DimStiller: Workflows for dimensional analysis and reduction

Stephen Ingram, Tamara Munzner*
University of British Columbia

Veronika Irvine, Melanie Tory†
University of Victoria

Steven Bergner, Torsten Möller‡
Simon Fraser University

Abstract

DimStiller is a system for dimensionality reduction and analysis. It frames the task of understanding and transforming input dimensions as a series of analysis steps where users transform data tables by chaining together different techniques, called operators, into pipelines of expressions. The individual operators have controls and views that are linked together based on the structure of the expression. Users interact with the operator controls to tune parameter choices, with immediate visual feedback guiding the exploration of local neighborhoods of the space of possible data tables. DimStiller also provides global guidance for navigating data-table space through expression templates called workflows, which permit re-use of common patterns of analysis.

1 Introduction

Tables are a common way to organize data that can be interpreted in terms of cardinality and dimensionality as a set of n points in m dimensional space. Many practical questions about a high dimensional dataset require understanding how the dimensions and points relate to each other and to an underlying space: Are my dimensions meaningful? How do my dimensions relate to each other? Are my clusters real?

A combination of known statistical and visualization techniques can help analysts answer the three questions above. For example, the question "how do my dimensions relate to each other?" may be answered using Principal Components Analysis and interpreting the magnitudes of the eigenvalues and eigenvectors of the correlation matrix. Each technique may produce different output with a corresponding specialized interpretation. The sheer proliferation of techniques makes navigating this analysis space a daunting prospect for many users, who do not fully understand how and when to use these techniques correctly. For instance, what parameter settings make sense for a particular dataset? When can the output of one technique be legitimately used as the input for another?

A large class of dimensionality reduction techniques have the holistic goal of determining a new set of synthetic dimensions that express most or all of the structure in the input dataset using fewer total dimensions than the original representation. For example, Principal Components Analysis (PCA) constructs synthetic dimensions that are a linear combination of some subset of the original ones [9], and other methods such as multidimensional scaling (MDS) use nonlinear combinations [1]. The goal of these techniques is to reveal hidden variables, for example where the phenomenon of interest could only be measured indirectly. In these cases, the data reside in a lower-dimensional manifold whose coordinate axes are best represented as a linear or nonlinear combination of the input dimensions. Many dimensionality estimation techniques have been proposed for finding this *intrinsic* dimensionality of the dataset, which is typically given as input to a dimensionality reduction algorithm.

In contrast to the profusion of previous work proposing better or faster techniques for specific aspects of dimensional analysis [5, 7], far less attention has been paid to creating systems that guide users through the larger process of analyzing high-dimensional data iteratively using a combination of techniques. For instance, many analysts who lack a deep knowledge of dimensional reduction simply project their data to some 2D space and plot these points using a scatterplot. However, if the intrinsic dimensionality of the dataset is larger than two, clusters or orthogonal axes may be projected on top of each other, occluding relevant structures of interest. Although experts confronted with the problematic result of a single undifferentiated blob could conjecture that there is a mismatch between intrinsic dimensionality of the dataset and the space they chose to project to, less sophisticated users are routinely misled. The reasons for their perplexity include the sheer number of proposed dimensionality reduction techniques [6], the complexity of the mathematics underlying them, the widespread availability of dimensionality reduction tools that create only a single 2D or 3D scatterplot [10], and the lack of clear characterizations in the literature of which techniques are suitable for what kinds of data and tasks. DimStiller was expressly designed to help users avoid this pitfall, with a dimension reduction workflow that includes an estimation of intrinsic dimensionality of the dataset to guide users in making an informed choice about how many dimensions to reduce to, if at all, and having the default view of a table of more than two dimensions be a scatterplot matrix rather than a single scatterplot.

The first contribution of this paper is the design and implementation of the DimStiller visualization framework. DimStiller gathers together a variety of techniques from dimensional analysis and reduction into a coherent framework that emphasizes the underlying dimensions and the relationships between them. For instance, it guides users through estimating the intrinsic dimensionality of a dataset before carrying out reduction. The analysis technique components of DimStiller are outfitted with interactive controls and linked views, allowing users to see and manipulate intermediate results at each analysis step. Section 4 describes the task of dimensionality reduction and analysis. We present the DimStiller architecture in Section 5, and two case studies showing how it can be used to analyze complex real-world datasets in Section 6.

