo-code

The actual R code to implement the algorithm described above takes a fair bit of space, but conceptually, the pseudo-code is straightforward.

```
# Aggregate the individual cluster member preferences to form cluster preferences
FOR EACH cluster
        FOR EACH member
                Calculate the palette and position preferences for the member
                Aggregate that to the cluster
# Assign clusters to palettes
# Number of clusters == total number of palette colors
landPaletteAssignment = LIST of length landPalColors with empty elements
seaPaletteAssignement = LIST of length seaPalColors with empty elements

landOrder               = LIST of all clusters sorted by landPalPref
seaOrder                = LIST of all clusters sorted by seaPalPref

WHILE NOT DONE
        Alternate current palette for assignment
        IF current palette is not full
                Assign next unassigned cluster from current order to that palette

landPalettePrefOrder = list clusters in landPaletteAssignment
                        sorted by land palette position preference
seaPalettePrefOrder = list clusters in seaPaletteAssignment
                        sorted by sea palette position preference

Assign colors to clusters in PalettePrefOrder
```

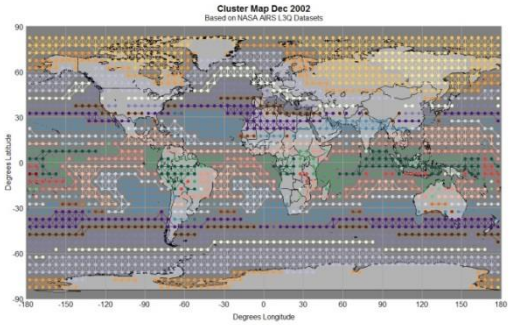
To extend the algorithm to any arbitrary number of palettes, assume an array of lists of PaletteAssignments, PaletteOrders, and PalettePrefOrders. Then instead of alternating the current preference advance it, and treat the array as a ring buffer.

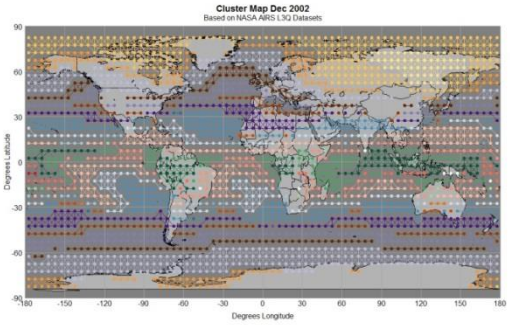
Other assignment algorithms are certainly possible; comparing the colorings from different assignment algorithms would make an interesting future study.

### 5.3.5 Coloring Stability

By the nature of the algorithm, changing the number of colors available or changing the number of clusters being studied can change the assigned coloring. In the following examples, the underlying data is the same for both sets of maps; the only difference is the number of target clusters being studied is 32 for the maps on the left and 31 for the maps on the right. The study set is three months of data from Dec 2004 to Feb 2005.

Because the only difference in the two studies is the number of target clusters, effectively, the only underlying difference in the two clusterings is that in the second study, two of the clusters from the first study are combined into a single cluster. This is due to the nature of the hierarchical clustering. This means that 30 of the 31 clusters in the second study are identical to the clusters in the first study.



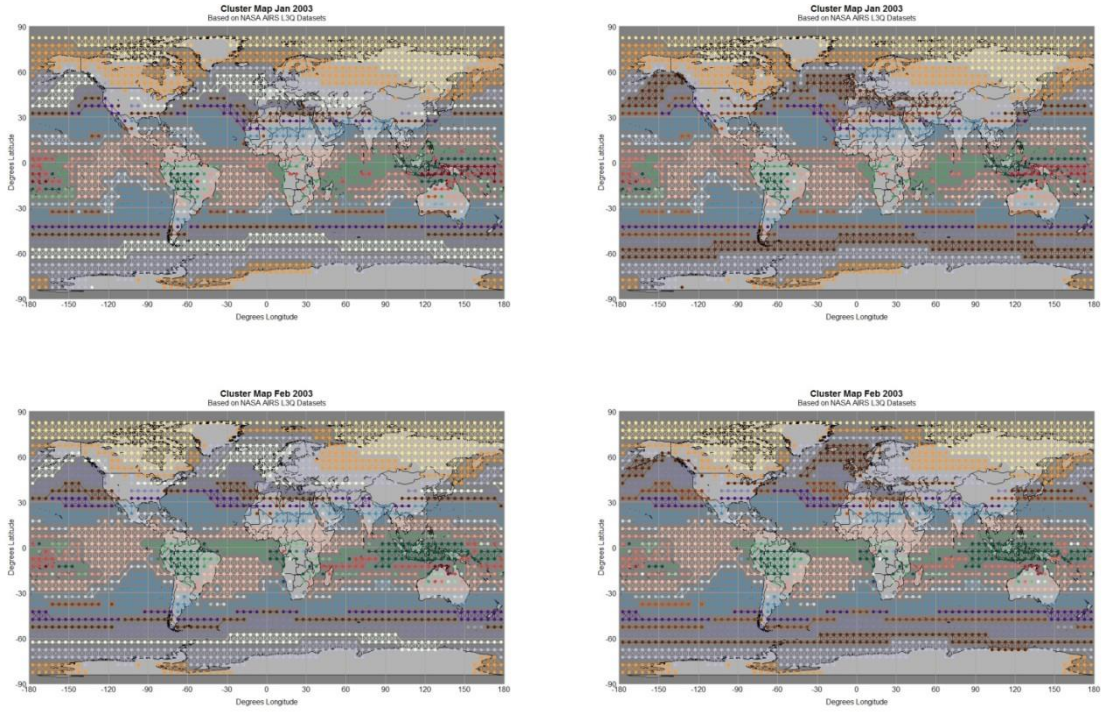



**Figure 32: 32 Colors, Left; 31 Colors, Right**

In the figure above, while the colorings across northern Europe and Russia appear to be the same in both studies, the area off the west coast of South America highlights the difference in coloring caused by clusters changing relative position in the palette preference and palette position preference rankings.

## 5.4 Multi-Dataset Examples

The basis for all these maps is a data structure that defines a layer per dataset, indexed by latitude and longitude $5^{o}$x$5^{o}$ grid cell numbers. In a nod to the related work referenced in Section 3, a variety of studies were conducted on "Winter" datasets. I will refer to the set of data {December 2002, January 2003, February 2003} as Winter 2002

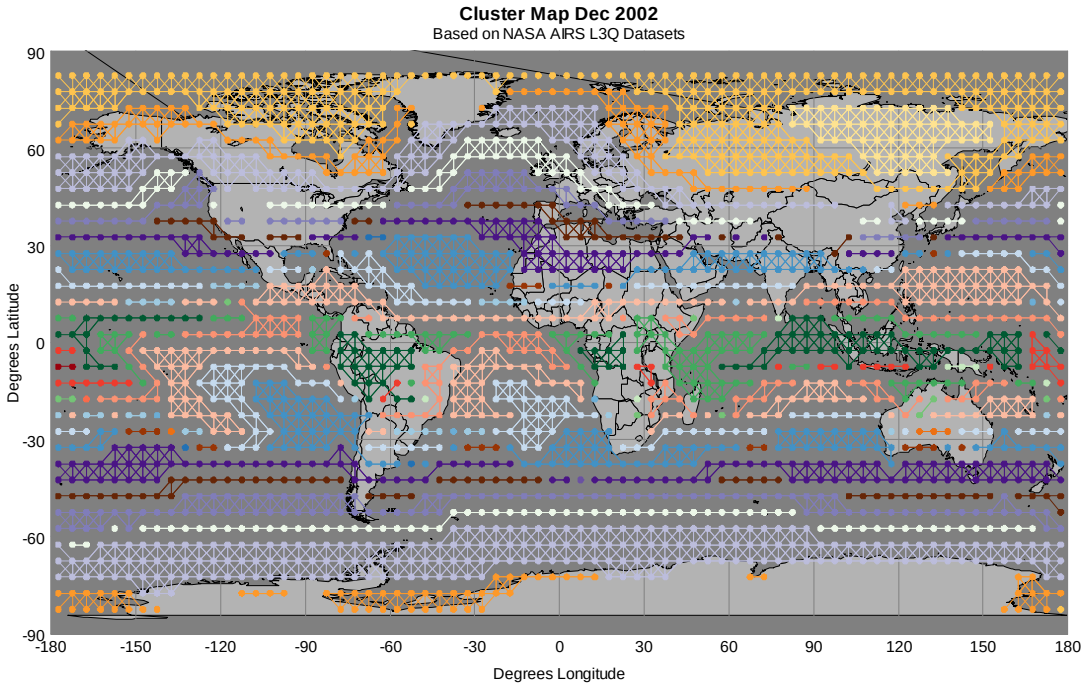
data. Unlike [Zhou, Shi 2011], who study the single month December 2002, here we analyze and visualize data from all of Winter 2002.

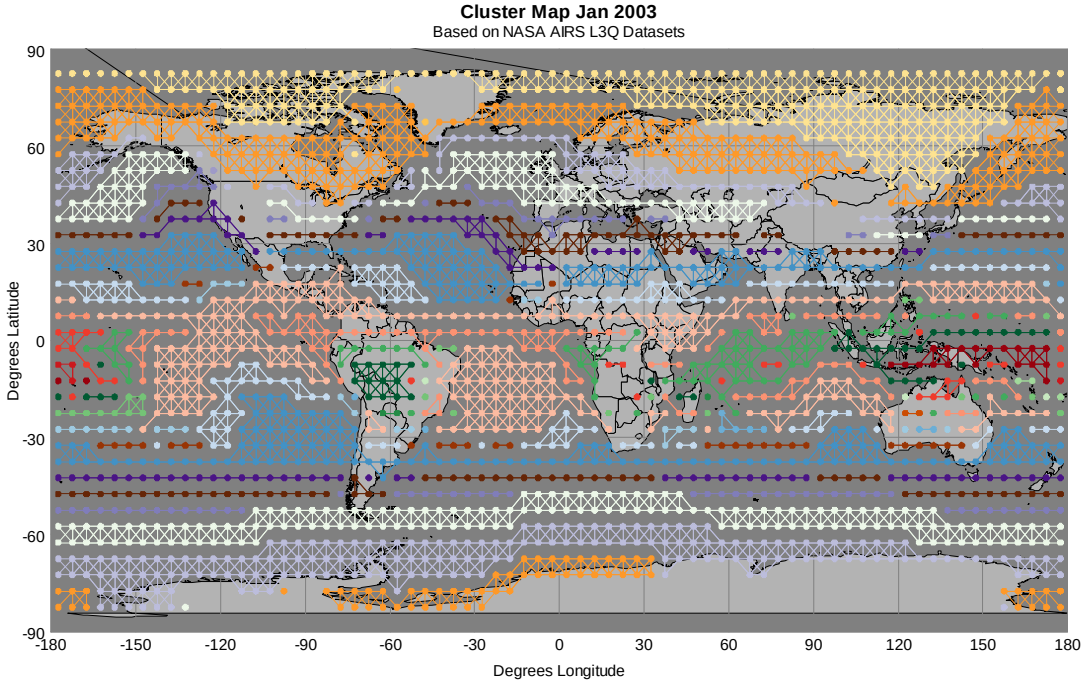
Unlike [Braverman et al, 2012b], who combine datasets into a single dataset before visualization and study, I maintain each month as a separate and analyzable component of the overall seasonal structure.

### 5.4.1 Full Data All Clusters

The following three images show the output of the initial color algorithm applied to the Winter 2002 data after processing. The process parameters set a representative cutoff threshold of 3, producing 148 representative clusters. The analysis and initial mapping required about 83 seconds of elapsed time on a Intel Core i7 laptop with 4 cores at 2.7GHz, and having 8GB of RAM. More detail on the timings is found later in 5.6 Other Datasets.

There are a number of differences with the images in the related work. The most obvious is that there are three maps, each with $5^o$ detail. The colored cells are not filled completely with squares, but rather with circles, which, if they are adjacent to another cell in the same cluster, are connected by line segments as well. Background in squares is lighter for primarily land grid cells and darker for water.


Cluster Map Dec 2002
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude
90
60
30
0
-30
-60
-90
-180
-150
-120
-90
-60
-30
0
30
60
90
120
150
180


Cluster Map Jan 2003
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude
90
60
30
0
-30
-60
-90
-180
-150
-120
-90
-60
-30
0
30
60
90
120
150
180

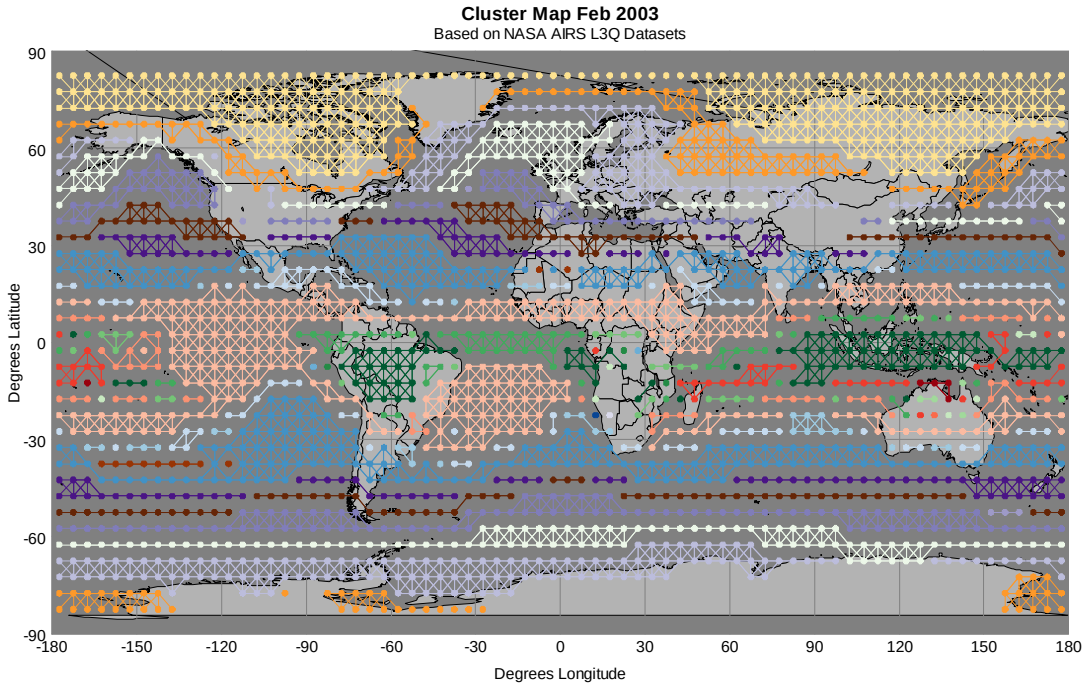



**Figure 33:Dec 02 , Jan 03, Feb 03 ( 143 Reps, 32 Colors )**

Further, as in [Carr, Braverman 2007] the data is clustered in the 18 dimensional principal components space, with assigned colors per cluster, rather than being colored on a single numeric scale associated with the value of one dimension from a multi-dimensional scaling of either the physical [Zhou, Shi 2011] or principal components space [Braverman et al, 2012b].

It also depends on how much data is being examined, across what boundaries. Is Winter the right timeframe, or is the process more complex than that? For example, seasonal summaries would miss similarities between Northern Hemisphere

temperate spring patterns and Southern Hemisphere temperate spring patterns, as they would have been converted into a Spring and a Fall dataset (assuming a researcher with a Northern Hemisphere bias). Examining the more granular monthly data would allow those patterns to be revealed.

### 5.4.2 Full Data with Focus Clusters

Selection of certain clusters for additional focus (and making them stand out more easily across images) could be accomplished many ways. In this research, expanding the size of the colored dots for in-focus items and reducing the size of the dots for out of focus items was selected. Other choices are, of course, possible. In the following single image from the Winter 2002 study case, a swath of four clusters taken vertically through North America east of the Rocky Mountains and due south just over the border into Mexico is selected. Each of these four cells is in a separate cluster.

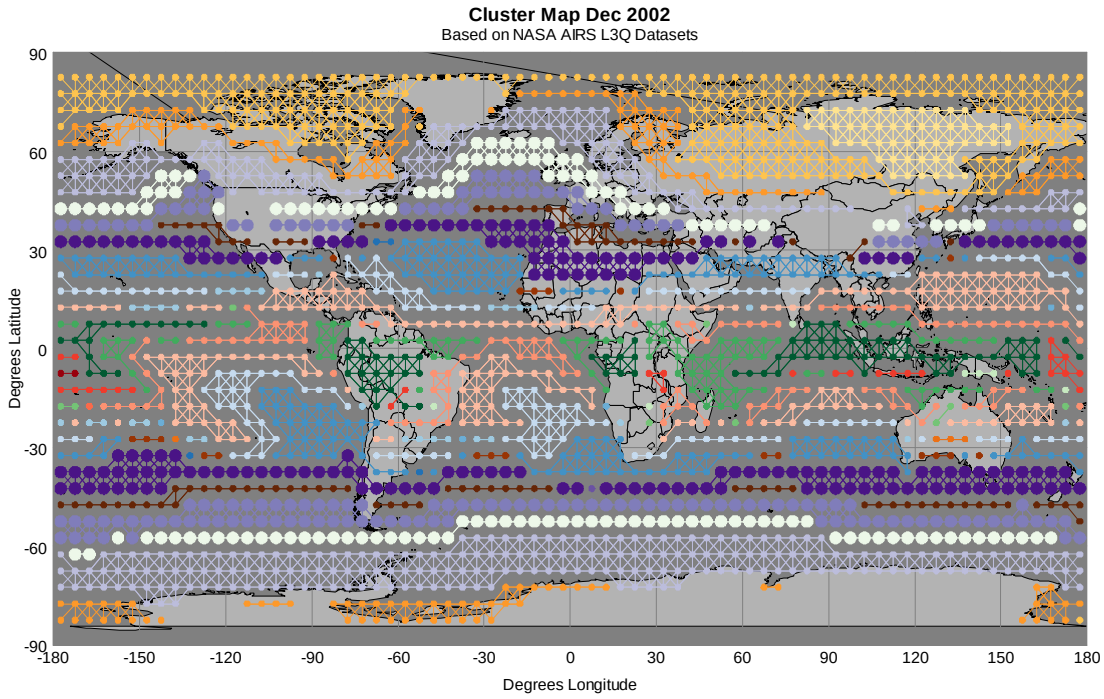



**Figure 34: Winter 2002 Study, Dec 2002 Data with Focus Clusters**

These focused data clusters are also highlighted in the other two months at the same time, as would be expected.

### 5.4.3 Full Data Single Cluster over Time

A cluster of grid cells corresponds to distribution of multivariate multi-altitude atmospheric descriptors. The purpose of clustering earth grid cells over time is to show the shifting in location and extent of such relatively stable atmospheric distributions. The shifting of extents means that the distribution can disappear and possibly re-emerge later. Focusing attention on a single cluster (distribution ) over time helps us to see the shifts in spatial location and temporal extent

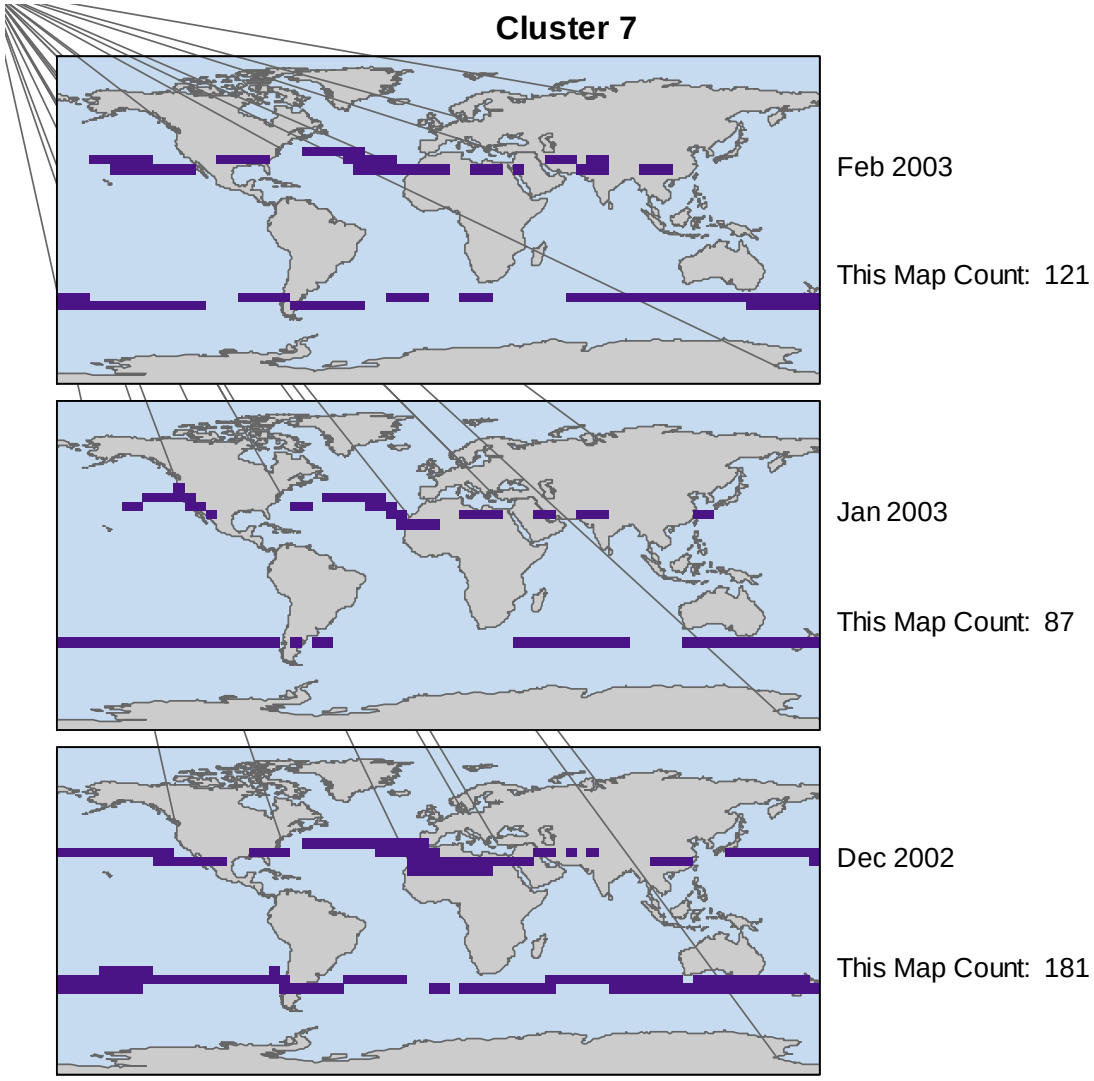



**Figure 35: Winter 2002, Cluster 7**

. This can be represented using animation or juxtaposed views, but both approaches have their deficiencies [Ware 2013] . While we can see changes in animation where our eyes are focused and changes can draw our attention, we cannot

simultaneously focus on and think about multiple location in a map. Further grid cells that remain a part of the same cluster over time may draw no attention and not be noticed. With animation, out of sight is quickly out of mind. New images fill our retinas.

Juxtaposed views provide time to observe and think but suffer from the problem of change blindness. When our eyes move in rapid motions called saccades from one focal point to another we are effectively blind. Our visual change detectors are at rest. We only retain a little area of focal attention in mind long enough to make comparisons between corresponding location in two juxtaposed maps. Detailed comparison of juxtaposed images requires back and forth scrutiny of small areas.

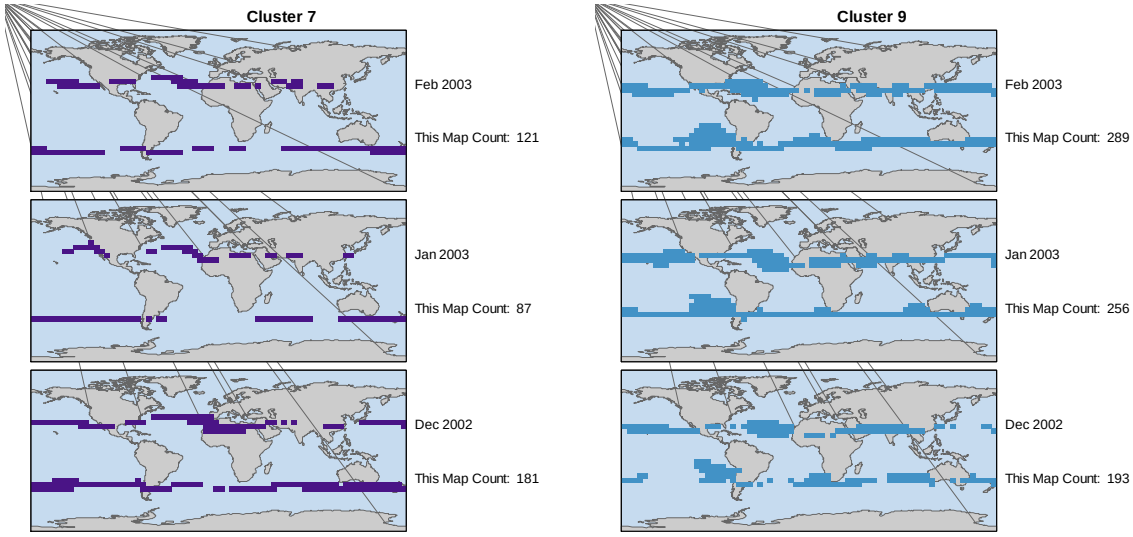



**Figure 36: Winter 2002, Multiple Clusters**

Figure 35 shows just one cluster to remove the distraction of other clusters. We quickly notice rough similarities over the three months, but detailed comparison is difficult.

Figure 36 juxtaposes three month plots for Clusters 7 and 9. It seems the groups of grid cells for the two clusters are contiguous for the same month, but detailed assessment is difficult. If we want to know where the clusters are and are not contiguous, it is better to superpose clusters in the same plots..

### 5.4.4 Masked Data

Given a limited number of colors, and the possibility of larger and larger datasets, I wanted a way to leverage the technique to allow more focus on individual areas and use the colorings to provide greater detail to an area of interest.

I employ a mask to do this, where only clusters having at least one cell in the region defined by the mask are allowed to occupy a palette. This allows me to define study regions such as the zone from $30^{o}$S to $30^{o}$N.

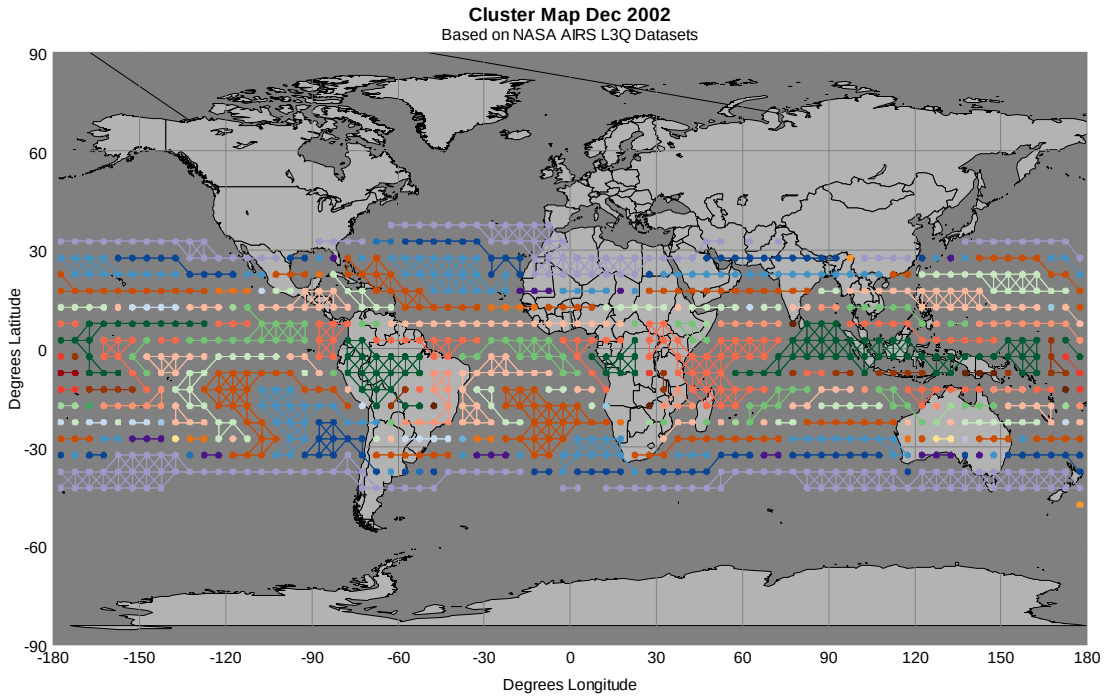



**Figure 37: December 2002 30°S to 30°N**

Because I include cells outside the study area that are members of the clusters within the study area, the edge of the study area is not a hard boundary. I chose this approach as it seemed like seeing all cells similar to the cells in the study area would be more valuable than not knowing which if any other earth grid cells are included in the clusters of interest.

### 5.4.5 Multi-set Hierarchical Earth Grid Clusters

One potential objection to the overall cluster maps over time produced by the PACA is that there are too many different clusters shown on the same set of maps, leading to an overwhelming amount of information on the graphics. Also, being forced to

display all the clusters in a single graphic implies that the total number of clusters in the study cannot exceed the number of clusters in the palettes being used for coloring.

The multi-set charts below address both these concerns. They display a small number of hierarchical clusters over time using multiple maps. They use an altered version of the Palettized Automated Coloring Algorithm locally, on a reduced color palette. This does allow the overall dataset to be broken into any arbitrary number of hierarchical clusters; overall display of the study set simply requires multiple multi-set maps.

The PACA was altered as in this case, I always expect to have as many palette positions for both the land and sea palettes as there are clusters. So, instead of alternatively taking the cluster with the greatest preference for each palette until all palette positions are filled or I run out of clusters, I allow the clusters to take a position on whichever palette they most prefer.

The graphic below shows multiple clusters, any small subset can be displayed.

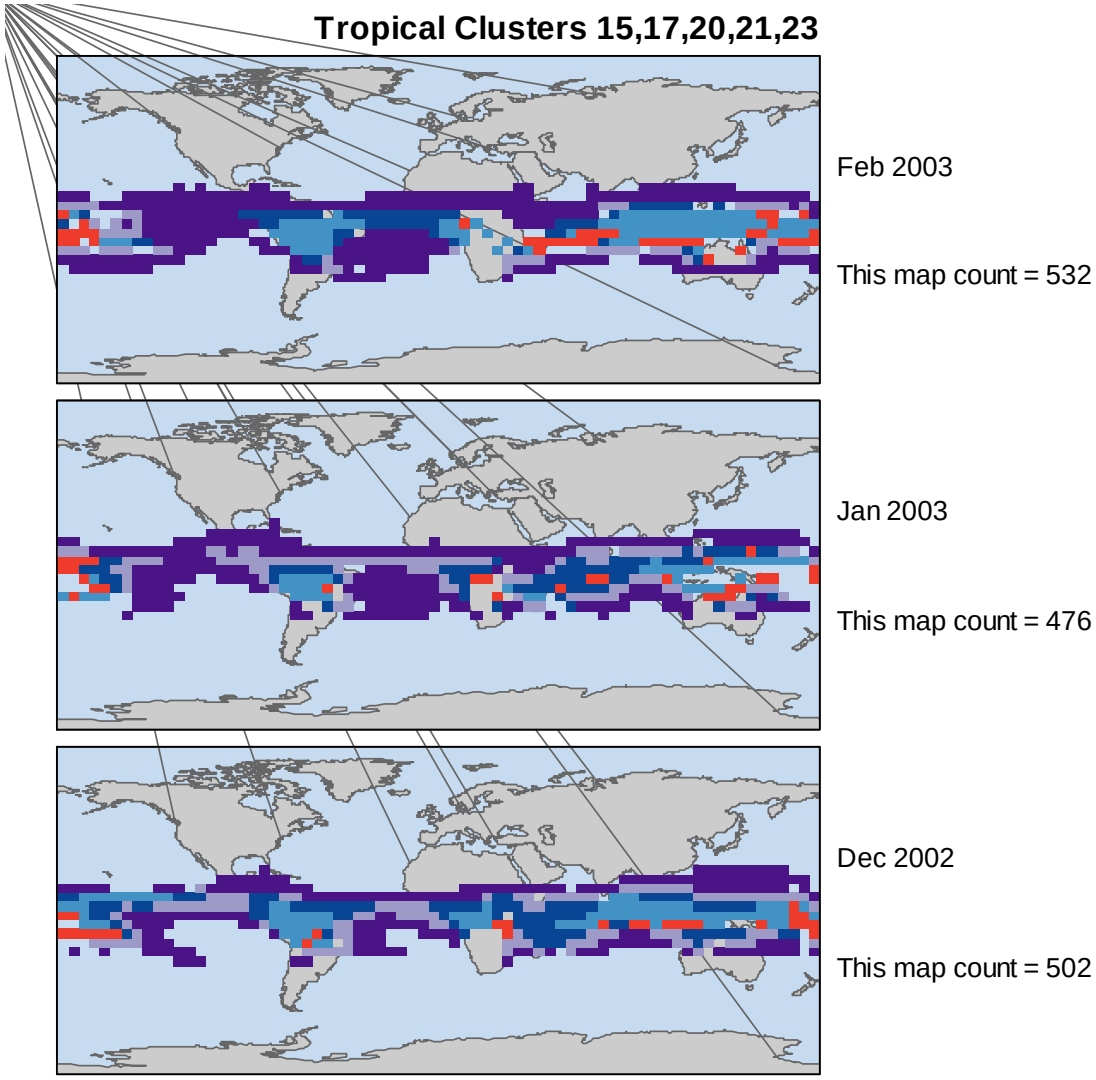



**Figure 38: (Small) Multiple Clusters over Time**

### 5.4.6 Single Set Hierarchical Earth Grid Clusters over Time

This graphic helps visualize the evolution of a single hierarchical cluster over time. Each pane of the map corresponds to a different underlying dataset. In each pane, the union of earth grid cells that ever are members of the cluster are outlined in light

green; those that are always part of the cluster are filled in dark green, and those that are members in that dataset but not all the datasets are colored medium green.

Effectively, the union of the earth grid cells can be thought of as a high water mark for the cluster's geographical extent. Using a different outline for various extents through time is a common technique for historical maps and I use the outlines to similar effect. There will always be at least one cell in the high water mark, and the number of cells with outlines will be the same in all panes, although currently "active" cells in the cluster will over plot the outlines.

The cells that are always members of the cluster footprint, should there be any, can be thought of as the invariant core of the cluster geographically. They are repeated in every pane, and cannot be overplotted.

Finally, any cells beyond the invariant core are shown in the medium green. These will change, pane by pane. Also, by definition, any cell that appears in solid medium green will not be part of the cluster in some dataset (or it would be a part of the invariant core of the cluster).

These features make the changes over time more apparent. Given that the high water marks would overlap, the technique is difficult to extend to multiple clusters unless the outline for the high water marks of the two clusters was changed to some sort of striped fill. A convention for overlaying high water marks and current cluster members would need to be established as well.

I then extended the technique by using two colors for the high water mark – one for cells which, relative to the current pane, were part of the cluster in the past and one

for cells which, relative to the current pane, are going to be part of the cluster in the future. Red for the past and blue for the future seemed sensible, in effect a “Doppler” palette, with items receding from the current point of view colored red, and those coming towards the current point of view being blue. Additional graphical enhancements were also incorporated ( Figure 39: The Final Single Set over Time Maps).

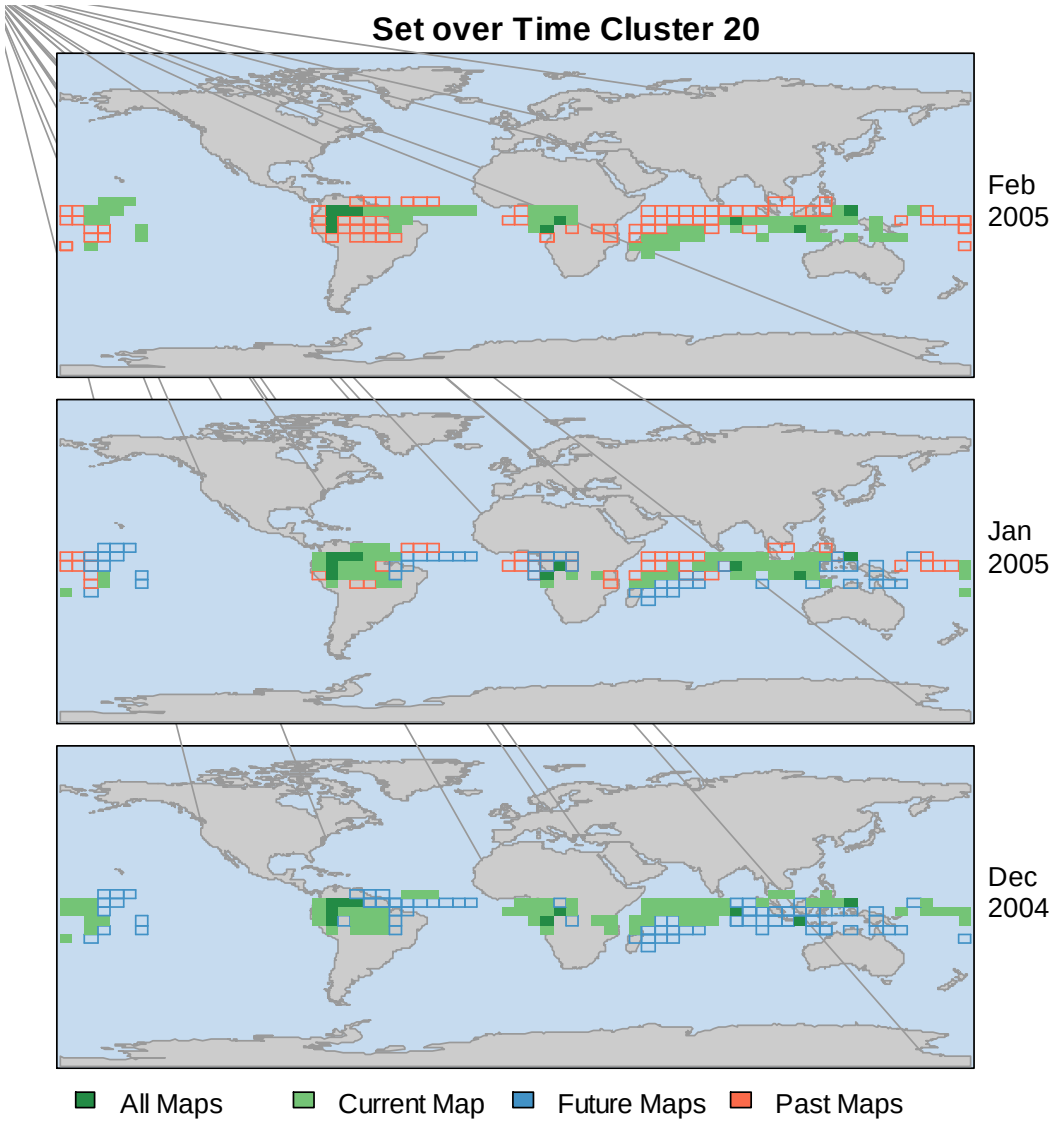



**Figure 39: The Final Single Set over Time Maps**

## 5.5 Study Areas with reduced coloring

Certainly, some study areas may not benefit from using the full palette I've defined of 32 colors. It is possible to reduce the number of colors being used which

simply reduces the number of clusters shown. This can be combined with the masking features above to produce analysis of smaller areas.

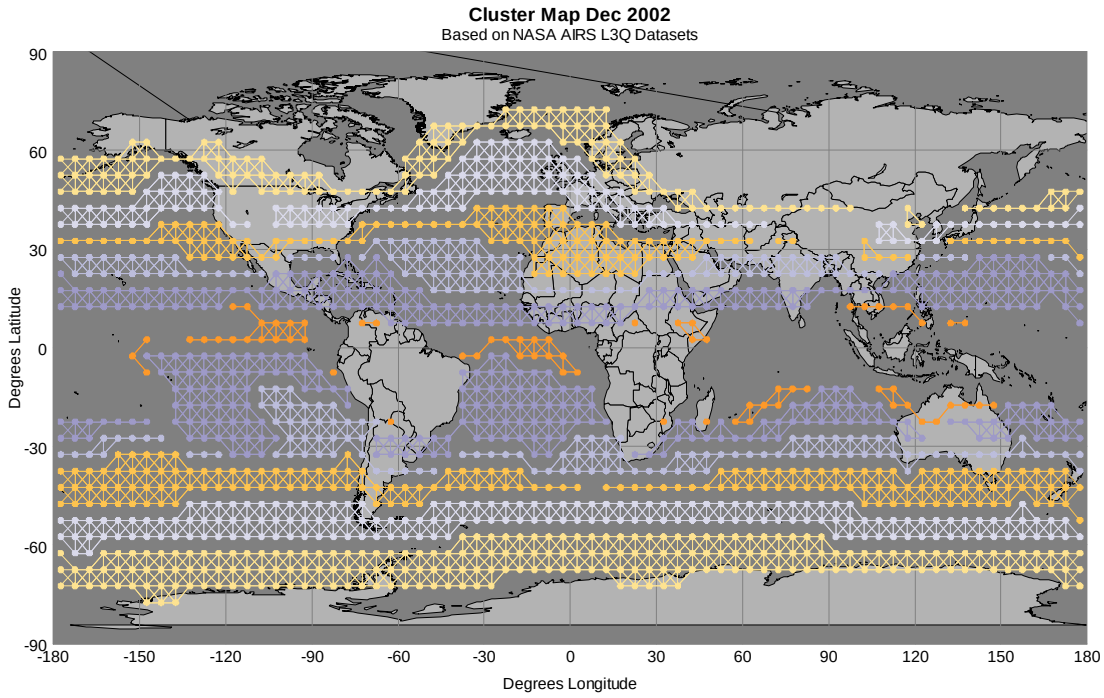


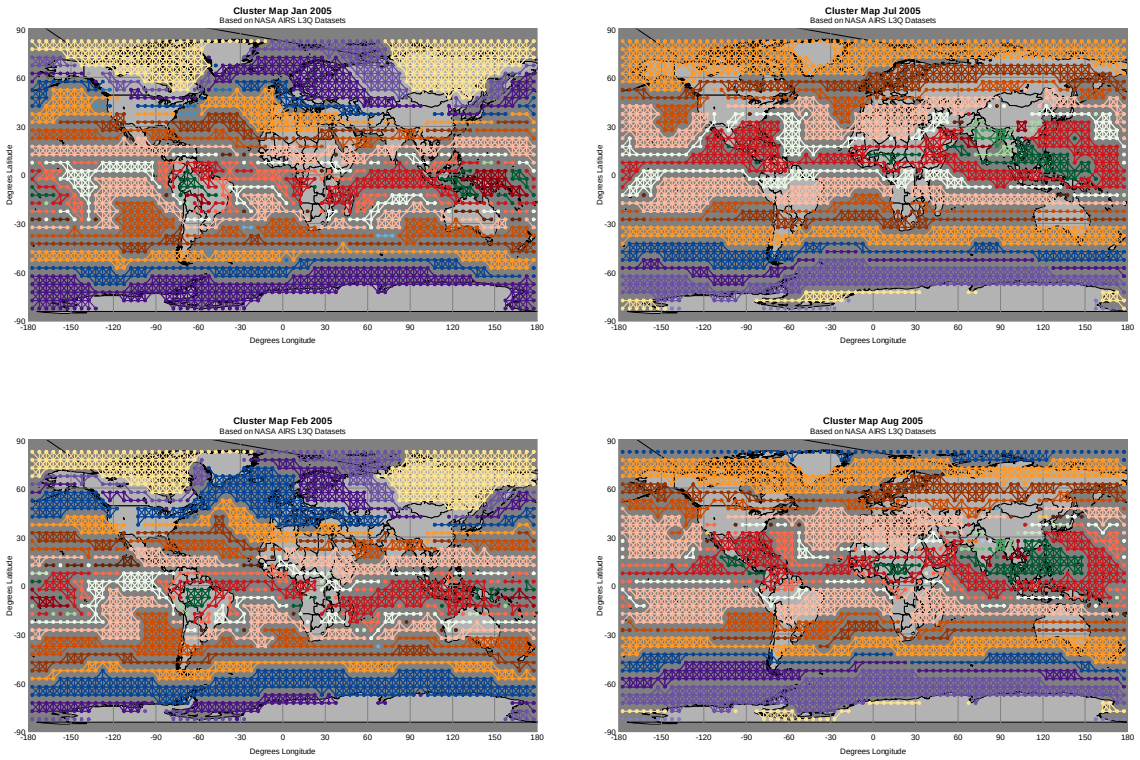

**Figure 40:Winter 2012, North America Study Area, 6 Colors**

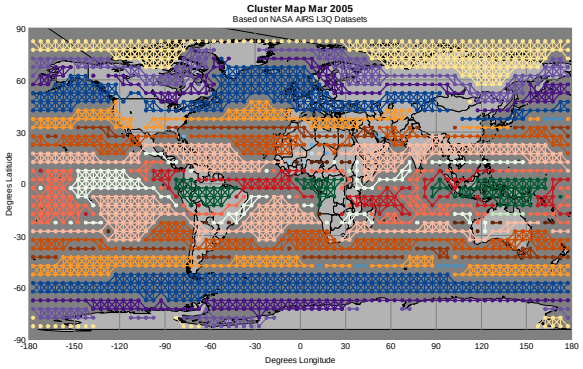
Figure 40, above, was generated using a study area covering North America from southern Canada to the Caribbean from Dec 2002-Feb 2002. The hierarchical cluster tree was cut in such a way that over the three months in the study, six clusters covered the study area. Every cell in any of those clusters, including those cells outside the area of study, were colored. With so few clusters, strong latitudinal banding is evident. We can

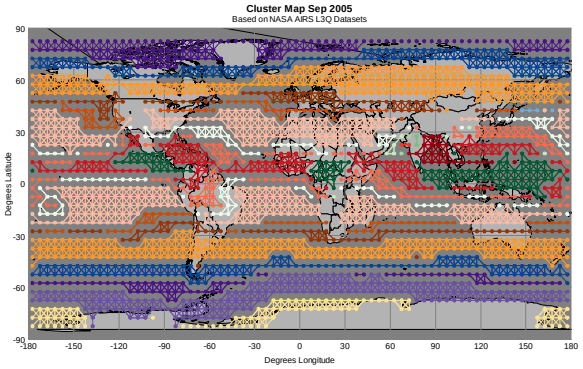
see that rainforested and monsoon areas in South America, sub-Saharan Africa, and the islands of Southeast Asia have no analogues in the North America study region. Likewise, large areas of the Arctic and the steppes of Asia have no high level analogues.