The second contribution is providing both local and global guidance for analysts engaged in navigating through the space of possible data tables during dimensional analysis and reduction, using the abstractions of operations, expressions, and workflows. *Expressions* instantiate a chaining together, or composition, of transformation *operators* on input data tables. Expressions show which operations have been applied to the data as well as the order in which they occur. *Workflows* are templates for exploration that consist of a specific expression along with the saved parameter values for each operator. Workflows bundle together the sequence of operators of an expression independent of the data on which they operate, permitting DimStiller users to re-use and share common patterns of analysis. While mechanisms such as expressions, operators, pipelines, and macros have been proposed previously in many contexts [4], the novelty in DimStiller is the way in which these are used to walk an analyst through a series of operations on data tables, providing guidance during the analysis process. In Section 2 we discuss the idea of guidance in further detail, and contrast our approach to previous work in Section 3.

*e-mail: {sfingram,tmm}@cs.ubc.ca
†e-mail: {vmi,mtory}@uvic.ca
‡e-mail: {sbergner,torsten}@cs.sfu.ca

2 Local and Global Guidance

The analyst engaged in the dimensional analysis and reduction process must choose from a vast number of possible transformations of the data table at every step – but only a relatively small subset of these transformations will yield meaningful information about the structure of the input dataset. We define providing *guidance* as structuring the exploration process to help users find this small, meaningful set from the huge space of possible transformations. We first describe this abstract analysis space, and then explain the two kinds of guidance, local and global, that DimStiller provides to support effective navigation in this space.

We model the dimensional analysis and reduction process as traversing *table space*, the space of all possible data tables. Conceptually, this space is like a graph where nodes are data tables, connected by edges representing a transformation. A path through the space begins at some node representing the input data table. Intermediate nodes are the tables that result from the transformations applied by the user, and the path terminates at the output data table. We thus must consider how to help data analysts locate, explore, and traverse relevant regions in table space.

At the global level, we guide the analyst who must find a path that traverses the space from some given start point to a useful end point. Workflows are a mechanism to represent entire paths in this space. They represent a chained pipeline of transformations that can be applied to an input dataset. The built-in workflows are a small set of paths intended to span the space between a few landmarks of potential interest. DimStiller is designed to help analysts find new, useful paths through table space, and save them as new workflows for later use.

At the local level, analysts also need to explore neighborhoods in table space: tuning the parameters of an individual transformation operator corresponds to searching for the most informative data table in the region of table space that is reachable with a single transformation that can be carried out by the chosen operator. DimStiller supports this local exploration through a chain of linked operator controls and views, so that users have immediate visual feedback about the effects of parameter tuning.

3 Related Work

Dimensionality reduction and analysis draws from an immense and varied body of work [6]. Here we focus on the most related work to our own, software systems that provide users with access to dimensionality reduction algorithms and visualizations of the results.

3.1 Programming Environments

We first consider full-fledged programming environments such as MATLAB [16] or R [12]. One strength of these systems is execution speed, but at the cost of requiring the user to program. In contrast, DimStiller is used through a graphical user interface, where programming is only required to augment the system's set of operator plugins. Another strength of these environments is their flexibility; in contrast, DimStiller only supports a more specialized set of operators, but provides both local and global guidance for exploring table space productively, even for non-expert users.

3.2 Toolkit Solutions

In specialized toolkits, several algorithms have been packaged together, usually with a GUI front end. For example, the Matlab Toolbox for Dimensionality Reduction[1] gathers together over 30 techniques for dimensionality reduction under one umbrella. Another example is the HIVE dataflow toolkit for dimensionality reduction [13]. While such tools reduce programmer time by providing easy access to a wide variety of analysis techniques, neither local nor global guidance is provided to the user.

3.3 Visual Dimensionality Analysis Environments

The previous systems most related to DimStiller are full-fledged environments for visual dimensionality analysis and reduction.

XmdvTool [17] supports interactive visual exploration of multivariate data sets through many types of views including scatterplot matrices, with interactive controls that include sophisticated linking and brushing techniques. It includes several approaches to collecting and culling dimensions that are based on hierarchical clustering of the dimensions using a variety of metrics, such as DOSFA [19] and VHDR [20]. Its Value and Relation (VaR) technique [18] does use MDS to create a scatterplot of dimensions, encoding information about the dimensions in an information-dense pixel-oriented glyph at each scatterplot point. However, XmdvTool is not primarily designed to support the workflows built around reduction through synthesizing new dimensions that are the core focus of DimStiller. In contrast, the GGobi system [2] is a visualization framework for high-dimensional analysis with dimensionality reduction techniques that create synthetic dimensions as a central focus, supporting interaction between multiple kinds of linked views including scatterplot matrices. It also features sophisticated high-dimensional navigation including projection pursuit and grand tours, and a plugin architecture for easy connection with R [12]. The limitation of both of these frameworks is that while they implicitly provide ways to access and explore many relevant paths through table space in useful and novel ways, they lack an explicit framework for local and global guidance.