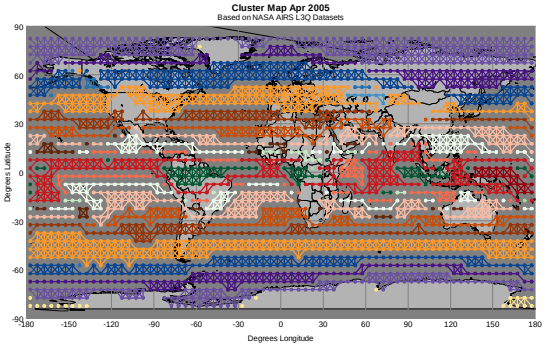
## 5.6 Other Datasets

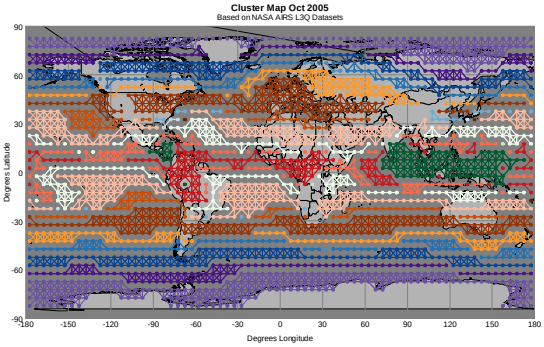
The methods outlined in this research are not limited to a single season. An analysis performed on the entire calendar year 2010, for instance, using a cutoff threshold of 3, generated 630 cluster representatives. The elapsed time to calculate the distance matrix was 1320.88 seconds (just over 20 minutes). The PACA was used to generate a 32 color map over the entire 12 months.

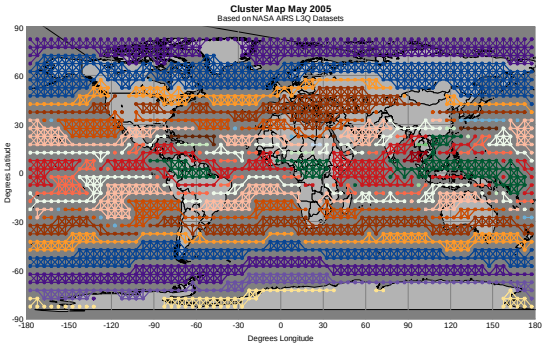


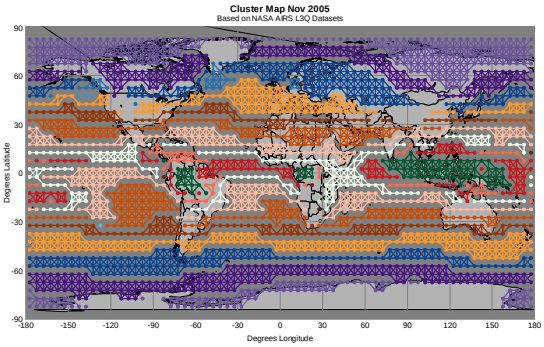

Cluster Map Mar 2005
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude


Cluster Map Sep 2005
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude


Cluster Map Apr 2005
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude


Cluster Map Oct 2005
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude


Cluster Map May 2005
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude


Cluster Map Nov 2005
Based on NASA AIRS L3Q Datasets
Degrees Latitude
Degrees Longitude

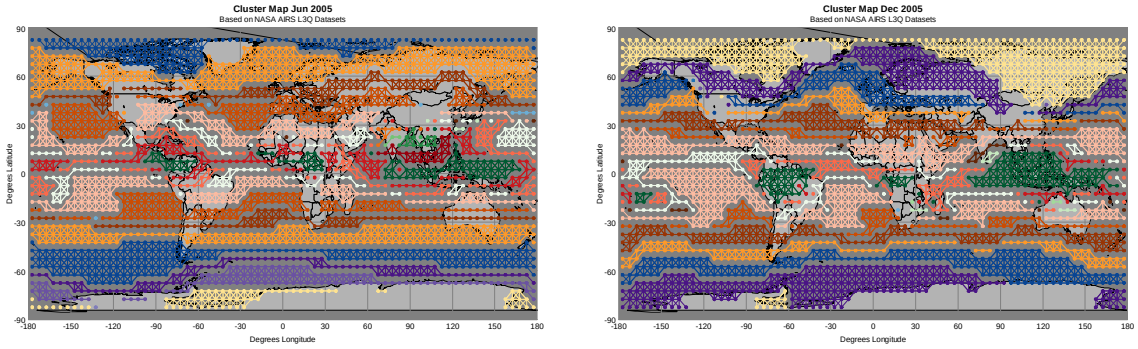



**Figure 41: PACA Coloring of Jan-Dec 2005**

The first column in the graphic above is January-June; the second column is July-December. Even with only 32 colors, interesting structure is revealed in the tropics, and various seasonal shifts are apparent. The method can be extended to longer and longer time horizons although obviously the ability to pick out detail is limited by the number of distinct colors available in the palette at this level of visualization. Using the multi-cluster and single set over time charts, the only practical limit on the number of clusters that can be displayed is the user's patience with the compute and with paging through multiple screens.

## 5.7 Computational Complexity of the Method

In general, the algorithms used in this research are reasonably performant at this time. Some straightforward tuning in most sections eliminated most bottlenecks. With that said, there remain three items outside the actual production of visualizations (dependent on the graphics processing power of the machine) that can be long enough to generate a need for a coffee break.

The first is, of course, the pre-process step of preparing a dataset for use with the tools I developed. The conversion from HDF4 to R and C friendly binary formats takes around a second; neglible for a one time cost. The calculation of the complete intra-month distance function, however, at roughly 7-8 core-hours on a CPU is a one time cost that could bear with some performance improvements. Future research in this arena could be fruitful, especially if a parallel/vector processor friendly solver for the LP Transport problem (i.e. an engine to compute Wassertein distances) can be developed. This is left as a future research opportunity.

The second is closely related to the first: the calculation of the Wasserstein distances between the cluster representatives. Certainly, a cache or compute strategy could be adopted with an ever-growing database of distance pairs. This is left as a future research opportunity as current analysis times are significantly reduced for a large set of cases.

The third is seen in Figure 43, in the step ColorMaskPre. The steps are explained in Table 2: Algorithm Analytic Phases. This step uses the study mask to define the cluster representative coverage of the mask. It then iteratively cuts the cluster hierarchy until it finds the cutset with all the "in mask" representatives that covers the specifically requested number of clusters, which it then passes on for coloring.

Without the use of a mask, the following is a typical performance-by-phase timing chart of elapsed time for an analysis. R also reports user and system times, which

are typically much less than elapsed time. In research aimed at cluster computation with a user time chargeback system that would be the appropriate measure. In this case, where the focus is more on data exploration by a user, elapsed time as perceived by a researcher seemed the more appropriate metric. The pattern below is largely preserved in the other time measures, but the magnitude of the values can be as little as $1/10^{th}$ of the elapsed time metric.

**Table 2: Algorithm Analytic Phases**

| **Phase** | **Description** |
|---|---|
| Get Representatives | Using the study hierarchical distance threshold, cut the trees of the study datasets and find the cluster representatives for each of those clusters. |
| Distance Calc | Populate the Wasserstein distance matrix for the cluster representatives. |
| Clustering | Cluster the representatives using the distance matrix, cutting the tree to prepare for the coloring of the requested number of clusters. |
| Color Mask Pre | Adjust the cutset of the tree to a study mask if required. |
| Color Pre | Mark/prepare the clusters for coloring |
| Cell Pref | Aggregate cell level palette and palette position preferences |
| Cluster Projection | Push cluster indexes down into the map layers. |
| Palettes | Assign palette preferences and palette colors to the clusters |
| Draw Map | Draw the top level maps for each study dataset |

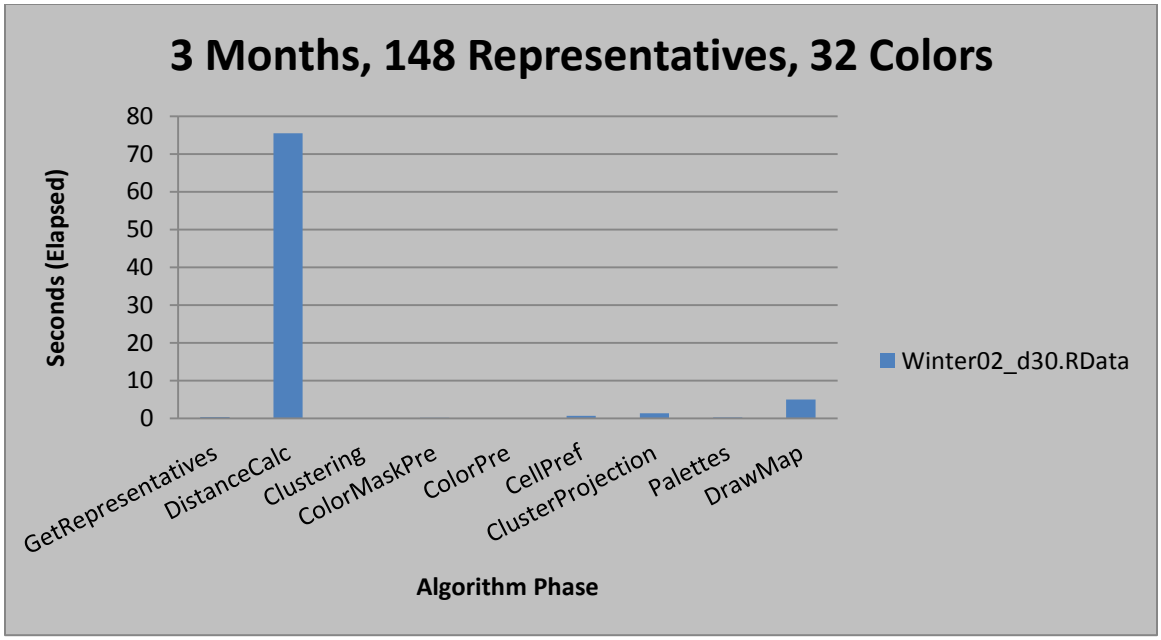



**Figure 42: Runtime by Phase Winter 2002, 32 Colors, Full Dataset**

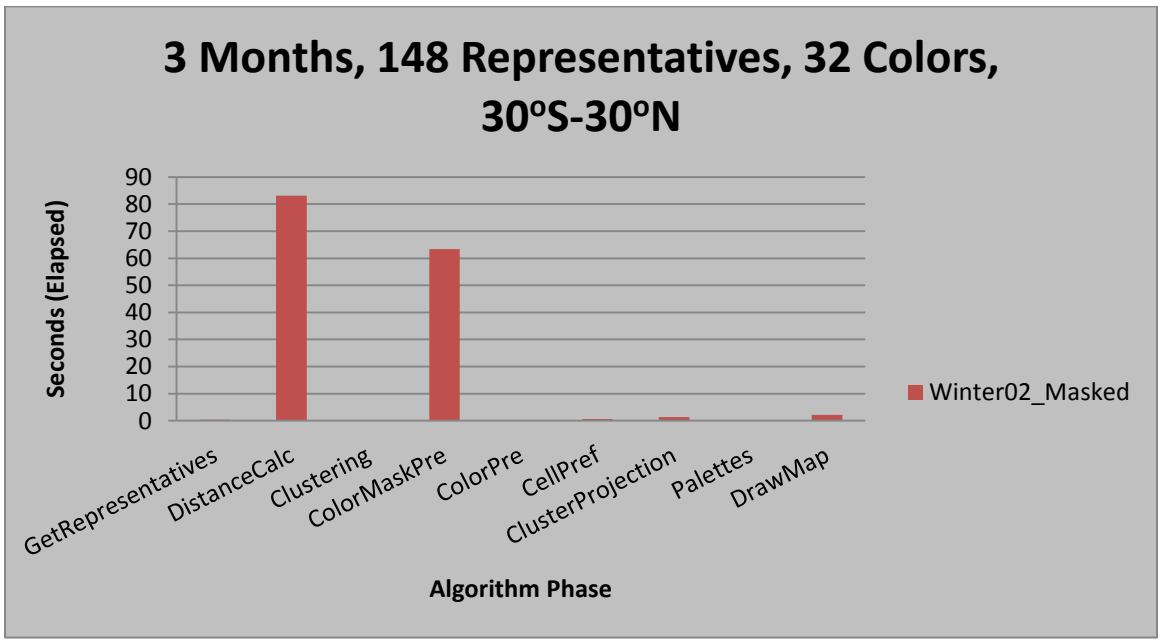



**Figure 43: Winter 02, 30°S to 30°N Timings**

From these two very typical performance charts, there is an obvious performance penalty associated with using the study masks, as the algorithm iteratively descends the

tree in the ColorMaskPre step, which ends up involving significant file IO. The distance matrix calculation is the same as the above Figure 42; the difference in elapsed time is 7.66 seconds, or less than 10% variance.

Earlier versions of the code had an additional performance bottleneck in the ClusterProjection step. Simple profile analysis with the charts above was enough to identify the area of concern; the code was slightly re-ordered to open files less often and the performance increased dramatically.

# 6 Visualization of Clustered Data in Physical Units

At this point, I have demonstrated production of global hierarchical sets of earth grid cell compressed data distributions. I have also demonstrated that these global hierarchical sets can be distinguished from each other and looked at over time using a variety of graphical and coloring techniques. What I have yet to show is a reasonable visualization of the underlying compressed data distributions in physically meaningful units. In Section 3 Related Work, we saw a number of visualizations that showed data for single cells, or a small subset of cells; or used multi-dimensional scatterplots and brushing to visualize points in the dataspace. None of these techniques, however, meets all of the following criteria that I wanted to address:

1. Be able to display all the physical data and key metadata simultaneously.
2. Preserve the connections inherent in the data.
3. Leverage the altitudinal structure of the data in the display.
4. Present the information in a geographic context.
5. Present information on a cluster wide basis – not be restricted to a single earth grid cell of data.

In the first sections below I quickly move through some "off the shelf" visualizations; I then develop, in 6.2 Customized Charting, a "pitchfork" chart which I contend meets all of the goals above and is conceptually extensible to other similar datasets.

## 6.1 Conditioned Scatterplot matrices

Scatterplot matrices are a common way to look at two or more variables, pairwise. Conditioned scatterplot matrices allow further investigation of structure within a pair of variables – in the example below, Altitude and Latitude condition scatterplots of Temperature and Cloud Cover using standard "R" conditioned scatterplot matrices.

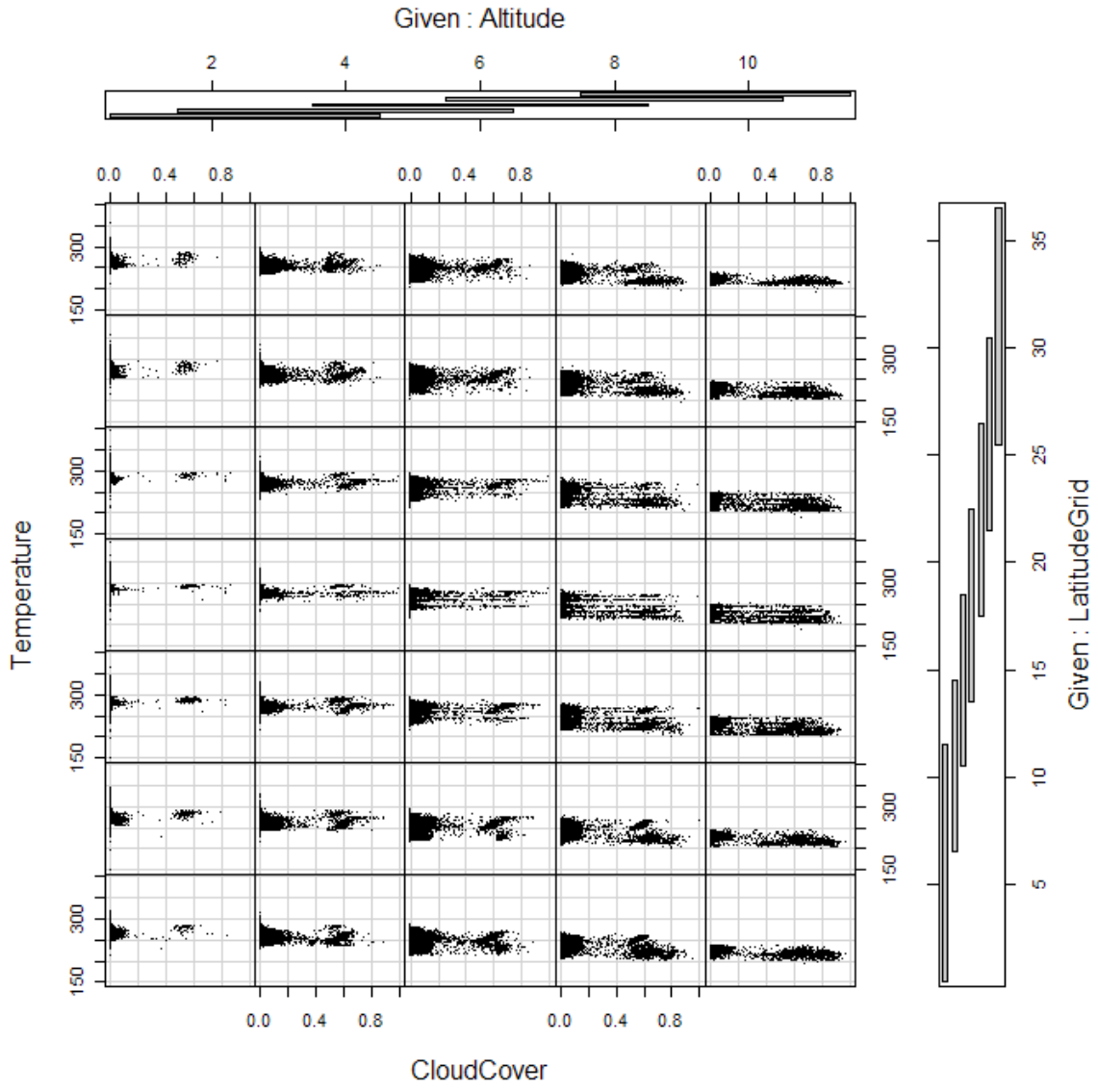



**Figure 44: Conditioned Scatterplot Matrix, Nov 2002**

In this case, Latitude is expressed in $5^{o}$ block indexes, temperature in Kelvin. We see that typical temperatures decrease with altitude and increase slightly at middle latitudes, and that cloud cover seems to have very little dependence on temperature.

The bands of the conditioning variables, Latitude and Altitude, are allowed to overlap. This plot has some temperature outliers (temperatures at low and medium altitude of under -100C and over 100C) that compress the data into a narrow central band.

This technique lets us look at the whole dataset, but doesn't capture specifics of what is happening in a grid cell or how values at one altitude depend on those at another altitude.

Figure 8 shows an alternative conditioned scatterplot style view of the data. While this is useful in its own right, it again does not preserve the relationship between different altitude levels of a given set of variables, although it does preserve the relationships intra-attitudinally.

## 6.2 Customized Charting

Figure 7 and Figure 10 are both examples of customized visualizations. In Figure 6, we see color and conditioning via small multiples being used to examine all three of the L3Q physical variables categorically by altitude using the Conditioned Choropleth chart. In Figure 10, all variables have been Z-scaled and color and area are being used to display one cell of data at a time.

The visualization developed, while obviously specific to the data under study, could easily be extended to other geospatial and altitudinal distributed data.

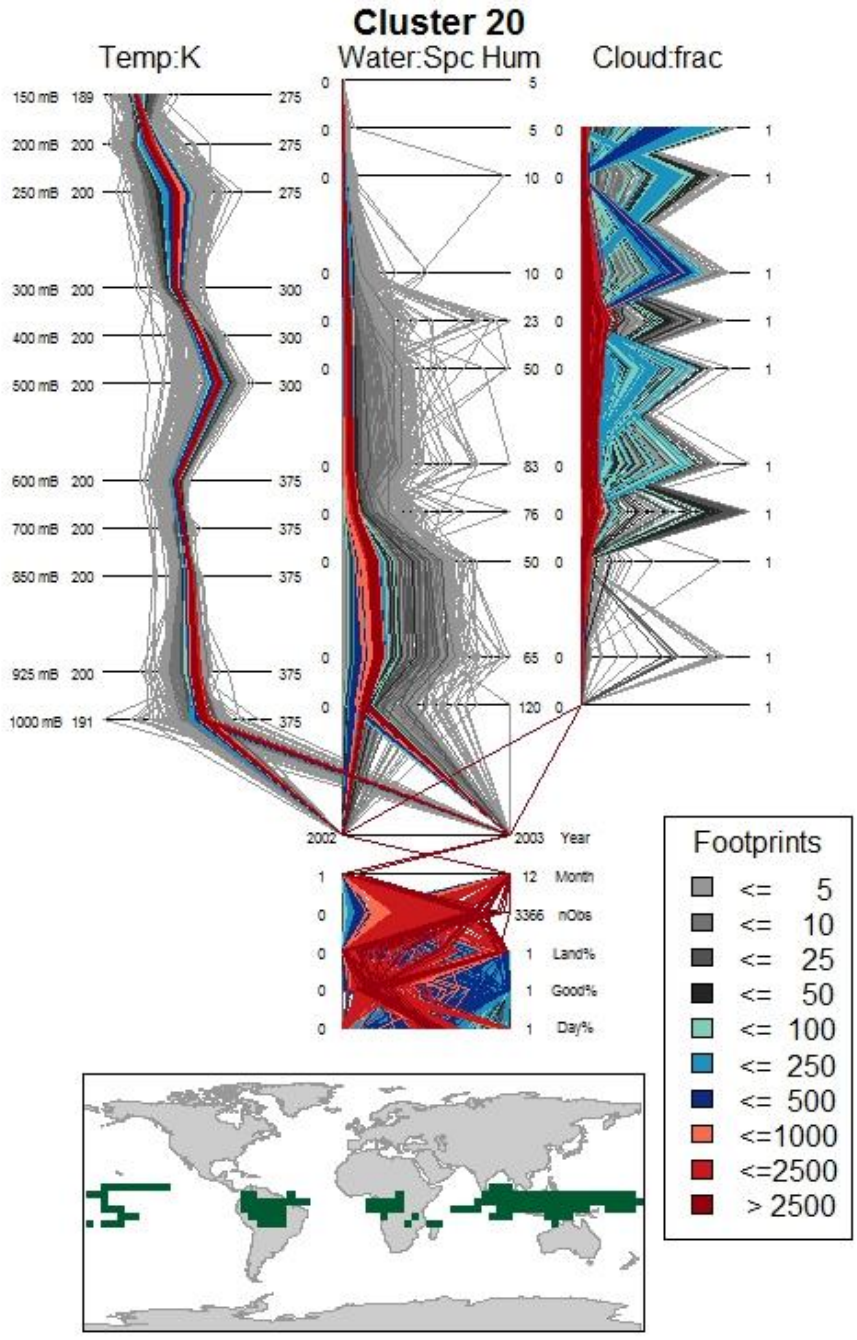



**Figure 45: L3Q Plot**

Horizontally, working up from the bottom of the page, there are three zones. The bottom is a cluster geographic footprint. This shows, over all the time in the study, the earth grid cells that are members of this cluster in at least one dataset. Land masses are displayed in grey to provide more geographic context. The cluster members are displayed in the same color they are depicted in on the overall map and the cluster summaries.

The next zone up is a parallel coordinates chart displaying the various metadata elements associated with the data vectors. From the bottom, Day, Land, and Good fractions per vector are displayed. These values range from 0 to 1. Good% is the fraction of the data observations in the original L2 dataset that were of quality Good or Best. Nobs is the number of observations in the compressed vector, and currently forms the basis for the coloring of the data lines and also for the order of plotting, which will be discussed separately below. Month and Year highlight the temporal aspects of the data.

The third zone is the physical data by altitude zone, which is further subdivided into three columns. In keeping with the data, altitude levels are indicated by millibars on the far right of the third zone.

The leftmost column is a parallel coordinates plot of temperature by altitude. Bars are labeled in Kelvins, with a nominal range provided that will be adjusted for data that might lie outside the predefined range. If the ranges were totally dynamic they would be very hard to compare. If they were totally fixed the ranges would be so large to accommodate outliers that most of the variation in the parallel coordinates lines would be hidden.

The center column is a parallel coordinates plot for water vapor, reported in specific humidity – water vapor content on a mass basis per unit mass of air. The bars are offset slightly relative to the temperature bars as this data represents the integration of a quantity over a range between the two pressure levels whereas the temperature is intended to be the value at a pressure level. Like the temperature bars, the ends are given a preset range that the data can override.

The rightmost column is the cloud fraction. Due to the nature of the data processing that assigns these values, the majority of values for any vector will be zero. The data ranges from 0 to 1, and there is no data calculated for the lowest atmospheric band.

In order to maintain the connection between the metadata and the physical observations, the parallel coordinates plots are connected from the top of the metadata section (Year) to the bottom of the three physical data values (Temperature, Water Vapor, Cloud Fraction).

Colors are assigned to vectors based on the number of observations in the data cluster. Clusters with smaller counts get increasingly dark shades of grey, those with intermediate counts get increasingly dark shades of blue, and the largest clusters get darker and darker shades of red. The order of display of the vectors is also driven by the number of observations – the smallest clusters are plotted first. So, larger clusters will overplot smaller ones, causing more intense colors to overplot less intense ones. The eye is thus drawn easily to the "core" of the distributions, but smaller clusters and outliers are still visible.

The charts can also have filters applied, and can be examined, filtered, side by side, for instance as in Figure 46 , which is the same cluster mapped in Figure 39.

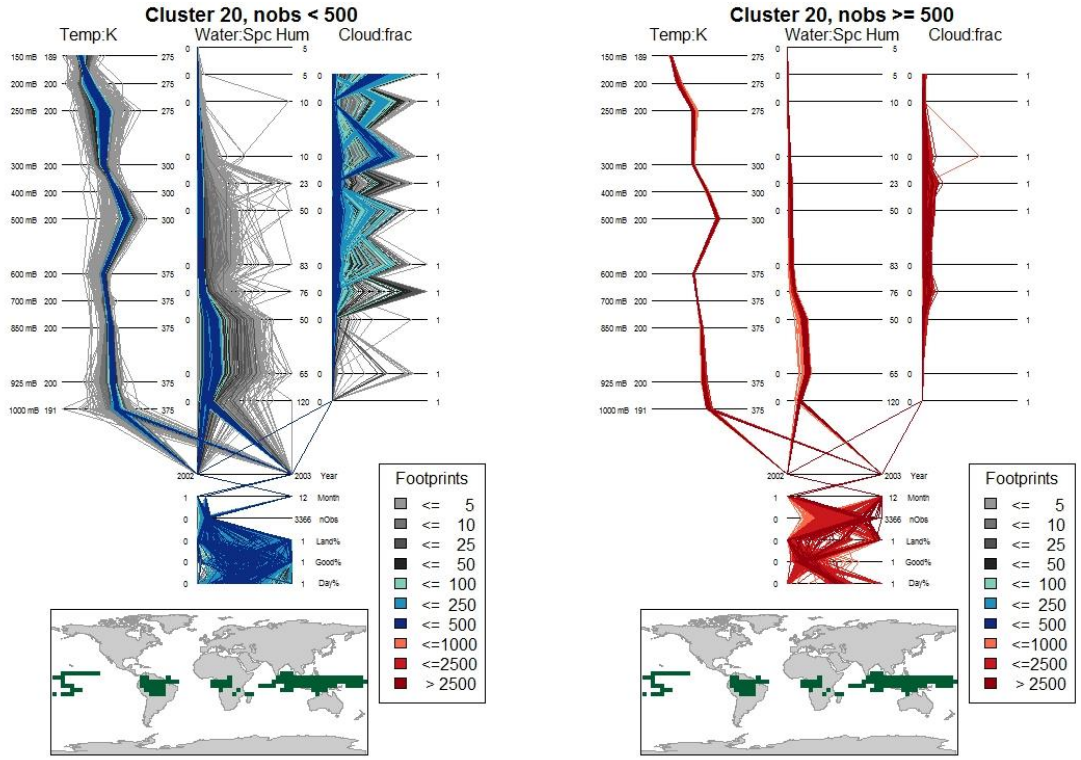



**Figure 46: Filtered Charts**

Alternative arrangements that have the same properties could be envisioned, of course. For example, the metadata could be split into two sections, and one placed over and one under the physical data. The meta-data bars could be widened to cover some

larger portion of the width of the physical data. The footprints map could be placed in an additional column. The whole layout could have been turned on its side, which would be perhaps more optimal for display on a computer screen. I chose to keep altitude vertical because that is a more natural and intuitive mapping.

More creative approaches might attempt to scale the width of the sections by some relative information theoretical weight, or even to scale the width of the bars by the relative contribution of each dimension to the various PCA clustering dimensional vectors, producing a possibility of a tour of the physical data with different "informational scalings" on display. All of these options would make interesting future work.

# 7 Putting it all together – Science

In this section, I will start by looking at the data analysis in the related work and showing that the techniques I have developed can provide similar insights into the data. I will also show where this research allows me to extend their analysis to ask additional questions of the data.

## 7.1 Cold Oceanic Upwelling in Winter 2002

In [Carr, Braverman 2007] they hypothesize that the southern hemisphere oceanic clusters off the west coast of South America and Africa (seen in Figure 5) are caused by atmospheric cooling over upwellings of cold Antarctic water. Can this research find similar clusters and demonstrate the atmospheric cooling they hypothesize?

My attempt to show this begins with clustering the Winter 2002 dataset used in [Carr, Braverman 2007] which includes December 2002, January 2003, and February 2003. I select a threshold that produces 143 cluster representatives across the three months and color with 32 colors to produce the maps of Figure 33.

In the figure below, I have extracted, for each month, the relevant section of the three maps in those figures.

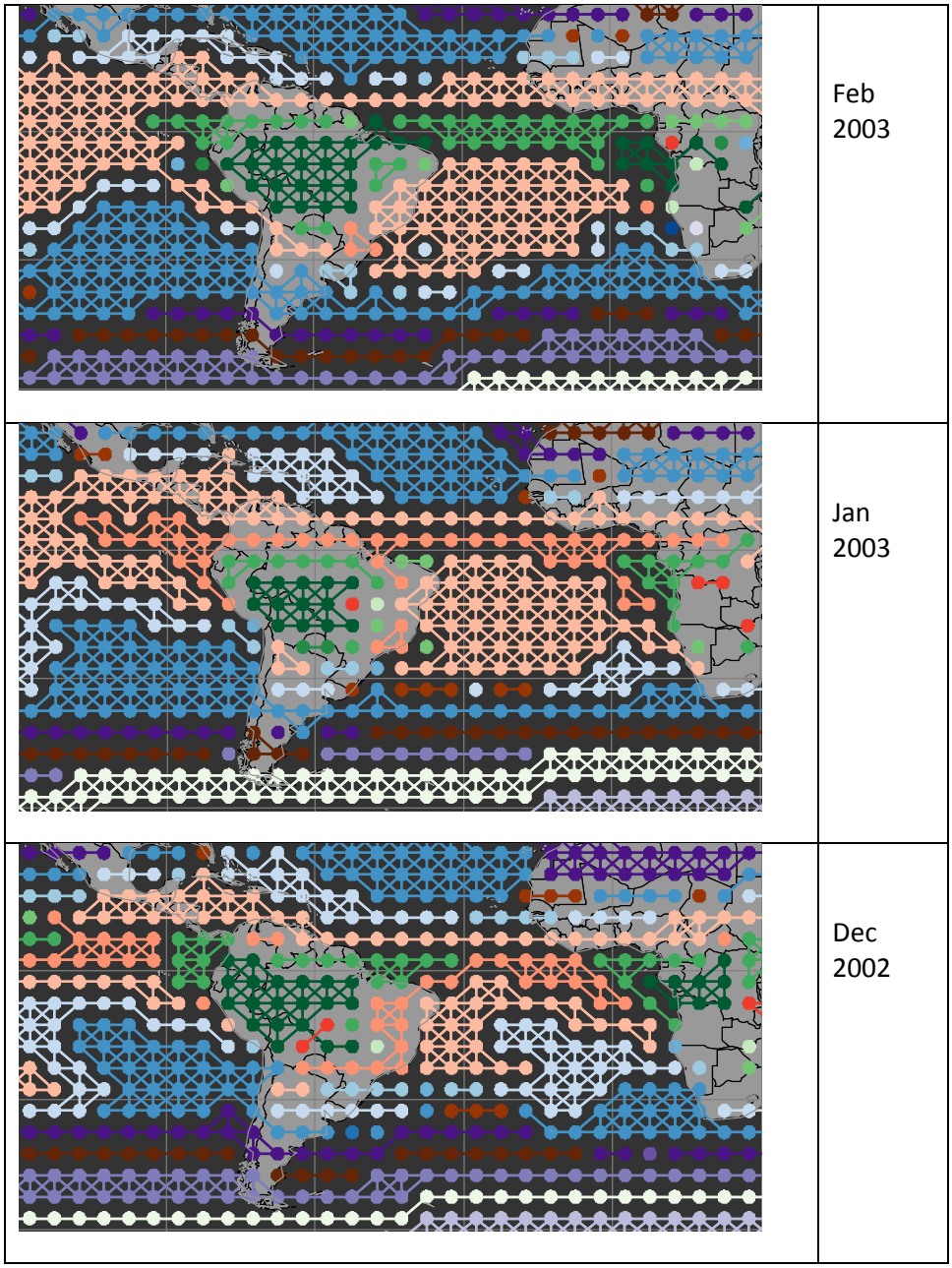
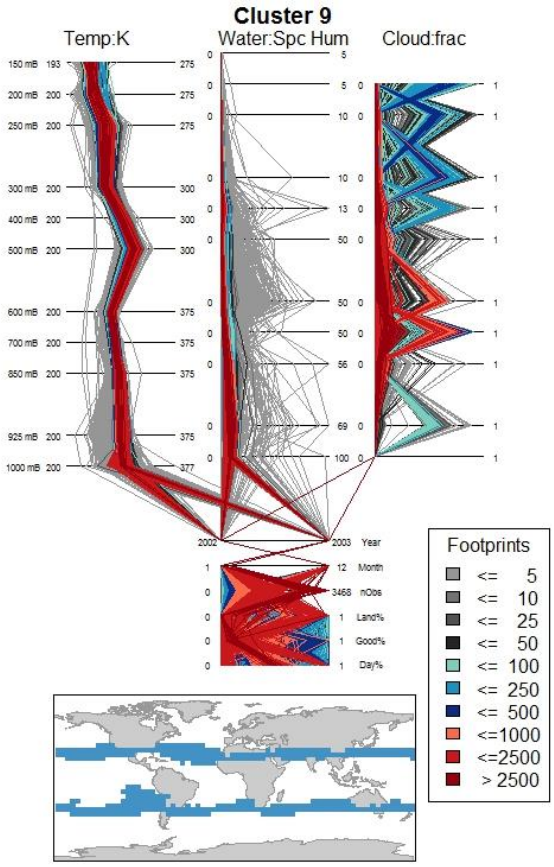



**Figure 47: Winter 2002 Antarctic Upwelling Study**

These clusters across the three months clear show some similarity to the clusters identified in the aggregated seasonal data in [Carr, Braverman 2007], Figure 5. Likely candidates include the medium blue off the coast of South America in all three months, the light blue area that edges the medium blue off South America and that appears of the African coast in December and February, and the pale red that dominates the South Atlantic in January.

Using an interactive function to get the cluster numbers, these are respectively found to be clusters 9, 12, and 15. What do these clusters look like, physically?


Cluster 9
Temp:K
Water:Spc Hum
Cloud:frac
150 mB
200 mB
250 mB
300 mB
400 mB
500 mB
600 mB
700 mB
850 mB
925 mB
1000 mB
2002
2003 Year
12 Month
3468 nObs
Land%
Good%
Day%
Footprints
<= 5
<= 10
<= 25
<= 50
<= 100
<= 250
<= 500
<=1000
<=2500
> 2500

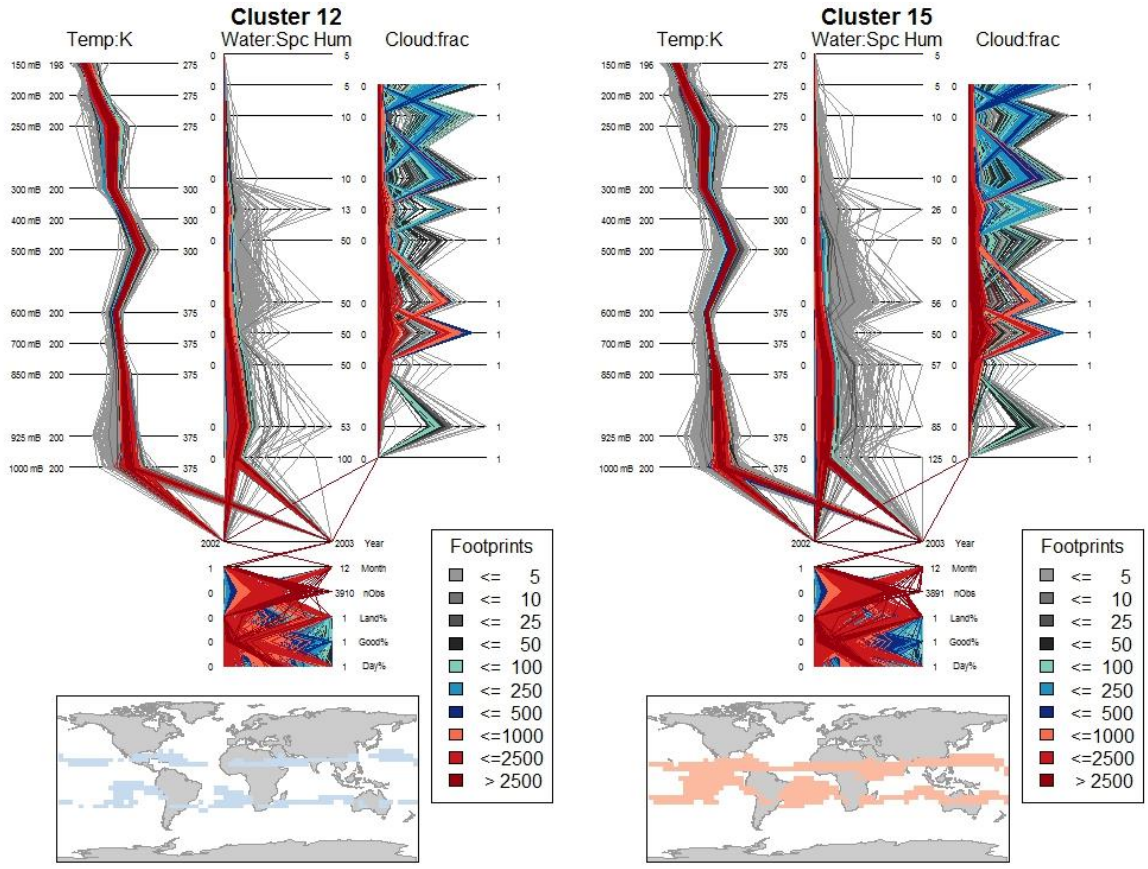



**Figure 48: Physical Cluster Upwelling Candidates**

Examining the footprints, first, we see that cluster 12 is perhaps the best fit, but all of these are broadly in line with the larger cluster identified in Figure 5. If the process backing this cluster includes upwelling of cold water, we would expect cold and, broadly, dry air. The temperature profiles of all three clusters are broadly similar, but cluster 12 seems to be slightly more tightly defined.

The water vapor scales for cluster 9 and 15 are significantly wider than those of cluster 12.

**Table 3: Comparison of Specific Humidities**

| Maximum Specific Humidity at Low Altitude | | | |
|---|---|---|---|
| Altitude | Cluster 9 | Cluster 12 | Cluster 15 |
| 850mb | 58.4 | 50.0 | 56.9 |
| 925mb | 68.9 | 53.1 | 85.4 |
| 1000mb | 100.0 | 100.0 | 125.1 |

With cluster 12 also appearing more tightly defined for the bulk of the largest clusters, it would seem that cluster 12 is dryer air than clusters 9 and 15.

Cloud cover is broadly similar, although perhaps cluster 12 has less overall low level cloud than clusters 9 and 15, even though the distribution of cloud cover at higher altitudes looks similar. It seems that the Winter cluster identified in [Carr, Braverman 2007] probably represents a general temperature pattern similar to the pattern in three of the clusters I have identified with a cluster of colder, drier air as a significant component. In this case, being able to drill into the physical data suggests that there is indeed a pattern of temperature and specific humidity in the data that is consistent with a cool upwelling off the coast of South America. Further advances in the exploratory power of the charts that could be explored in the future (outside of R) would be brushing and profile filtering.

## 7.2 North American Clusters in Winter 2002

In this case, comparing with [Zhou, Shi 2011], there is a limited study area, a single month, and a small set of clusters to compare our results to. There is a visualization of both the geographical clustering and also of some of the individual sets of data vectors. The question, then, is can we cluster such an area, using a small set of clusters, and visualize them. As a reminder of this target, please refer to Figure 9 and Figure 10.

I have already shown I can generate a map with restricted sets of colors, and also that is derived only from cells that cover a given area, in Figure 40. I have chosen to include mapping of members of the cluster from outside the study area to provide additional context.