The rank-by-feature framework of Seo and Shneiderman [14] allows the user to visually inspect and explore dimensional relationships, but only with subsets of the original dimensions, so the huge part of table space that can only be reached via constructing synthetic dimensions cannot be explored. The data exploration environment of Guo [3] has a component-based architecture for finding clusters of the data with unique dimensional relationships. DimStiller is more focused on the overall dimensional relationships of a selected table in the data rather than analyzing differing relationships across different subspaces. Moreover, it uses a simpler, pipeline component architecture with a standardized protocol for component parameter modification, thus allowing better support for both local and global guidance.

Johansson and Johansson's [8] system is the closest in spirit to ours, since it does indeed embody the concept of guiding the user through analysis stages. Users can craft quality metrics from combinations of correlation, clustering, or other measures and cull dimensions according to these measures. However, their system only supports one hardwired global workflow, where the only flexibility is in setting parameter values at each local stage. Another limitation is the inability to construct synthetic dimensions. DimStiller includes this capability at its core, and also permits the user to factor in dimension culling at any point in a pipeline of dimension transformations, greatly increasing the scope of analysis tasks that it supports.

4 Users and Tasks

We now describe in detail the intended target user population for DimStiller, the questions that DimStiller is designed to help these users answer, and common techniques currently used to answer such questions. While these are not the only questions an analyst might be interested in, we argue that they are a good place to start when considering a new dataset, especially one with unclear provenance that is not necessarily well curated.

[1] homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction.html last visited on 2/01/2010

4.1 Target User Population

DimStiller is aimed at bridging the gap between state of the art techniques in visually oriented dimensionality analysis, and the current practices of many users and potential users who do not already have deep knowledge of their data and the mathematics of reduction. Although dimensional analysis is sufficiently complex that we do not target casual users, we argue that this middle ground between utter novices and fully confident experts is a sizeable group that is underserved by the current set of available systems.

For example, a visualization researcher called on to help somebody analyze a dataset may be completely unfamiliar with the dataset characteristics and the tasks of the researchers at the beginning of the analysis process. Furthermore, the person might be a visualization generalist rather than a specialist in the mathematical foundations of high dimensional techniques in particular. Another example is end users who have expertise in their own domain and the desire to do some dimensional analysis, but not deep knowledge of reduction mathematics. They might be developing algorithms to generate or process the data, and seek to evaluate the quality of their results or fine-tune parameter settings.

DimStiller is particularly aimed at providing major process improvements for analysts who must deal with messy datasets that may have unclear provenance. By providing both local and global guidance through table space, we aim to support analyses that might otherwise seem too daunting and decrease the chances that non-expert users draw incorrect conclusions, supporting a qualitatively different analysis process than with previous tools. For those analysts dealing with curated datasets where the meaning of each row and column are already fully understood, DimStiller may simply speed up a previously feasible, but slow, analysis process.

4.2 Are my dimensions meaningful?

Sometimes an input dimension may actually contain little or no useful information at all. Because of this, it is important for an analyst to be able to characterize the dataset in terms of which dimensions have useful information versus the "meaningless" ones. This understanding is not critical for downstream analysis algorithms in the same analysis session, since the mathematics of dimension reduction will handle creating the correct lower-dimensional projection. However, discovering that a given dimension is culled could reveal problems with the data source, with major upstream consequences in later iterations of the larger analysis loop: the data might be gathered differently, or the algorithms to generate it might be refined.

One such criterion is simply to check the underlying variance of the input dimensions and cull those beneath a small noise threshold. Another possibility is to use an information entropy cutoff.

4.3 How do my dimensions relate to each other?

Much of multivariate statistical analysis is concerned with how the individual dimensions relate to each other. Many popular metrics such as Pearson's correlation coefficient measure pairwise relationships between individual dimensions. More holistic methods such as Principal Components Analysis determine how all the dimensions may actually express a smaller subset. Other methods uncover more complex nonlinear relationships between the input data, such as multidimensional scaling or many of the manifold-following variants.

4.4 Are my clusters real?

While the previous questions are related to dimensions, the question of cluster membership relates to the points. Clustering is the assignment of a unique label to specific regions of the input data's feature space and the points that occupy them. Cluster labels can be computed from any of a myriad of clustering algorithms.