Examining the roughly analogous cells from [Zhou, Shi 2011], I get Figure 49.

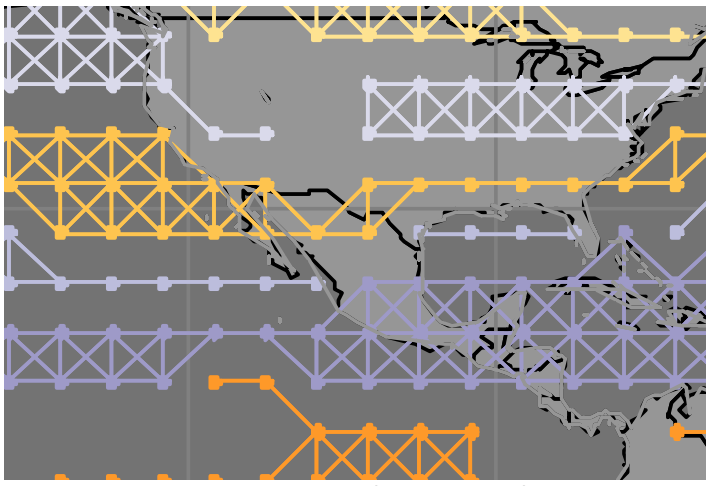


**Figure 49: Dec 2002, North AmerFocus Clusters**

Following [Zhou, Shi 2011], I select three cells in a vertical strip off the coast of Baha California, from two different clusters. Note that in my clustering, the middle cell clusters with the northern rather than the southern cluster.

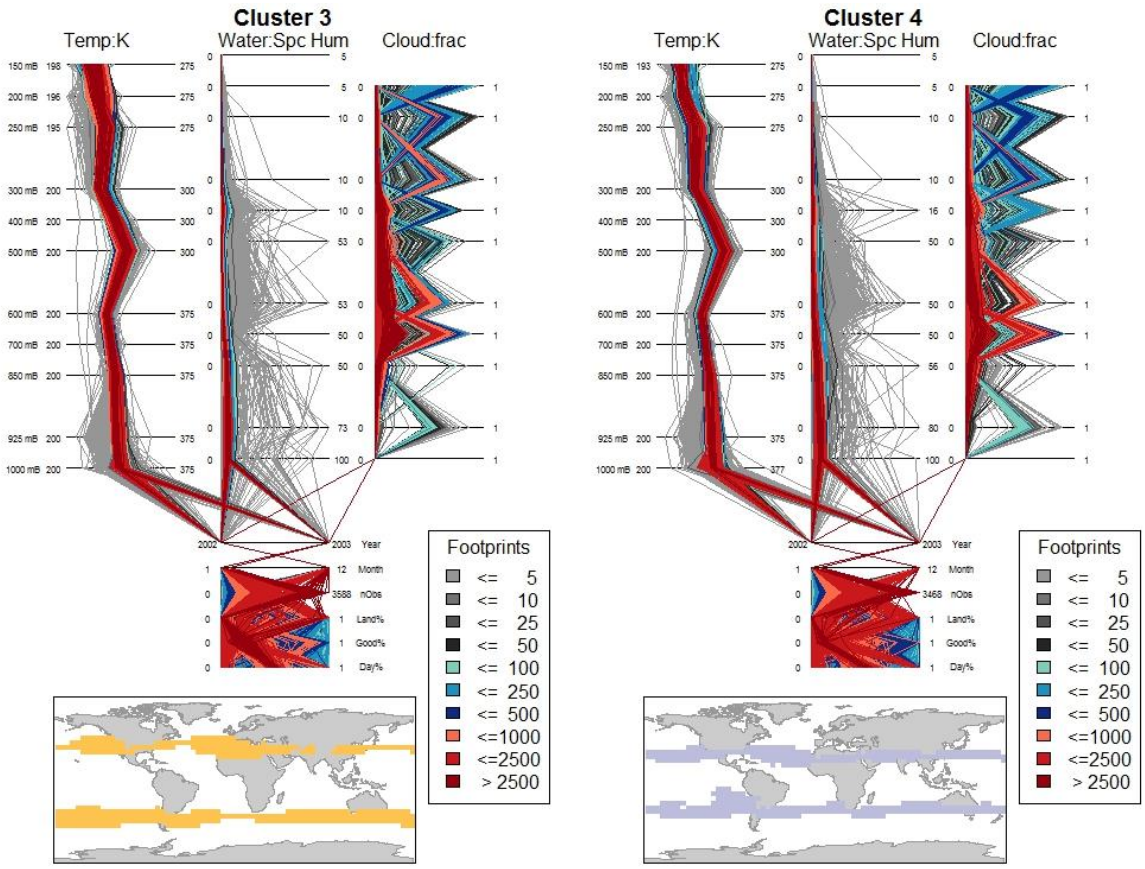



**Figure 50 Clusters off the coast of North America**

Does the view above support give the same types of insight as Figure 10? From Figure 50, it appears that Cluster 3 has: a tighter spread of slightly cooler temperatures except in the middle atmosphere where Cluster 4 is more tightly distributed; is generally drier air, with more cloud in the range from 800-300mb.

In the diagram in [Zhou, Shi 2011], we would see the neighboring cells from the equivalent of cluster 3 (the northern cluster, A1) show slightly lower temperatures compared to the southern cluster (4 / A3). The air is slightly drier to the north, and there is a significant difference in cloudiness at altitudes above 800mb.

Certainly, it would be possible to adapt my plot to show a single cluster or even two clusters, overlaid, using two different palettes. This would more closely duplicate the actual content of the visualization from [Zhou, Shi 2011] but even looking at the cluster level we can see the differences – and this plot is not limited to single cells. Nevertheless, this drilled down comparison would make interesting future work.

## 7.3 Vertical Velocity from MDS Scaling for Winter 2002

In [Braverman et al, 2012b], high and low values of the second dimension of 6D multidimensional scaling are shown to correspond roughly with an atmospheric measurement known as vertical velocity. This value is not directly derivable from the values in the L3Q data. Values of vertical velocity that correspond to strong upward movements of air tend to be associated with moist rising air, which cools rapidly, creates high-topped clouds,  and releases significant precipitation. In Figure 13, we see negative

values west of Australia and high positive values over Indonesia. Examining the data from **Error! Reference source not found.**, we can select clusters that correspond to the low and high value cases – clusters 9 and 20 in my analysis. We see evidence for higher vertical velocity in Cluster 20 in the pair of plots below.

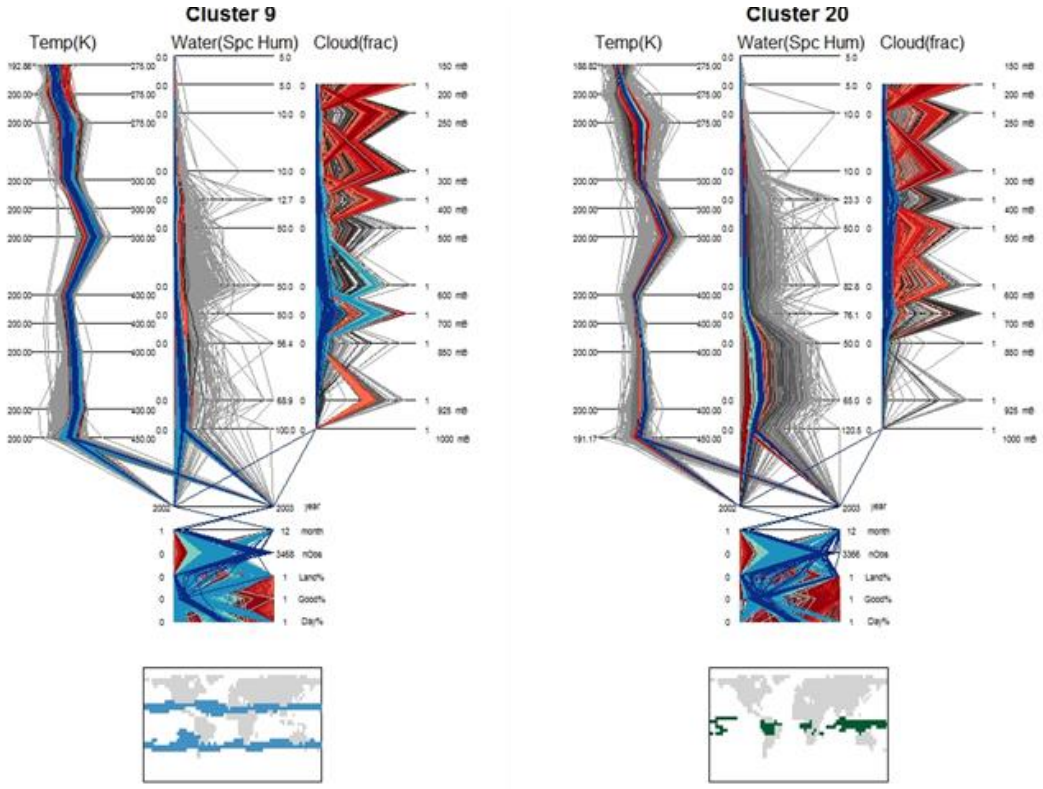



**Figure 51: Evidence supporting vertical velocity**

Specifically, it is apparent that there is significantly higher specific humidity below the 700mb level in Cluster 20 than in Cluster 9. Cluster 9 has the heaviest

concentration of vectors with cloud cover maximums between 700 and 600mb. Cluster 20, in contrast, has peak weighted cloud cover between 400 and 300mb.

While this research does not produce a number that is directly correlated with vertical velocity, nor would it easily identify areas of high or low vertical velocity, it is apparent that it can show the physical signature of high and low values of vertical velocity in the data.

# 8 Conclusions

The explicitly stated goals of this research were threefold:

- to develop a method to analyze multiple AIRS L3Q datasets by producing hierarchical clusters of the data across time in reasonable time frames on minimal hardware,
- to develop a technique to automatically produce reasonable initial visualizations of this clustered data,
- and to develop a visualization that shows all the physical data values for a cluster.

In Section 3 Related Work, we see in [Braverman et al, 2012b] a technique for combining multiple AIRS L3Q datasets using repeated application of the ECVQ data compression technique across time. Analysis of this data via multi-dimensional scaling produces individual values per grid cell as is shown also in [Zhou, Shi 2011]. Finally, in both [Carr, Braverman 2007] and [Zhou, Shi 2011] we see various visualizations of the data, either on a single cell basis or on a set of points basis.

In Section 4 Creating Globally Hierarchically Clustered Sets, I demonstrate a method of selecting representatives from multiple datasets and clustering the representatives to produce multi-dataset clusters at dataset granularity. After the once-per-dataset preprocessing, arbitrary combinations of datasets can be combined in reasonable (minutes to an hour) timeframes depending on the number of cluster representatives used.

In Section

5. Displaying Multi-Dataset Clusters, a variety of visualizations are demonstrated for the clustered data outputs. This builds on the work in [Ashley, Carr 2011] and adds the ability to highlight focus clusters and study areas with variable size palettes for the initial coloring.

In Section

6 Visualization of Clustered Data in Physical Units, a novel graphic based on multiple connected parallel coordinate plots and a map is developed. This graphic leverages the structure of the physical values to help drive the layout, which should help with developing intuitions about the data.

In Section 7 Putting it all together – Science, I show that the tools developed here can easily duplicate most of the analysis from the related work and can indeed extend on that work in many cases.

While the research has shown that it is possible to meet the threefold goals it set out to achieve, it has also uncovered a variety of future research opportunities for the techniques.

- A study of cluster stability and the impact of using different representatives from the same datasets for overall analysis.
- A study of possible GPU/vector friendly parallel solvers for the Wasserstein distance.
- A study of alternative color palettes.

- A study of application of the Palettized Autoomated Coloring Algorithm to other datasets.
- A study of alternative layouts, component sizes, and colors of the physical data plots.
- A study allowing more interactive exploration – filtering and brushing – for the physical data plots.
- A study of differential statistical analysis of multiple clusters.
- A study of using K-means or ECVQ to produce cluster summaries for visualization clarity.
- A study of how to include additional information in the physical plots from other datasets – this would potentially greatly enhance the range of science that could be done.
- A study of GPU enablement of the ECVQ compression process to allow on-the-fly compression and analysis of datasets and portions of datasets beyond AIRS L3Q.

This research, then, has been successful in what it set out to do; but it has also opened many doors along the way, which will likely reward further investment of time and effort going forward.

## Appendix 1 – Key Source Code

| | |
|---|---|
| FILE: | Hdf_convert.m |
| Purpose: | Convert HDF4 data files into multiple R compatible binary data files |

```
hdffiles = dir('../AIR3XQPM/*.hdf')
numfiles = length(hdffiles)
AIRSyears = zeros(numfiles,1);
AIRSmonths = zeros(numfiles,1);

for k=1:numfiles
    %NumClusters =
hdfread('C:\Users\jashley\Documents\AIR3XQPM\AIRS.2002.09.01.L3.R
etQuant030.v5.0.14.0.G07284021707.hdf', '/L3Quant/Data
Fields/NumClusters', 'Index', {[1  1],[1  1],[36  72]});
    [htok,ttok] = strtok(hdffiles(k).name,'.');
    [fyear,ttok] = strtok(ttok,'.');
    [fmonth,ttok] = strtok(ttok,'.');
    AIRSyears(k) = str2num(fyear);
    AIRSmonths(k)= str2num(fmonth);
        LatCenter   =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/LatCenter', 'Index', {[1  1],[1
1],[36  72]});
        LonCenter =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/LonCenter', 'Index', {[1  1],[1
1],[36  72]});
        NumClusters =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/NumClusters', 'Index', {[1  1],[1
1],[36  72]});
        NumObsInCluster =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/NumObsInCluster', 'Index', {[1  1
1],[1  1  1],[100   36   72]});
        ClusterMeanSquaredError =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/ClusterMeanSquaredError', 'Index',
{[1  1  1],[1  1  1],[100   36   72]});
        ClusterDistortion =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
```

```
.name), '/L3Quant/Data Fields/ClusterDistortion', 'Index', {[1  1
1],[1  1  1],[100   36   72]});
        Entropy =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/Entropy', 'Index', {[1  1  1],[1  1
1],[200   36   72]});
        GridCellMeanSquaredError =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/GridCellMeanSquaredError', 'Index',
{[1  1  1],[1  1  1],[200   36   72]});
        NormalizedValues =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/NormalizedValues', 'Index', {[1  1
1  1],[1  1  1  1],[100   18   36   72]});
        PhysicalValues =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/PhysicalValues', 'Index', {[1  1  1
1],[1  1  1  1],[100   35   36   72]});
        PentadComposition =
hdfread(strcat('C:\Users\jashley\Documents\AIR3XQPM\',hdffiles(k)
.name), '/L3Quant/Data Fields/PentadComposition', 'Index', {[1  1
1  1],[1  1  1  1],[100    6   36   72]});

        %File Names
        fnameH =
strcat('C:\Users\jashley\Documents\AIR3XQPM\',fyear,'.',fmonth,'.
')

        %36x72 -- % Column major order ie first dim changes
fastest.
        f=fopen(strcat(fnameH,'36x72.single.LatCenter'),'w+')
        fwrite(f,LatCenter,'single')
        fclose(f)

        f=fopen(strcat(fnameH,'36x72.single.LonCenter'),'w+')
        fwrite(f,LonCenter,'single')
        fclose(f)

        f=fopen(strcat(fnameH,'36x72.int16.NumClusters'),'w+')
        fwrite(f,NumClusters,'int16')
        fclose(f)

        %100x36x72
```

```
f=fopen(strcat(fnameH,'100x36x72.int16.NumObsInCluster'),'w+')
        fwrite(f,NumObsInCluster,'int16')
        fclose(f)


f=fopen(strcat(fnameH,'100x36x72.single.ClusterMeanSquaredError')
,'w+')
        fwrite(f,ClusterMeanSquaredError,'single')
        fclose(f)


f=fopen(strcat(fnameH,'100x36x72.single.ClusterDistortion'),'w+')
        fwrite(f,ClusterDistortion,'single')
        fclose(f)

        %200x36x72
        f=fopen(strcat(fnameH,'200x36x72.single.Entropy'),'w+')
        fwrite(f,Entropy,'single')
        fclose(f)


f=fopen(strcat(fnameH,'200x36x72.single.GridCellMeanSquaredError'
),'w+')
        fwrite(f,GridCellMeanSquaredError,'single')
        fclose(f)

        %100x18x36x72

f=fopen(strcat(fnameH,'100x18x36x72.single.NormalizedValues'),'w+
')
        fwrite(f,NormalizedValues,'single')
        fclose(f)

        %100x35x36x72
        size(PhysicalValues)

f=fopen(strcat(fnameH,'100x35x36x72.single.PhysicalValues'),'w+')
        fwrite(f,PhysicalValues,'single')
        fclose(f)

        %100x6x36x72
        size(PentadComposition)
```

```
f=fopen(strcat(fnameH,'100x6x36x72.int16.PentadComposition'),'w+'
)
        fwrite(f,PentadComposition,'int16')
        fclose(f)

end
```

| | |
|---|---|
| FILE: | DistEMD_XX.r |
| Purpose: | Computes the Earth Mover's Distance for a given granule |

```
#Build distance matrix for Year X Month X and Year Y Month Y

#Load library
library("lpSolve",character.only=TRUE)

#Function takes xYr,xMo,yYr,yMo and creates a file dist.xYr.xMo.yYr.yMo
DistEMD_XX <- function(xYr, xMo) {
        xFnameHead = sprintf('%i.%02i.',xYr,xMo)

        #Read XXXX.xx.36x72.int16.NumClusters
        fName = paste(".\\",xFnameHead,"36x72.int16.NumClusters",sep="")
        binfile = file(fName,'rb')
        inBin = readBin(binfile,integer(), size=2, n=(36*72))
        close(binfile)
        xNumClusters = array(data=inBin,dim=c(36,72))

        #Read XXXX.xx.100x18x36x72.single.NormalizedValues
        fName = paste(".\\",xFnameHead,"100x18x36x72.single.NormalizedValues",sep="")
    binfile = file(fName,'rb')
        inBin = readBin(binfile,numeric(), size=4, n=(100*18*36*72))
    close(binfile)
        xNormVals = array(data=inBin,dim=c(100,18,36,72))

        #Read XXXX.xx.100x36x72.int16.NumObsInCluster
        fName = paste(".\\",xFnameHead,"100x36x72.int16.NumObsInCluster",sep="")
    binfile = file(fName,'rb')
        inBin = readBin(binfile,integer(), size=2, n=(100*36*72))
    close(binfile)
        xNOBSClust = array(data=inBin,dim=c(100,36,72))

        distEMD = array(data=NA,dim=c(36,72,36,72))
        timeEMD = array(data=NA,dim=c(36,72,36,72))

        nDim = 18

        #For all cells in X
    for (xLat in 1:36) {  for (xLon in 1:72) {
                # is there a distance to compute
```

```
		if ((xNumClusters[xLat,xLon]) == 0) next

		#For all cells in Y
		for (yLat in xLat:36) { for (yLon in xLon:72) {
			# is there a distance to compute
			if ((xNumClusters[yLat,yLon]) == 0) next

			if ( xLat == yLat && xLon == yLon) next

			#Prep X
			#x,px,nx
			xNC = xNumClusters[xLat,xLon]
			xNV = matrix(data=xNormVals[1:xNC,18,xLat,xLon],xNC,nDim)
			xp = array(data=xNOBSClust[1:xNC,xLat,xLon],xNC)
			xps = sum(xp)
			xp = xp / xps

			#Prep Y
			#y,py,ny
			yNC = xNumClusters[yLat,yLon]
			yNV = matrix(data=xNormVals[1:yNC,18,yLat,yLon],yNC,nDim)
			yp = array(data=xNOBSClust[1:yNC,yLat,yLon],yNC)
			yps = sum(yp)
			yp = yp / yps

			#Prep L2
			L2= matrix(NA,xNC,yNC)
			for (i in 1:xNC) {
				for (j in 1:yNC) {
					L2[i,j] = sqrt(sum((xNV[i,]-yNV[j,])**2))
				}
			}

			#Calc Distance
			row.signs <- rep("=", xNC)
			col.signs <- rep("=", yNC)
			elpTime <- system.time(pout <-
lp.transport(L2,direction="min",row.signs,xp,col.signs,yp,integers=NULL))
			distEMD[xLat,xLon,yLat,yLon] = pout$objval
			timeEMD[xLat,xLon,yLat,yLon] = as.numeric(elpTime[2])
		}} #End Y
```

```
        }} #End X
        fOut = paste(xFnameHead,"self","36x72x36x72.double.EMD",sep="")
        writeBin(object=as.vector(distEMD,"double"),"double",con=fOut)
        fOut2 = paste(xFnameHead,"self","36x72x36x72.double.TIM",sep="")
        writeBin(object=as.vector(timeEMD,"double"),"double",con=fOut2)
        return(0)
} #end function
```

| | |
|---|---|
| FILE: | ClusterReps.r |
| Purpose: | Find the cluster representatives for a given threshold in a granule |

```
ClusterReps <- function(xYr, xMo, dThresh) {
        xFnameHead = sprintf('%i.%02i.',xYr,xMo)

        fName = paste(xFnameHead,"hclust",sep="")
        load(fName) # cellPtr, localClust, di, dj, dClust

        aTree = cutree(localClust,h=dThresh)

        maxCN = max(aTree)
        cRep = array(data=NA,dim=maxCN)
    nRep = array(data=NA,dim=maxCN)
        for (k in 1:maxCN) {
                worklist = which (aTree == k)
                worklen = length(worklist)
                nRep[k] = worklen
                if (worklen == 1) {
                        cRep[k] = worklist[1]
                } else {
                        # figure this out.
                        m = array(data=0,dim=worklen)
                        for( n in 1:worklen) {
                                dL = which(worklist < worklist[n])
                                uL = which(worklist > worklist[n])
                                m[n] = sum(dClust[worklist[n],worklist[dL]]) +
sum(dClust[worklist[uL],worklist[n]])
                        } # for n
           mm = which (m == min(m))
                        cRep[k] = worklist[mm[1]]
                }  # end if worklen
        } # for k
        jRep = array(data=NA, dim=c(maxCN,4))
        ClusterReps = cRep
        jRep[,1] = xYr
        jRep[,2] = xMo
        jRep[,3] = cRep
        jRep[,4] = nRep
        # Now read in enough data to save everything needed for the next steps
```

```
        #Read XXXX.xx.36x72.int16.NumClusters
    fName = paste(".\\",xFnameHead,"36x72.int16.NumClusters",sep="")
    binfile = file(fName,'rb')
        inBin = readBin(binfile,integer(), size=2, n=(36*72))
    close(binfile)
        xNumClusters = array(data=inBin,dim=c(36,72))

        #Read XXXX.xx.100x18x36x72.single.NormalizedValues
        fName = paste(".\\",xFnameHead,"100x18x36x72.single.NormalizedValues",sep="")
    binfile = file(fName,'rb')
        inBin = readBin(binfile,numeric(), size=4, n=(100*18*36*72))
    close(binfile)
        xNormVals = array(data=inBin,dim=c(100,18,36,72))

        #Read XXXX.xx.100x36x72.int16.NumObsInCluster
        fName = paste(".\\",xFnameHead,"100x36x72.int16.NumObsInCluster",sep="")
    binfile = file(fName,'rb')
        inBin = readBin(binfile,integer(), size=2, n=(100*36*72))
    close(binfile)
        xNOBSClust = array(data=inBin,dim=c(100,36,72))

        xPrimeNumClusters = array(data=NA, dim=maxCN )              # parallel to cellPtr
        xPrimeNormVals    = array(data=NA, dim=c(100,18,maxCN) )    # parallel to cellPtr
        xPrimeNOBSClust   = array(data=NA, dim=c(100,maxCN) )       # parallel to cellPtr

        for (i in 1:maxCN) {
                k = cRep[i]
                xx = cellPtr[k,1]
                yy = cellPtr[k,2]
                xPrimeNumClusters[i]    = xNumClusters[xx,yy]
                xPrimeNormVals[,,i]     = xNormVals[,,xx,yy]
                xPrimeNOBSClust[,i]     = xNOBSClust[,xx,yy]
        } # end for k

        #Save the representative
        fName = paste(xFnameHead,"clustrep",sep="")
        save(maxCN, jRep, aTree, cellPtr,
       xPrimeNumClusters, xPrimeNormVals, xPrimeNOBSClust, file = fName)
}
```

| | |
|---|---|
| FILE: | dCalcFuncr.r |
| Purpose: | Computes earth mover's distance between cluster representatives |

```
# needs xNumClusters, xNormVals, xNOBSClust, nDim
dCalc <- function(x,y) {
	library("lpSolve",character.only=TRUE)
	#Prep X
	#x,px,nx
	xNC = xNumClusters[x]
	xNV = matrix(data=xNormVals[1:xNC,18,x],xNC,nDim)
	xp = array(data=xNOBSClust[1:xNC,x],xNC)
	xps = sum(xp)
	xp = xp / xps

	#Prep Y
	#y,py,ny
	yNC = xNumClusters[y]
	yNV = matrix(data=xNormVals[1:yNC,18,y],yNC,nDim)
	yp = array(data=xNOBSClust[1:yNC,y],yNC)
	yps = sum(yp)
	yp = yp / yps

	#Prep L2
	L2= matrix(NA,xNC,yNC)
	for (i in 1:xNC) {
		for (j in 1:yNC) {
			L2[i,j] = sqrt(sum((xNV[i,]-yNV[j,])**2))
		}
	}
	#Calc Distance
	row.signs <- rep("=", xNC)
	col.signs <- rep("=", yNC)
	elpTime <- system.time(pout <-
lp.transport(L2,direction="min",row.signs,xp,col.signs,yp,integers=NULL))
	if (xSource[x] == xSource[y] && dThresh > pout$objval) {
		return(dThresh)
	} else {
		return(pout$objval)
	}
}
```

| | |
|---|---|
| FILE: | ClusterRepsParallel.r |
| Purpose: | Builds the cluster representatives list in parallel |

```
#Designed to split the total work over N instances of R
clusterRepsParallel <- function(z) {
        source(".\\Rcode\\ClusterReps.R")
        ClusterReps(zYr[z],zMo[z],dThresh)
}
```

| | |
|---|---|
| FILE: | layerMapSource.r |
| Purpose: | Original dot-and-link maps code. |

```
DrawSquares <- function(xcen,ycen,width) {
 psq <- matrix(NA,ncol=2,nrow=5)
 psq[1,1] <- xcen-width; psq[1,2] <- ycen-width
 psq[2,1] <- xcen-width; psq[2,2] <- ycen+width
 psq[3,1] <- xcen+width; psq[3,2] <- ycen+width
 psq[4,1] <- xcen+width; psq[4,2] <- ycen-width
 return(psq)
}

DrawLayerMap <- function(MainTitle,
                                    drawGrid, paletteCluster, positionCluster,
                UtilityPalette, LandPalette, SeaPalette,
                dotSize, connectorLineWidth, latMin, latMax, lonMin, lonMax,
                                    linesOn = TRUE, focusClusters=NA, focusDotSize=1.5) {
        longitudeGrid <- c(-180,180)
        latitudeGrid  <- c(-90,90)
        myMapColor = UtilityPalette[2] # Black
    myMapFillColor = UtilityPalette[5] # light Grey
        dotSize = 1.25

        plot(longitudeGrid,latitudeGrid,type="n",main=MainTitle,xlab="",ylab="",
        xaxs="i",yaxs="i",axes=FALSE)

        mtext(side=3,line=.5,"Based on NASA AIRS L3Q Datasets")

        polygon(x=c(-180,-180,180,180,NA),y=c(-90,90,90,-90,NA),col=gray(0.2),border=NA)
#grey
    map(add=TRUE,fill=TRUE,col=gray(0.6),lwd=1,resolution=0)

        axis(side=1,at=seq(-180,180,by=30),tck=1,col='#808080',
        mgp=c(2,.1,0),col.axis='black')
        mtext(side=1,line=1.5,'Degrees Longitude')

        axis(side=2,at=seq(-90,90,by=30),tck=1,col='#808080',
        mgp=c(2,.3,0),col.axis='black',las=1)
        mtext(side=2,line=1.6,'Degrees Latitude')
```

```
	#dot pass
	for (lat in (latMin:latMax) ) {
		for (lon in (lonMin:lonMax) ) {
			dC = drawGrid[lat,lon]
			if (dC != NODATA) {
				xcen = (((lon-1)*5)+2.5)-180
				ycen = (((lat-1)*5)+2.5) -90
				if (paletteCluster[dC] == LANDPAL) {
					useCol = LandPalette[positionCluster[dC]]
				} else {
					useCol = SeaPalette[positionCluster[dC]]
				} #land or sea
				inFocus = which(focusClusters == dC)
				if (length(inFocus) > 0 ) {
					useDotSize = focusDotSize
				} else {
					useDotSize = dotSize
				}
				draw.circle(xcen,ycen,useDotSize,border=useCol,col=useCol)
			}# if there is data
		} # lon
	} #lat

	# linepass
	if (linesOn) {
		for (lat in ((latMin+1):(latMax-1)) ) { # need to write special north-south loop for
min/max
			for (lon in ((lonMin+1):(lonMax-1)) ) { # need to write special east-west
loop for min-max
				dC = drawGrid[lat,lon]
				if (dC == NODATA) { next }
				xcen = (((lon-1)*5)+2.5)-180
				ycen = (((lat-1)*5)+2.5) -90
				if (paletteCluster[dC] == LANDPAL) {
					useCol = LandPalette[positionCluster[dC]]
				} else {
					useCol = SeaPalette[positionCluster[dC]]
				} #land or sea

				#check all directions
```

```
                                    nW = drawGrid[(lat+1),(lon-1)]
                                    nN = drawGrid[(lat+1),(lon)]
                                    nE = drawGrid[(lat+1),(lon+1)]

                                    sW = drawGrid[(lat-1),(lon-1)]
                                    sS = drawGrid[(lat-1),(lon)]
                                    sE = drawGrid[(lat-1),(lon+1)]

                                    eE = drawGrid[(lat),(lon+1)]
                                    wW = drawGrid[(lat),(lon-1)]

                                    if (dC == nW) { segments(xcen,ycen,xcen-
5,ycen+5,col=useCol,lwd=2) }
                                    if (dC == nN) { segments(xcen,ycen,xcen
,ycen+5,col=useCol,lwd=2) }
                                    if (dC == nE) {
segments(xcen,ycen,xcen+5,ycen+5,col=useCol,lwd=2) }

                                    if (dC == sW) { segments(xcen,ycen,xcen-5,ycen-
5,col=useCol,lwd=2) }
                                    if (dC == sS) { segments(xcen,ycen,xcen  ,ycen-
5,col=useCol,lwd=2) }
                                    if (dC == sE) { segments(xcen,ycen,xcen+5,ycen-
5,col=useCol,lwd=2) }

                                    if (dC == eE) { segments(xcen,ycen,xcen+5,ycen
,col=useCol,lwd=2) }
                                    if (dC == wW) { segments(xcen,ycen,xcen-5,ycen
,col=useCol,lwd=2) }
                            } #lon
                    } #lat
          }# endif
          abline(h=seq(from=-60,to=60,by=30),v=seq(from=-
150,to=150,by=30),col="#CCCCCC80") #light grey grid
          map(add=TRUE,fill=FALSE,col=MyMapColor,lwd=1, interior=FALSE)
}#DONE
```

| | |
|---|---|
| FILE: | InteractiveClusteringFuncs.r |
| Purpose: | Miscellaneous mapping and utility functions . |

```
#Graphical Maps
#       zWin = drawMaps(...)            uses parameters to draw maps for the defined layers
#       zWin = clearMaps()              closes all windows in zWin

#Graphical Data
#       plotCluster2(clusterNo, ...)    generates L3QPlot from given cluster number

#Windows
#       qYM = getActiveWindow()             uses zWin, zYr, zMo to return "YYYY.MM" for
active window
#       setActiveWindow()                   given aYr, aMo, uses zWin, zYr, zMo to set
active window
#       qL = getActiveLayer()           given zWin, zYr, zMo returns index of the active layer in
zWin

#Focus
#       focusList = addFocus(focusList)         uses selectCell to add a cluster to the focus list
fL
#       focusList = clearFocus()                clears the focus list fL

#Utility
#       c = selectCell()                returns lat/lon index of clicked point in currently active
window. minimal error handling
#       c = selectCluster()             returns a cluster

getActiveWindow <- function() {
# Assumes parallel arrays zYr, zMo, zWin for each layer
        aWin = dev.cur()
        aLayer = which(zWin == aWin)
        return(paste(zYr[aLayer],zMo[aLayer],sep="."))
}

setActiveWindow <- function(aYr, aMo) {
# Assumes parallel arrays zYr, zMo, zWin for each layer
        dWin = dev.cur();
        yFind = which(zYr == aYr)
```

```
	mFind = which(zMo[yFind] == aMo)
	aLayer = yFind[mFind]
	aWin = zWin[aLayer]
	dev.set(aWin)
	return(getActiveWindow())
}

getActiveLayer <- function() {
# Assumes parallel arrays zYr, zMo, zWin for each layer
	aWin = dev.cur()
	aLayer = which(zWin == aWin)
	return(aLayer)
}

selectCell <- function() {
	p = locator(n=1,type="n")
	retVal = p
	retVal$y = ((p$x+180) %/% 5) +1 # map to 1-72
	retVal$x = ((p$y+ 90) %/% 5) +1 # range now 1-36
	return(retVal)
}

selectCluster <- function() {
	tL = getActiveLayer()
	tC = selectCell()
	tRet = layeredClusters[tL, tC$x, tC$y]
}

clearMaps <- function() {
# Assumes parallel arrays zYr, zMo, zWin for each layer
	for ( i in 1:zLen) {
		dev.off(zWin[i])
		zWin[i] = NA
	}
	return( zWin )
}

drawMaps <- function() {
# uses lots of globals...
	for (layer in (1:numLayers) ) {
		windows(width=winWidth,height=winHeight)
```

```
                zWin[layer] = dev.cur()
                dynamicLabel = sprintf('%i.%02i',zYr[layer],zMo[layer])
                mainLabel = paste("Cluster Map ",dynamicLabel,sep="")
                DrawLayerMap(MainTitle = mainLabel, drawGrid = layeredClusters[layer,,],
                        paletteCluster = assignClust, positionCluster = assignPalPos,
                        UtilityPalette = UtilityPalette, LandPalette = LandPalette, SeaPalette =
SeaPalette,
                        dotSize = 1, connectorLineWidth = 1, latMin = 2, latMax = 35, lonMin =
1, lonMax = 72,
                        linesOn = lineStatus, focusClusters = focusList, focusDotSize = 2.5)
        }
        return( zWin )
}

clearFocus <- function() {
        return(NA)
}

addFocus <- function(focusList) {
        tcell = selectCell()
        tlayer = getActiveLayer()
        tClust = layeredClusters[tlayer, tcell$x,  tcell$y]
        if (length(focusList) == 1 && is.na(focusList)) {
                fL = tClust
                } else {
                fL = c(focusList,tClust)
        }
        return(fL)
}

plotCluster <- function(clusterNo, scalingMode=0) {
        wNC = 0
        for (i in 1:zLen) { # counting pass
                xYr = zYr[i]
                xMo = zMo[i]
                xFnameHead = sprintf('%i.%02i.',xYr,xMo)
        fName = paste(xFnameHead,"36x72.int16.NumClusters",sep="")
                if (!file.exists(fName)) { return(0) }
                binfile = file(fName,'rb')
                inBin = readBin(binfile,integer(), size=2, n=(36*72))
                close(binfile)
```

```
		xNC = array(data=inBin,dim=c(36,72))
		for ( x in 1:36 ) {
			for ( y in 1:72 ) {
				if (layeredClusters[i,x,y] == clusterNo ) {
					wNC = wNC + xNC[x,y]
				}
			}
		}
}
# now we have a count we can declare various arrays
physVec = matrix(nrow=wNC, ncol=35)
numObs = array(dim=wNC)
years  = numObs
months = years
lats	= months
lons   = lats
cols   = lons

wPtr = 0
for (i in 1:zLen) { # counting pass
		xYr = zYr[i]
		xMo = zMo[i]
		xFnameHead = sprintf('%i.%02i.',xYr,xMo)

fName = paste(xFnameHead,"36x72.int16.NumClusters",sep="")
		if (!file.exists(fName)) { return(0) }
		binfile = file(fName,'rb')
		inBin = readBin(binfile,integer(), size=2, n=(36*72))
		close(binfile)
		xNC = array(data=inBin,dim=c(36,72))

		fName = paste(xFnameHead,"100x36x72.int16.NumObsInCluster",sep="")
		if (!file.exists(fName)) { return(0) }
		binfile = file(fName,'rb')
		inBin = readBin(binfile,integer(), size=2, n=(100*36*72))
		close(binfile)
		xNOC = array(data=inBin,dim=c(100,36,72))


		fName = paste(xFnameHead,"100x35x36x72.single.PhysicalValues",sep="")
		if (!file.exists(fName)) { return(0) }
```

```
                binfile = file(fName,'rb')
                inBin = readBin(binfile,numeric(), size=4, n=(100*35*36*72))
                close(binfile)
                xPV = array(data=inBin,dim=c(100,35,36,72))



                for ( x in 1:36 ) {
                        for ( y in 1:72 ) {
                                if (layeredClusters[i,x,y] == clusterNo ) {
                                        #loop it up!
                                        nC = xNC[x,y]
                                        for (k in 1:nC) {
                                                wPtr = wPtr + 1
                                                physVec[wPtr,] = xPV[k, ,x,y]
                                                numObs[wPtr]   = xNOC[k,x,y]
                                                years[wPtr]    = zYr[i]
                                                months[wPtr]   = zMo[i]
                                                lats[wPtr]     =  x  #(5*(x-1))-  87.5
                                                lons[wPtr]           =  y  #(5*(y-1))- 178.5
                                                if (assignClust[clusterNo] == LANDPAL) {
                                                        cols[wPtr] =
LandPalette[assignPalPos[clusterNo]]
                                                } else {
                                                        cols[wPtr] =
SeaPalette[assignPalPos[clusterNo]]
                                                } #land or sea
                                        } # end k
                                } # end if
                        } # end y
                } # end x
        } # end i



        L3QPlot(scalingMode = scalingMode, numA = wNC , numB = 0, physVec=physVec,
numObs = numObs,
                years = years, months = months,
                lats=lats , lons=lons, cols=cols, map=p36x72_land )

}
```

```
plotCluster2 <- function(clusterNo, scalingMode=0) {
	wNC = 0
	for (i in 1:zLen) { # counting pass
		xYr = zYr[i]
		xMo = zMo[i]
		xFnameHead = sprintf('%i.%02i.',xYr,xMo)
	fName = paste(xFnameHead,"36x72.int16.NumClusters",sep="")
		if (!file.exists(fName)) { return(0) }
		binfile = file(fName,'rb')
		inBin = readBin(binfile,integer(), size=2, n=(36*72))
		close(binfile)
		xNC = array(data=inBin,dim=c(36,72))
		for ( x in 1:36 ) {
			for ( y in 1:72 ) {
				if (layeredClusters[i,x,y] == clusterNo ) {
					wNC = wNC + xNC[x,y]
				}
			}
		}
	}
	# now we have a count we can declare various arrays
	physVec = matrix(nrow=wNC, ncol=35)
	numObs = array(dim=wNC)
	years  = numObs
	months = years
	lats	= months
	lons   = lats
	cols   = lons

	wPtr = 0
	for (i in 1:zLen) { # counting pass
		xYr = zYr[i]
		xMo = zMo[i]
		xFnameHead = sprintf('%i.%02i.',xYr,xMo)

	fName = paste(xFnameHead,"36x72.int16.NumClusters",sep="")
		if (!file.exists(fName)) { return(0) }
		binfile = file(fName,'rb')
		inBin = readBin(binfile,integer(), size=2, n=(36*72))
		close(binfile)
		xNC = array(data=inBin,dim=c(36,72))
```

```
fName = paste(xFnameHead,"100x36x72.int16.NumObsInCluster",sep="")
if (!file.exists(fName)) { return(0) }
binfile = file(fName,'rb')
inBin = readBin(binfile,integer(), size=2, n=(100*36*72))
close(binfile)
xNOC = array(data=inBin,dim=c(100,36,72))


fName = paste(xFnameHead,"100x35x36x72.single.PhysicalValues",sep="")
if (!file.exists(fName)) { return(0) }
binfile = file(fName,'rb')
inBin = readBin(binfile,numeric(), size=4, n=(100*35*36*72))
close(binfile)
xPV = array(data=inBin,dim=c(100,35,36,72))



for ( x in 1:36 ) {
        for ( y in 1:72 ) {
                if (layeredClusters[i,x,y] == clusterNo ) {
                        #loop it up!
                        nC = xNC[x,y]
                        for (k in 1:nC) {
                                wPtr = wPtr + 1
                                physVec[wPtr,] = xPV[k, ,x,y]
                                numObs[wPtr]   = xNOC[k,x,y]
                                years[wPtr]    = zYr[i]
                                months[wPtr]   = zMo[i]
                                lats[wPtr]     =  x  #(5*(x-1))-  87.5
                                lons[wPtr]           =  y  #(5*(y-1))- 178.5
                                if (assignClust[clusterNo] == LANDPAL) {
                                        cols[wPtr] =
LandPalette[assignPalPos[clusterNo]]
                                } else {
                                        cols[wPtr] =
SeaPalette[assignPalPos[clusterNo]]
                                } #land or sea
                        } # end k
                } # end if
        } # end y
```

```
                } # end x
        } # end i

        tLim = array(data=0,dim=c(11,2))
        wLim = array(data=0,dim=c(11,2))
        tLim[,1] = 200
        tLim[,2] = c( 450, 400,400,400,400, 300,300,300, 275,275,275)
        wLim[,2]=  c(  100,50,50, 50,50,50, 10,10,10, 5, 5)

        L3QPlot2(numA = wNC , numB = 0, physVec=physVec, numObs = numObs,
                years = years, months = months,
                lats=lats , lons=lons, cols=cols, map=p36x72_land, tLim=tLim, wLim=wLim ,
cTitle=clusterNo )

}

plotClusterF <- function(clusterNo, scalingMode=0, filterString, in_tlim=NA, in_wlim=NA) {

        wNC = 0

        for (i in 1:zLen) { # counting pass

                xYr = zYr[i]

                xMo = zMo[i]

                xFnameHead = sprintf('%i.%02i.',xYr,xMo)

        fName = paste(xFnameHead,"36x72.int16.NumClusters",sep="")

                if (!file.exists(fName)) { return(0) }

                binfile = file(fName,'rb')

                inBin = readBin(binfile,integer(), size=2, n=(36*72))

                close(binfile)

                xNC = array(data=inBin,dim=c(36,72))

                for ( x in 1:36 ) {

                        for ( y in 1:72 ) {

                                if (layeredClusters[i,x,y] == clusterNo ) {
```

```
                                        wNC = wNC + xNC[x,y]
                                }
                        }
                }
}
# now we have a count we can declare various arrays
physVec = matrix(nrow=wNC, ncol=35)
numObs = array(dim=wNC)
years  = numObs
months = years
lats       = months
lons   = lats
cols   = lons

wPtr = 0
for (i in 1:zLen) { # counting pass
        xYr = zYr[i]
        xMo = zMo[i]
        xFnameHead = sprintf('%i.%02i.',xYr,xMo)

fName = paste(xFnameHead,"36x72.int16.NumClusters",sep="")
        if (!file.exists(fName)) { return(0) }
        binfile = file(fName,'rb')
        inBin = readBin(binfile,integer(), size=2, n=(36*72))
```

```
close(binfile)
xNC = array(data=inBin,dim=c(36,72))

fName = paste(xFnameHead,"100x36x72.int16.NumObsInCluster",sep="")
if (!file.exists(fName)) { return(0) }
binfile = file(fName,'rb')
inBin = readBin(binfile,integer(), size=2, n=(100*36*72))
close(binfile)
xNOC = array(data=inBin,dim=c(100,36,72))

fName = paste(xFnameHead,"100x35x36x72.single.PhysicalValues",sep="")
if (!file.exists(fName)) { return(0) }
binfile = file(fName,'rb')
inBin = readBin(binfile,numeric(), size=4, n=(100*35*36*72))
close(binfile)
xPV = array(data=inBin,dim=c(100,35,36,72))

for ( x in 1:36 ) {
        for ( y in 1:72 ) {
                if (layeredClusters[i,x,y] == clusterNo ) {
                        #loop it up!
```

```
                                        nC = xNC[x,y]
                                        for (k in 1:nC) {
                                                wPtr = wPtr + 1
                                                physVec[wPtr,] = xPV[k, ,x,y]
                                                numObs[wPtr]   = xNOC[k,x,y]
                                                years[wPtr]    = zYr[i]
                                                months[wPtr]   = zMo[i]
                                                lats[wPtr]     =  x  #(5*(x-1))-  87.5
                                                lons[wPtr]           =  y  #(5*(y-1))- 178.5
                                                if (assignClust[clusterNo] == LANDPAL) {
                                                        cols[wPtr] =
LandPalette[assignPalPos[clusterNo]]
                                                } else {
                                                        cols[wPtr] =
SeaPalette[assignPalPos[clusterNo]]
                                                } #land or sea
                                        } # end k
                                } # end if
                        } # end y
                } # end x
        } # end i

        tLim = array(data=0,dim=c(11,2))
        if (length(in_tlim) != 11) {
                tLim[,1] = 200
```

```
        tLim[,2] = c( 375, 375,375,375,375, 300,300,300, 275,275,275)
    } else {
        tLim[,1] = 80
        tLim[,2] = in_tlim
    }
    wLim = array(data=0,dim=c(11,2))
    if (length(in_wlim) != 11) {
        wLim[,2]=  c(  100,50,50, 50,50,50, 10,10,10, 5, 5)
    } else {
        wLim[,2] = in_wlim
    }

    L3QPlotF(numA = wNC , numB = 0, physVec=physVec, numObs = numObs,
        years = years, months = months,
        lats=lats , lons=lons, cols=cols, map=p36x72_land, tLim=tLim, wLim=wLim ,
cTitle=clusterNo, filter=filterString )

}
```

| | |
|---|---|
| FILE: | MultiClusterSets |
| Purpose: | R code & example for using it to produce multi-cluster set maps . |

```
clusterCaricatures4 <- function(bBot = 1, bTop = 1, bLeft = 1, bRight = 24, bMid = 1, moBox = 4,
yrBox = 3,
                                cTitle, cList,
                                layeredClusters, zYr, zMo,
                                paletteCluster,
                                positionCluster,
                                UtilityPalette,
                                ShortLandPalette,
                                ShortSeaPalette, map, mapThresh = 0.3) {
        nPanes = length(zYr) # of clusters to be displayed
        nLayers = length(zYr)
        monthLabel = c("Jan","Feb","Mar", "Apr","May","Jun", "Jul","Aug","Sep",
"Oct","Nov","Dec")

        palLength = min(length(ShortLandPalette),length(ShortSeaPalette))

        vStripe = 36
        useCol = array(data=0,dim=c(length(cList)))
        lPref = useCol
        sPref = useCol

        numClusts = min(length(cList), palLength)

        # pick the preferred palette
        for (i in (1:numClusts)) {
                if (sea_Clust[cList[i]] >= landClust[cList[i]]) {
                        sPref[i] = 1
                } else {
                        lPref[i] = 1
                }
        }

        #position on the palette?
        lp = which(lPref>0)
        sp = which(sPref>0)
```

```
	lPosPref = clusterPalette[LANDPAL,cList[lp],PAL_END] /
clusterPalette[LANDPAL,cList[lp],PAL_CNT]
	sPosPref = clusterPalette[SEA_PAL,cList[sp],PAL_END] /
clusterPalette[SEA_PAL,cList[sp],PAL_CNT]
	lp_Ord = order(lPosPref)
	sp_Ord = order(sPosPref)
	# cList[sp[sp_Ord]] or cList[lp[lp_Ord]] to assign colors

	numPositions = palLength
	numSeas = length(sp)
	if (numSeas > 0) {
		fairPlaces = trunc((numPositions / numSeas ) * c(1:numSeas))
		useCol[sp[sp_Ord]] = ShortSeaPalette[fairPlaces]
	}

	numLand = length(lp)
	if (numLand > 0 ) {
		fairPlaces = trunc((numPositions / numLand ) * c(1:numLand))
		useCol[lp[lp_Ord]] = ShortLandPalette[fairPlaces]
	}

	xMax = 72 + bLeft + bRight + bMid
	yMax = nPanes * ( vStripe + bBot + bTop )
	#File	continent boundary
	#By	Daniel B. Carr
	#Date	October 9, 2007
	#dir = "C:\\Documents and Settings\\Daniel Carr\\My Documents\\JPL\\AIRS Data Four
Winters\\"
	fil = "continent.txt"
	#ans = read.table(paste(dir,fil,sep=''),colClasses=c('numeric','numeric'))
	ans = read.table(paste(fil,sep=''),colClasses=c('numeric','numeric'))
	rx = range(ans[,1],na.rm=T)
	ry = range(ans[,2],na.rm=T)
	# continents
	BlueColors <- c( #ColorBrewer Blues 9:9-3
			8	,69,	148,
			33	,113,	181,
			66	,146,	198,
			107	,174,	214,
			158	,202,	225,
			198	,219,	239
```

```
                        )
        bluePal = rgb(matrix(BlueColors,byrow=TRUE,ncol=3),maxColorValue=255)

        windows()
        par(mai=c(0.1,0.1,0.1,0.1));
        plot(c(0,xMax), c(0,yMax), axes = FALSE, xlab="", ylab="",type="n")
        title(main=cTitle,line=-1)

        for (pane in 1:nPanes) {
                yShift = (pane-1)*(vStripe + bBot + bTop)
                paneMap = matrix(data=FALSE,ncol=72,nrow=36)
                paneCount = 0

                # draw the map
                leftSide = 0 + bLeft; rightSide = leftSide+72;
                botMap  = yShift+bBot; topMap = botMap+36;
                xBord = c(leftSide, leftSide, rightSide, rightSide, leftSide)
                yBord = c(botMap, topMap, topMap, botMap, botMap)

                dX = (1/72)*(rightSide-leftSide)
                dY = (1/36)*(topMap-botMap)
                mapX = leftSide         + (rightSide-leftSide)* (ans[,1]+180)/360
                mapY = botMap           + (topMap-botMap)    * (ans[,2]+90)/180
                polygon(xBord, yBord,col=bluePal[6])
                #polygon(x=mapX,y=mapY,col=gray(0.8),density=0)
                polygon(x=mapX,y=mapY,border=gray(0.4),col=gray(0.8),density=NA)

                for (i in 1:36) {
                        for (j in 1:72) {
                                xC = leftSide + dX/2 + (j-1)*dX
                                yC = botMap   + dY/2 + (i-1)*dY
                                #if (map[i,j] > mapThresh) {
                                #       rect(xC-dX/2, yC-dY/2,xC+dX/2,yC+dY/2,col="light
grey",border=NA )
                                #} # mapThresh
                                for (k in (1:length(cList))) {
                                        if (layeredClusters[pane,i,j]==cList[k]) {
                                                rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col=useCol[k],border=NA )
                                                paneCount = paneCount +1
                                        } # paneMap
```

```
                                        }
                                } # j
                        } # i
                        polygon(xBord, yBord)

                        # draw the calendar
                        dLabel = sprintf("%s %d", monthLabel[zMo[pane]],zYr[pane])
                        paneString = sprintf("This map count = %d", paneCount)
                        text(rightSide,botMap+2*(topMap-botMap)/3,dLabel,pos=4)
                        text(rightSide,botMap+1*(topMap-botMap)/3,paneString,pos=4)
        } #pane
}
clusterCaricatures4Old <- function(bBot = 1, bTop = 1, bLeft = 1, bRight = 24, bMid = 1, moBox =
4, yrBox = 3,
                                        cTitle, cList,
                                        layeredClusters, zYr, zMo,
                                        paletteCluster,
                                        positionCluster,
                                        UtilityPalette,
                                        LandPalette,
                                        SeaPalette, map, mapThresh = 0.3) {
        nPanes = length(zYr) # of clusters to be displayed
        nLayers = length(zYr)

        vStripe = 36
        useCol = array(data=0,dim=c(length(cList)))

        for (i in (1:length(cList))) {
                if (assignClust[cList[i]] == LANDPAL) {
                        useCol[i] = LandPalette[assignPalPos[cList[i]]]
                } else {
                        useCol[i] = SeaPalette[assignPalPos[cList[i]]]
                } #land or sea
        }

        xMax = 72 + bLeft + bRight + bMid
        yMax = nPanes * ( vStripe + bBot + bTop )

        windows()
        par(mai=c(0.1,0.1,0.1,0.1));
        plot(c(0,xMax), c(0,yMax), axes = FALSE, xlab="", ylab="",type="n")
```

```
	title(main=cTitle,line=-1)

	for (pane in 1:nPanes) {
		yShift = (pane-1)*(vStripe + bBot + bTop)
		paneMap = matrix(data=FALSE,ncol=72,nrow=36)
		paneCount = 0

		# draw the map
		leftSide = 0 + bLeft; rightSide = leftSide+72;
		botMap  = yShift+bBot; topMap = botMap+36;
		xBord = c(leftSide, leftSide, rightSide, rightSide, leftSide)
		yBord = c(botMap, topMap, topMap, botMap, botMap)

		dX = (1/72)*(rightSide-leftSide)
		dY = (1/36)*(topMap-botMap)
		for (i in 1:36) {
			for (j in 1:72) {
				xC = leftSide + dX/2 + (j-1)*dX
				yC = botMap   + dY/2 + (i-1)*dY
				if (map[i,j] > mapThresh) {
					rect(xC-dX/2, yC-dY/2,xC+dX/2,yC+dY/2,col="light
grey",border=NA )
				} # mapThresh
				for (k in (1:length(cList))) {
					if (layeredClusters[pane,i,j]==cList[k]) {
						rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col=useCol[k],border=NA )
						paneCount = paneCount +1
					} # paneMap
				}
			} # j
		} # i
		polygon(xBord, yBord)
		# draw the calendar
		paneString = sprintf("Dataset count = %d", paneCount)
		text(rightSide,botMap+2*(topMap-
botMap)/3,zYr[pane]*100+zMo[pane],pos=4)
		text(rightSide,botMap+1*(topMap-botMap)/3,paneString,pos=4)
	} #pane
}
```

| | |
|---|---|
| FILE: | SetsOverTime.r |
| Purpose: | R code & example for using it to produce single cluster over time maps |

```
setDoppler2 <- function(bBot = 2, bTop = 1, bLeft = 1, bRight = 8, bMid = 1, moBox = 4, yrBox = 3,
                                    cTitle, cList,
                                    layeredClusters, zYr, zMo,
                                    paletteCluster,
                                    positionCluster,
                                    UtilityPalette,
                                    ShortLandPalette,
                                    ShortSeaPalette, map, mapThresh = 0.3) {
        nPanes = length(zYr) # of clusters to be displayed
        nLayers = length(zYr)
        monthLabel = c("Jan","Feb","Mar", "Apr","May","Jun", "Jul","Aug","Sep",
"Oct","Nov","Dec")

        setFootprints = array(data=0,dim=c(36,72))

        for (i in (1:nLayers)) { for (j in (1:36)) { for (k in (1:72)) {
                if (layeredClusters[i,j,k]==cList[1]) {
                        setFootprints[j,k] = setFootprints[j,k] + 1
                } # end if
        }}} # i j k

        #File    continent boundary
        #By      Daniel B. Carr
        #Date    October 9, 2007
        #dir = "C:\\Documents and Settings\\Daniel Carr\\My Documents\\JPL\\AIRS Data Four
Winters\\"
        fil = "continent.txt"
        #ans = read.table(paste(dir,fil,sep=''),colClasses=c('numeric','numeric'))
        ans = read.table(paste(fil,sep=''),colClasses=c('numeric','numeric'))
        rx = range(ans[,1],na.rm=T)
        ry = range(ans[,2],na.rm=T)
        # continents

        palLength = min(length(ShortLandPalette),length(ShortSeaPalette))

        vStripe = 36
        useCol = array(data=0,dim=c(length(cList)))
```

```
        lPref = useCol
        sPref = useCol

        numClusts = min(length(cList), palLength)

        # pick the preferred palette
        for (i in (1:numClusts)) {
                if (sea_Clust[cList[i]] >= landClust[cList[i]]) {
                        sPref[i] = 1
                } else {
                        lPref[i] = 1
                }
        }

        #position on the palette?
        lp = which(lPref>0)
        sp = which(sPref>0)
        lPosPref = clusterPalette[LANDPAL,cList[lp],PAL_END] /
clusterPalette[LANDPAL,cList[lp],PAL_CNT]
        sPosPref = clusterPalette[SEA_PAL,cList[sp],PAL_END] /
clusterPalette[SEA_PAL,cList[sp],PAL_CNT]
        lp_Ord = order(lPosPref)
        sp_Ord = order(sPosPref)
        # cList[sp[sp_Ord]] or cList[lp[lp_Ord]] to assign colors

        numPositions = palLength
        numSeas = length(sp)
        if (numSeas > 0) {
                fairPlaces = trunc((numPositions / numSeas ) * c(1:numSeas))
                useCol[sp[sp_Ord]] = ShortSeaPalette[fairPlaces]
        }

        numLand = length(lp)
        if (numLand > 0 ) {
                fairPlaces = trunc((numPositions / numLand ) * c(1:numLand))
                useCol[lp[lp_Ord]] = ShortLandPalette[fairPlaces]
        }

        GreenColors <- c( # ColorBrewer Greens 7:1-7
                        237     ,248    ,233,
                        199     ,233    ,192,
```

```
                    161     ,217    ,155,
                    116     ,196    ,118,
                    65      ,171    ,93,
                    35      ,139    ,69,
                    0       ,90     ,50
                    )
BlueColors <- c( #ColorBrewer Blues 9:9-3
                    8       ,69,    148,
                    33      ,113,   181,
                    66      ,146,   198,
                    107     ,174,   214,
                    158     ,202,   225,
                    198     ,219,   239
                    )
RedColors <- c( #ColorBrewer Reds 7:2-7
                    252     ,187,   161,
                    252     ,146,   114,
                    251     ,106,   74,
                    239     ,59,    44,
                    203     ,24,    29,
                    153     ,0,     13
                    )

miniPal = rgb(matrix(GreenColors,byrow=TRUE,ncol=3),maxColorValue=255)
redPal  = rgb(matrix(RedColors,byrow=TRUE,ncol=3),maxColorValue=255)
bluePal = rgb(matrix(BlueColors,byrow=TRUE,ncol=3),maxColorValue=255)


xMax = 72 + bLeft + bRight + bMid
yMax = nPanes * ( vStripe + bBot + bTop )

windows()
par(mai=c(0.1,0.1,0.1,0.1));
plot(c(0,xMax), c(0,yMax), axes = FALSE, xlab="", ylab="",type="n")
title(main=cTitle,line=-1)

for (pane in 1:nPanes) {
        yShift = (pane-1)*(vStripe + bBot + bTop)
        paneMap = matrix(data=FALSE,ncol=72,nrow=36)
        paneCount = 0
```

```
		# draw the map
		leftSide = 0 + bLeft; rightSide = leftSide+72;
		botMap  = yShift+bBot; topMap = botMap+36;
		pB = 0.2
		xBord = c(leftSide-pB, leftSide-pB, rightSide+pB, rightSide+pB, leftSide-pB)
		yBord = c(botMap-pB, topMap+pB, topMap+pB, botMap-pB, botMap-pB)

		dX = (1/72)*(rightSide-leftSide)
		dY = (1/36)*(topMap-botMap)

		mapX = leftSide		+ (rightSide-leftSide)* (ans[,1]+180)/360
		mapY = botMap		+ (topMap-botMap)    * (ans[,2]+90)/180

		polygon(xBord, yBord,col=bluePal[6])
		polygon(x=mapX,y=mapY,col=gray(0.8),border=gray(0.6))
		for (i in 1:36) {
			for (j in 1:72) {
				xC = leftSide + dX/2 + (j-1)*dX
				yC = botMap   + dY/2 + (i-1)*dY
				#if (map[i,j] > mapThresh) {
				#	rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col="grey",border=NA )
				#} # mapThresh
				for (k in (1:length(cList))) {
					if (layeredClusters[pane,i,j]==cList[k]) {
						rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col=miniPal[4], border=NA, density = NA) #density = 80, angle=45 )
						paneCount = paneCount +1
					} # paneMap
				} # k
				if (setFootprints[i,j] > 0) {
					if (setFootprints[i,j] == nPanes) { # part of the core
						rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col=miniPal[6],border=NA, density=NA )
					} else { # part of the union
						#redshift
						if (pane > 1) {
							for (k in (1:(pane-1))) {
								if
(layeredClusters[k,i,j]==cList[1] && layeredClusters[pane,i,j] != cList[1]) {
```

```
                                                                        #rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col=NA,border=redPal[3], density=NULL )
                                                                        xPoly = c(xC-dX/2, xC-
dX/2, xC+dX/2, xC+dX/2, xC-dX/2)
                                                                        yPoly = c(yC-dY/2,
yC+dY/2, yC+dY/2, yC-dY/2, yC-dY/2)
                                                                        for (v in (2:5)) {

        segments(x0=xPoly[v-1], y0=yPoly[v-1], x1=xPoly[v], y1=yPoly[v], col=redPal[3],lwd=1)
                                                                        }
                                                                }
                                                        }
                                                }
                                                #blueshift
                                                if (pane < nPanes) {
                                                        for (k in ((pane+1):nPanes)) {
                                                                if
(layeredClusters[k,i,j]==cList[1] && layeredClusters[pane,i,j] != cList[1]) {
                                                                        #rect(xC-dX/2, yC-
dY/2,xC+dX/2,yC+dY/2,col=NA,border=bluePal[3], density=NULL )
                                                                        xPoly = c(xC-dX/2, xC-
dX/2, xC+dX/2, xC+dX/2, xC-dX/2)
                                                                        yPoly = c(yC-dY/2,
yC+dY/2, yC+dY/2, yC-dY/2, yC-dY/2)
                                                                        for (v in (2:5)) {

        segments(x0=xPoly[v-1], y0=yPoly[v-1], x1=xPoly[v], y1=yPoly[v], col=bluePal[3],lwd=1)
                                                                        }
                                                                }
                                                        }
                                                }
                                        } # end if/else
                                } # end if/else
                        } # j
                } # i

                # draw the calendar
                paneString = sprintf("Current (Green) Set Members = %d", paneCount)
                dateString1 = sprintf("%s",monthLabel[zMo[pane]])
                dateString2 = sprintf("%4.0f",zYr[pane])
                text(rightSide,botMap+0.6*(topMap-botMap),dateString1,pos=4)
```

```
                text(rightSide,botMap+0.5*(topMap-botMap),dateString2,pos=4)
                #text(rightSide,botMap+0.45*(topMap-botMap),paneString,pos=4)

        } #pane
        legendL = c("All Maps", "Current Map", "Future Maps", "Past Maps")
        fillPal = c(miniPal[6], miniPal[4], bluePal[3], redPal[3])
        legend(x=0.3,y=2,horiz=TRUE,legend=legendL, fill=fillPal, bty="n")
}
#fL = c(20)
#myTitle = sprintf("Set over Time %s",fL[1])
#setDoppler2(cTitle = myTitle,cList = fL , layeredClusters = layeredClusters,
                                    #zYr = zYr, zMo = zMo,
                                    #paletteCluster = assignClust,
                                    #positionCluster = assignPalPos,
                                    #UtilityPalette = UtilityPalette,
                                    #ShortLandPalette = ShortLandPalette,
                                    #ShortSeaPalette = ShortSeaPalette,
                                    # map=p36x72_land)
```

| | |
|---|---|
| FILE: | L3QPlot.r |
| Purpose: | R code & example for using it to produce pitchfork charts . |

```
barLine <- function(x,y,wide,left,right,xNudge) {
        lineLeft  = x+xNudge*wide
        lineRight = x+(1-xNudge)*wide
        segments(lineLeft,y,lineRight,y)
        text(x,y,left,cex=0.5)
        text(x+wide,y,right,cex=0.5)
}
barLine32 <- function(x,y,wide,left,right,xNudge) {
        lineLeft  = x+xNudge*wide
        lineRight = x+(1-xNudge)*wide
        segments(lineLeft,y,lineRight,y)
        text(x,y,sprintf("%3.0f",left),cex=0.5)
        text(x+wide,y,sprintf("%3.0f",right),cex=0.5)
}
barLine31 <- function(x,y,wide,left,right,xNudge) {
        lineLeft  = x+xNudge*wide
        lineRight = x+(1-xNudge)*wide
        segments(lineLeft,y,lineRight,y)
        text(x,y,sprintf("%3.0f",left),cex=0.5)
        text(x+wide,y,sprintf("%3.0f",right),cex=0.5)
}

L3QPlotF <- function(howWide = 8, howHigh = 12,
                numA = 0, numB = 0, physVec , numObs ,
                years, months, lats, lons, cols, map, mapThresh=0.3,
                tLim, wLim, cTitle, filter) {
        windows(width=howWide,height=howHigh)
        yMax = 33; ySpace = 1.0; yGroup = 4;
        xMax = 28; xSpace = 1.0; mapBorder = 2;
        boxHigh = 1; boxWide = 7;
        xNudge = 0.1;
        yNudge = 0.4;

        par(mai=c(0.1,0.1,0.1,0.1));
        plot(c(0,xMax), c(0,yMax), axes = FALSE, xlab="", ylab="",type="n")
        if (nchar(filter) > 1) {
                title(main=sprintf("Cluster %s, %s", cTitle, filter),line=-0.5)
```

```
} else {
        title(main=sprintf("Cluster %s", cTitle),line=-0.5)
}

colA = 2; colB = 10; colC = 18;
baseAlts = 16; baseMeta = 7; baseClusterNo = 5;
baseMap = 2; mapHigh = 4;

lineBreaks = c(0,1, 3,4,5, 7,8,9, 11,12,13)
lineBreaks = lineBreaks*1.25;
ynudge = (lineBreaks[2]-lineBreaks[1])/2.0

leftTLabels = c(0,0, 0,0,0, 0,0,0, 0,0,0)
riteTLabels = c(1,1, 1,1,1, 1,1,1, 1,1,1)

leftWLabels = c(0,0, 0,0,0, 0,0,0, 0,0,0)
riteWLabels = c(1,1, 1,1,1, 1,1,1, 1,1,1)

leftCLabels = c(0,0, 0,0,0, 0,0,0, 0,0,0)
riteCLabels = c(1,1, 1,1,1, 1,1,1, 1,1,1)

leftMLabels = c(0,0,0,0)
riteMLabels = c(1,1,1,1)

pressureLevels = c(1000,925,  850,700,600, 500,400,300, 250,200,150)
metaLevels = c("Day%","Good%","Land%","nObs","Month","Year")

tMins = array(data=NA, dim=11);
tMaxs = array(data=NA, dim=11);
wMins = array(data=NA, dim=11);
wMaxs = array(data=NA, dim=11);
cMins = array(data=0, dim=11);
cMaxs = array(data=1, dim=11);
mMins = array(data=NA, dim=4); # land, good, day
mMaxs = array(data=NA, dim=4);

#Colorbrewer colors
vGrey  = c(150,150,150, 115,115,115, 82,82,82, 37,37,37)
vBlue  = c(127,205,187,  29,145,192, 12,44,132)
vPurple= c( )
vGreen = c( )
```

```
	#VecColors <- c()
	VecColors <- c( # ColorBrewer Grey, Red, Blue
			150,150,150,
			115,115,115,
			82,82,82,
			37,37,37,
			127,205,187,
			29,145,192,
			12,44,132,
			251,106,74,
			203,24,29,
			153,0,13
			)
	VecPalette <- rgb(matrix(c(VecColors),byrow=TRUE,ncol=3),maxColorValue=255)
	numVecs = numA+numB
	numColors = length(VecPalette)
	#repColors = sort(rep(seq(1:numColors),(numVecs/numColors)+1)) #V1, next version
will use custom breakpoints
	#repColors = repColors[1:numVecs]
	colBreaks = c(5,10,25,50, #greys
				100, 250, 500, #reds
				1000, 2500, max(max(numObs[1:(numA+numB)]),10000))
	repColors = array(data=1,dim=c(numVecs))
	nobsOrder = order(numObs[1:(numA+numB)])
	j = 1;
	for (i in 1:numVecs) {
		if (numObs[nobsOrder[i]] > colBreaks[j] ) { j = j+1; }
		repColors[i] = j
	}

	#filter in unscaled units
	useFilter = FALSE
	if (nchar(filter) > 1 ) {
		useFilter = TRUE
		fToks = unlist(strsplit(filter," "))
		if (fToks[1] == "nobs") {
			if (fToks[2] == "<") {
				fLim = as.numeric(fToks[3])
				fF = numObs < fLim
			}
```

```
                        if (fToks[2] == ">=") {
                                fLim = as.numeric(fToks[3])
                                fF = numObs >= fLim
                        }
                }
        }

        for(i in 1:11) {
                tMins[i] = min(min(physVec[1:(numA+numB),i]),tLim[i,1])
                tMaxs[i] = max(max(physVec[1:(numA+numB),i]),tLim[i,2])
                wMins[i] = min(min(physVec[1:(numA+numB),i+11]),wLim[i,1])
                wMaxs[i] = max(max(physVec[1:(numA+numB),i+11]),wLim[i,2])
                if (i > 1) {
                        cMins[i] = 0
                        cMaxs[i] = 1
                }
                if ( i < 4 ) {
                        mMins[i] = 0
                        mMaxs[i] = 1
                }
        }

        mMins[4] = 0
        mMaxs[4] = max(numObs[1:(numA+numB)])


        for(i in 1:11) {
                physVec[1:(numA+numB),i] = (physVec[1:(numA+numB),i] - tMins[i] ) / (tMaxs[i]
- tMins[i])
                physVec[1:(numA+numB),i+11] = (physVec[1:(numA+numB),i+11] - wMins[i] ) /
(wMaxs[i] - wMins[i])
                if (i > 1) {
                        physVec[1:(numA+numB),i+21] = (physVec[1:(numA+numB),i+21] -
cMins[i] ) / (cMaxs[i] - cMins[i])
                }
                if (i < 4) {
                        physVec[1:(numA+numB),i+32] = (physVec[1:(numA+numB),i+32] -
mMins[i] ) / (mMaxs[i] - mMins[i])
                }
        }
        #nobs
```

```
	numObs = (numObs - mMins[4]) / (mMaxs[4] - mMins[4])
	#years
	yrMin = min(years[1:(numA+numB)])
	yrMax = max(years[1:(numA+numB)])
	if (yrMin == yrMax) { yrMin <- yrMax -1 }
	years = (years - yrMin ) / (yrMax - yrMin)
	#months
	months = ( months - 1) / 11

	text(colA,baseAlts+lineBreaks[11]+1,"Temp:K",pos=4)
	text(colB,baseAlts+lineBreaks[11]+1,"Water:Spc Hum",pos=4)
	text(colC,baseAlts+lineBreaks[11]+1,"  Cloud:frac",pos=4)


	for( i in 1:11) {
		#text(colC+boxWide+1.5,baseAlts+lineBreaks[i],paste(pressureLevels[i],"
mB"),cex=0.5)
		text(colA-1.5,baseAlts+lineBreaks[i],paste(pressureLevels[i],"mB"),cex=0.5)
	}
	for (i in 1:6) {
		text(colB+boxWide+1.5,baseMeta+i,metaLevels[i],cex=0.5)
	}


	#temperature 1:11 lower to upper atmosphere
	leftEnd =  colA +xNudge    *boxWide
	rightEnd = colA +(1-xNudge)*boxWide
	for (i in 1:11) {
		barLine32(colA,baseAlts+lineBreaks[i],boxWide,tMins[i],tMaxs[i],xNudge)
		if (i>1) {
			for ( j in 1:(numA+numB) ) {
				jj = nobsOrder[j]
				if ( !useFilter || (useFilter && fF[jj]) ) {
					x1 = leftEnd + physVec[jj,(i)]*(rightEnd-leftEnd)
					x0 = leftEnd + physVec[jj,(i-1)]*(rightEnd-leftEnd)
					segments(x0,baseAlts+lineBreaks[i-
1],x1,baseAlts+lineBreaks[i],col=VecPalette[repColors[j]])
				}
			}
		}
	}
```

```
        #Water Vapor 12:22 lowerr to upper
        leftEnd =  colB +xNudge    *boxWide
        rightEnd = colB +(1-xNudge)*boxWide
        for (i in 1:11) {

        barLine31(colB,baseAlts+yNudge+lineBreaks[i],boxWide,wMins[i],wMaxs[i],xNudge)
                if (i>1) {
                        for ( j in 1:(numA+numB) ) {
                                jj = nobsOrder[j]
                                if ( !useFilter || (useFilter && fF[jj]) ) {
                                        x1 = leftEnd + physVec[jj,(i+11)]*(rightEnd-leftEnd)
                                        x0 = leftEnd + physVec[jj,(i+10)]*(rightEnd-leftEnd)
                                        segments(x0,baseAlts+yNudge+lineBreaks[(i-
1)],x1,baseAlts+yNudge+lineBreaks[i],col=VecPalette[repColors[j]])
                                }
                        }
                }

        }

        #Cloud Cover 23:33 lower to upper
        leftEnd =  colC +xNudge    *boxWide
        rightEnd = colC +(1-xNudge)*boxWide
        for (i in 1:10) {

        barLine(colC,baseAlts+yNudge+lineBreaks[i],boxWide,leftCLabels[i],riteCLabels[i],xNudg
e)
                if (i>1) {
                        for ( j in 1:(numA+numB) ) {
                                jj = nobsOrder[j]
                                if ( !useFilter || (useFilter && fF[jj]) ) {
                                        x1 = leftEnd + physVec[jj,(i+22)]*(rightEnd-leftEnd)
                                        x0 = leftEnd + physVec[jj,(i+21)]*(rightEnd-leftEnd)
                                        segments(x0,baseAlts+yNudge+lineBreaks[(i-
1)],x1,baseAlts+yNudge+lineBreaks[i],col=VecPalette[repColors[j]])
                                }
                        }
                }
        }
```

```
        #MetaData
        leftEnd =  colB +xNudge    *boxWide
        rightEnd = colB +(1-xNudge)*boxWide
        for (i in 1:6) {
                if (i == 1) { #DayFrac
                        barLine(colB,baseMeta + i,boxWide,0,1,xNudge) # bounds for
metadatas
                        for ( j in 1:(numA+numB) ) {
                                jj = nobsOrder[j]
                                if ( !useFilter || (useFilter && fF[jj]) ) {
                                        x0 = leftEnd + physVec[jj,35]*(rightEnd-leftEnd)
                                        x1 = leftEnd + physVec[jj,34]*(rightEnd-leftEnd)

        segments(x0,baseMeta+i,x1,baseMeta+i+1,col=VecPalette[repColors[j]])
                                }
                        }
                }
                if (i == 2) { #Good
                        barLine(colB,baseMeta + i,boxWide,0,1,xNudge) # bounds for
metadatas
                        for ( j in 1:(numA+numB) ) {
                                jj = nobsOrder[j]
                                if ( !useFilter || (useFilter && fF[jj]) ) {
                                        x0 = leftEnd + physVec[jj,34]*(rightEnd-leftEnd)
                                        x1 = leftEnd + physVec[jj,33]*(rightEnd-leftEnd)

        segments(x0,baseMeta+i,x1,baseMeta+i+1,col=VecPalette[repColors[j]])
                                }
                        }
                }

                if ( i == 3 ) { # Land
                        barLine(colB,baseMeta + i,boxWide,0,1,xNudge) # bounds for
metadatas
                        for ( j in 1:(numA+numB) ) {
                                jj = nobsOrder[j]
                                if ( !useFilter || (useFilter && fF[jj]) ) {
                                        x0 = leftEnd + physVec[jj,33]*(rightEnd-leftEnd)
                                        x1 = leftEnd + numObs[jj]*(rightEnd-leftEnd)

        segments(x0,baseMeta+i,x1,baseMeta+i+1,col=VecPalette[repColors[j]])
```

```
                        }
                    }
                }
                if ( i == 4 ) { # nObs
                    barLine(colB,baseMeta + i,boxWide,mMins[4],mMaxs[4],xNudge) #
bounds for metadatas
                    for ( j in 1:(numA+numB) ) {
                        jj = nobsOrder[j]
                        if ( !useFilter || (useFilter && fF[jj]) ) {
                            x0 = leftEnd + numObs[jj]*(rightEnd-leftEnd)
                            x1 = leftEnd + months[jj]*(rightEnd-leftEnd)

        segments(x0,baseMeta+i,x1,baseMeta+i+1,col=VecPalette[repColors[j]])
                        }
                    }
                }
                if ( i == 5 ) { # month
                    barLine(colB,baseMeta + i,boxWide,1,12,xNudge) # bounds for
metadatas
                    for ( j in 1:(numA+numB) ) {
                        jj = nobsOrder[j]
                        if ( !useFilter || (useFilter && fF[jj]) ) {
                            x0 = leftEnd + months[jj]*(rightEnd-leftEnd)
                            x1 = leftEnd + years[jj]*(rightEnd-leftEnd)

        segments(x0,baseMeta+i,x1,baseMeta+i+1,col=VecPalette[repColors[j]])
                        }
                    }
                }
                if ( i == 6) { # year
                    barLine(colB,baseMeta + i,boxWide,yrMin,yrMax,xNudge) # bounds for
metadatas
                }

        }

        #join the branches to the trunk
        for ( j in 1:(numA+numB) ) {
                        jj = nobsOrder[j]
                        if ( !useFilter || (useFilter && fF[jj]) ) {
                            leftEnd =  colB +xNudge    *boxWide
```

```
                                        rightEnd = colB +(1-xNudge)*boxWide
                                        x0 = leftEnd + years[jj]*(rightEnd-leftEnd)
                                        x2 = leftEnd + physVec[jj,12]*(rightEnd-leftEnd)

                                        leftEnd =  colA +xNudge    *boxWide
                                        rightEnd = colA +(1-xNudge)*boxWide
                                        x1 = leftEnd + physVec[jj,1]*(rightEnd-leftEnd)

                                        leftEnd =  colC +xNudge    *boxWide
                                        rightEnd = colC +(1-xNudge)*boxWide
                                        x3 = leftEnd + physVec[jj,23]*(rightEnd-leftEnd)


        segments(x0,baseMeta+6,x1,baseAlts+lineBreaks[1],col=VecPalette[repColors[j]])

        segments(x0,baseMeta+6,x2,baseAlts+yNudge+lineBreaks[1],col=VecPalette[repColors[j
]])

        segments(x0,baseMeta+6,x3,baseAlts+yNudge+lineBreaks[1],col=VecPalette[repColors[j
]])
                                }
        }

        #land map for reference / others
        BlueColors <- c( #ColorBrewer Blues 9:9-3
                        8       ,69,    148,
                        33      ,113,   181,
                        66      ,146,   198,
                        107     ,174,   214,
                        158     ,202,   225,
                        198     ,219,   239
                        )
        bluePal = rgb(matrix(BlueColors,byrow=TRUE,ncol=3),maxColorValue=255)
        #File    continent boundary
        #By      Daniel B. Carr
        #Date    October 9, 2007
        #dir = "C:\\Documents and Settings\\Daniel Carr\\My Documents\\JPL\\AIRS Data Four
Winters\\"
        fil = "continent.txt"
        #ans = read.table(paste(dir,fil,sep=''),colClasses=c('numeric','numeric'))
        ans = read.table(paste(fil,sep=''),colClasses=c('numeric','numeric'))
```

```
	rx = range(ans[,1],na.rm=T)
	ry = range(ans[,2],na.rm=T)

	yBot = 0.1; yTop = baseMeta - 0.2; yHt = yTop - yBot
	xLeft = max(0+mapBorder, (xMax/2)-2*yHt)
	xRight = min(xMax-mapBorder, (xMax/2)+yHt)
	dX = (1/72)*(xRight-xLeft)
	dY = (1/36)*(yTop-yBot)

	xBord = c(xLeft, xLeft, xRight+dX/2 , xRight+dX/2, xLeft)
	yBord = c(yBot, yTop, yTop, yBot, yBot)

	leftSide = xLeft
	rightSide = xRight
	botMap = yBot
	topMap = yTop
	mapX = leftSide		+ (rightSide-leftSide)* (ans[,1]+180)/360
	mapY = botMap		+ (topMap-botMap)    * (ans[,2]+90)/180

	#polygon(xBord, yBord,col=bluePal[6])
	polygon(x=mapX,y=mapY,col=gray(0.8),border=gray(0.6),density=NA)

	#for (i in 1:36) {
	#	for (j in 1:72) {
	#		if (map[i,j] > mapThresh) {
	#			xC = xLeft + j*dX
	#			yC = yBot  + i*dY
	#			rect(xC-dX/2, yC-dY/2,xC+dX/2,yC+dY/2,col="light
grey",border=NA )
	#		}
	#	}
	#}
	polygon(xBord, yBord)
	# geo-footprint for cluster
	for (j in 1:(numA+numB) ) {
		xC = xLeft + lons[j]*dX
		yC = yBot  + lats[j]*dY
		rect(xC-dX/2, yC-dY/2,xC+dX/2,yC+dY/2,col=cols[j],border=NA )
	}
	legendL = c("<=     5","<=    10","<=    25","<=    50", #greys
				"<=  100", "<=  250", "<=  500", #reds
```

```
                                        "<=1000", "<=2500", " > 2500")
        legend(x=(colC+0.5*boxWide), y= (baseMeta+6.5),title="Footprints",
legend=legendL,fill=VecPalette,bty="o")
}
```

## References


http://airs.jpl.nasa.gov/, NASA JPL AIRS Homepage, 25 March 2009

http://disc.gsfc.nasa.gov/AIRS/overview, Goddard Space Flight Center AIRS Homepage, 25 March 2009

http://disc.sci.gsfc.nasa.gov/AIRS/data-holdings/by-data-product/ AIRS Data Sets, 25 March 2009

Barbour, A.D., Xia, Aihua; "On Stein's Factors for Poisson Approximation in Wasserstein Distance", Bernoulli, Vol. 12, No. 6, Dec 2006

Belili, Nacreddine, Heinich, Henri; "Approximation pour la distance Wasserstein", C.R. Acad. Sci. Paris, Ser. I 335 (2002)

Braverman, Amy J.; Fetzer , Eric J.; Kahn, Brian H.; Manning, Evan M; Oliphant, Robert B; Teixeira, Joao P.; "Massive Dataset Analysis for NASA's Atmospheric Infrared Sounder", Technometrics, 54:1, 1-15, 2012 [b]

Braverman, Amy J.; "Compressing Massive Data Sets Using Quantization", Journal of Computational and Graphical Statistics, 12(4), pp. 44-62, 2002.

Braverman, Amy J.; Kahn, B.; "Visual Data Mining for Quantized Spatial Data", Proceedings in Computational Statistics 2004, Jaromir Antoch, editor, Physica-Veriag/Springer. 2004

Braverman, Amy J., Fetzer, E. ; "A Probabilistic Approach to Mining Massive Earth Science Data Sets"; GMU Presentation 2006

Brewer, Cynthia A,. 2002-9 http://colorbrewer2.org, accessed 2008

Carr, Daniel B.; Ashley, John M.; "Visualizing Grid Cell Multivariate Summaries of Global Atmospheric Data as Clusters over Space and Time", Poster presentation at JSM 2011

Carr, Daniel B.; Braverman, Amy J. ; “Visualizing Cluster-Compressed Multivariable and Multialtitude Atmospheric Data”; GMU Presentation 2007 [a]

Carr, Daniel B.; Braverman, Amy J. ; “Visualizing Cluster-Compressed Multivariable and Multialtitude Atmospheric Data”; Taipei Conference Presentation 2007 [b]

Chou, P.A., Lookabaugh, T. , Gray, R.M.; “Entropy-constrained vector quantization;” IEEE Trans. on Acoustics, Speech, and Signal Processing, Jan 1989,Vol 37, Issue 1, 1989

Cleveland, William S., “Visualizing Data”, Hobart Press 1993

Cressie, Noel A.C.; “Statistics for Spatial Data”, Wiley 1993

Cressie N. and Wikle, C. K.; “Statistics for Spatio-Temporal Data”, John Wiley & Sons, Hoboken NJ. 2011

Horowitz, Joseph; Karandikar; Rajeeva; “Mean rates of convergence of empirical measures in the Wasserstein metric”, Journal of Computational and Applied Mathematics, Vol. 55, Iss. 3, Nov 1994

Huang, Hsin-Cheng ; Cressie, Noel; Gabrosek, John; “Fast, Resolution-Consistent Spatial Prediction of Global Processes From Satellite Data”
Journal of Computational and Graphical Statistics, , Vol. 11, No. 1, 63-88: 63-88. Mar 2002

Johnson, M. H.; Ladner, R.; Riskin, E.A.; "Fast Nearest Neighbor Search for ECVQ and Other Modified Distortion Measures", in Proceedings of the IEEE International Conference on Image Processing, Lausanne, Switzerland, pp. 423—426, Sept. 1996

Kreitmeier , Wolfgang; “Optimal vector quantization in terms of Wasserstein distance”, Journal of Multivariate Analysis 102 (2011)

Lance, G.; Williams,W;; “A General Theory of Classification Sorting Strategies”; Computer Journal 9, pg 373-380, 1967

Moustafa, Rida E. A.; “QGPCP: Quantized Generalized Parallel Coordinate Plots for Large Multivariate Data Visualization”, Journal of Computational and Graphical Statistics, Vol. 18, No. 1, 32-51: 32-51, Mar 2009

Olson, Braveman, Granger, Manning;
http://disc.sci.gsfc.nasa.gov/AIRS/documentation/documentation/v5_docs/AIRS_V5_Release_User_Docs/V5_L3_Quantization_QuickStart.pdf AIRS L3 Quantization product Documentation, 25 March 2009

Rubner, Yossi; Tomasi, Carlo; Guibas, Leonidas J.; "The Earth Mover's Distance as a Metric for Image Retrieval"; International Journal of Computer Vision 40(2), 99–121, Kluwer Academic Publishers 2000

Schimek, Michael G.; editor; "Smoothing and Regression Approaches, Computation, and Application"; Wiley 2000

Shirdhonkar ,Sameer; Jacobs , David W.; "Approximate Earth Mover's Distance in Linear Time", CVPR 2008

Tufte, Edward R.; "The Visual Display of Quantitative Information"; Graphics Press 1983

Ware, C. "Information Visualization", Third Edition, Morgan Kaufman/Elsevier 2013

Xu, Rui; Wunsch II, Donald C; Clustering; IEEE Press 2009

Zhou, D. and Shi, T. ; "Statistical Inference based on Distances between Empirical Distributions"; Preprint 848, Department of Statsitics, The Ohio State University 2011

Zhou, D. and Shi, T.; "Statistical Inference based on Distances between Empirical Distributions with Applications to AIRS Level 3 Data" Proceedings of the NASA Conference on Intelligent Data Understanding (CIDU) 2011

## Curriculum Vitae

John M. Ashley received a Bachelor of Science in Electrical Engineering degree from Michigan State University in 1989 and a Master of Science in Electrical Engineering degree from Michigan State University in 1991.

He has worked for a variety of companies in a variety of roles, including the National Superconducting Cyclotron Laboratory, PJM Interconnection, KPMG Consulting, BearingPoint, American Management Systems, Maximus, Entigence, Silicon Graphics, and is currently employed by NVIDIA in the United Kingdom.

He is a certified Project Management Professional and co-inventor of United States Patent 7,930,254 “Property value estimation using feature distance from comparable sales”.