Clustering relates to the input dimensionality in a reciprocal way. If the analyst trusts the dimensional basis in which the data is represented, then point clusters in such a space will be considered real clusters with higher confidence than without such a trust. Likewise, if the analyst is given a clustering that is trusted to be real, and the space in which the data is projected maintains the clustering's coherence, then this result increases confidence in the current dimensions. Thus, a clustering can inform the quality of a dimensionality reduction, and vice versa.

5 DIMSTILLER ARCHITECTURE

It is clear that an analysis tool that provides users with the ability to load their data into the system, transform their data with different analysis techniques, and scrutinize their data before and after these transformations would help users answer these questions. What is not clear is how such a tool should organize the results of applying the transformations or how to link these transformations together with visualizations to keep users focused on the analysis.

DimStiller organizes dimensionality analysis and reduction as a pipeline of transformations to a data table and linked views of it at different pipeline stages, as shown in Figure 1. The DimStiller model is based on an abstraction called an *expression* which encapsulates a sequence of transformations, called *operators*, that act upon tables of data where rows are points and columns are dimensions. Operators transform tables by adding, removing, or changing points or dimensions. Operators may have control panels and associated views that provide a visual representation of a table at that pipeline stage. All views are linked, and selections are propagated up and down the pipeline appropriately. A key aspect of the DimStiller architecture is the ability to instantiate expressions from pre-existing *workflows* that capture useful analysis patterns.

The DimStiller expression and operator abstractions were partially inspired by the Expression and Operator patterns in [4]. However, DimStiller expressions have a simple, linear topology that defers processing to the operators in contrast to the general tree structure suggested by the Expression pattern. Likewise, DimStiller operators are composable processing units similar to the Operator pattern, but compute general transformations and not necessarily visual mappings.

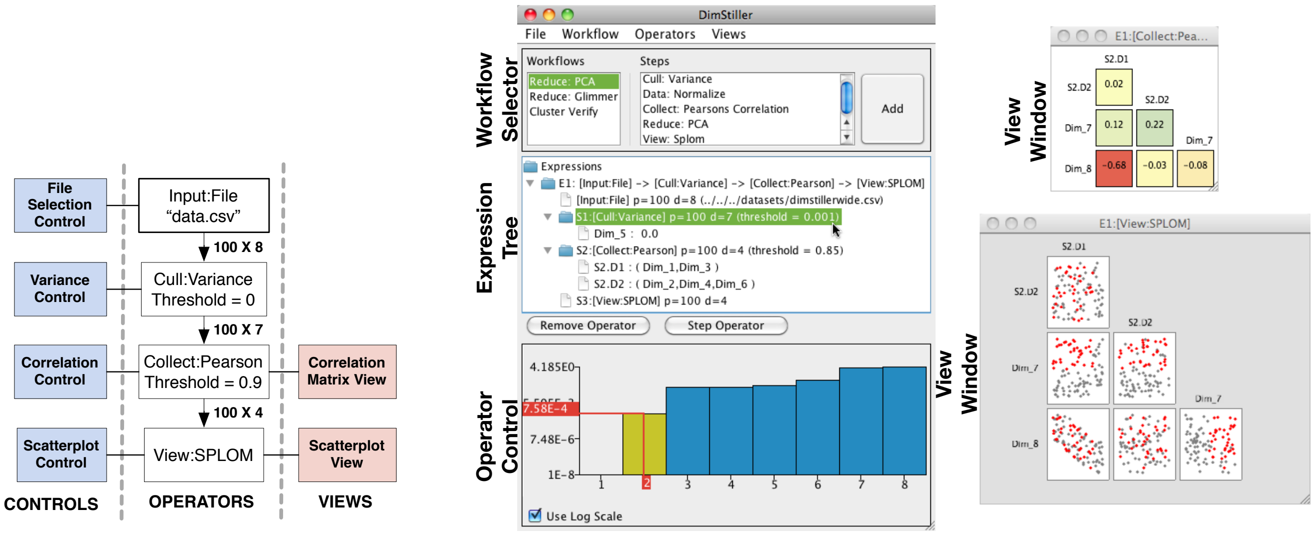
5.1 Input Tables and Dimension Types

DimStiller supports an abstract interface to data via a *table* model. Conceptually, a table can represent anything from physical entries in a disk file, to a cross-network database, or even evaluations of a simulator. The current implementation only supports simple file-based tables. There are two kinds of dimensions: Data, and Attribute. Data dimensions are either Quantitative or Categorical. Internally, both are represented by floating point values, and DimStiller maintains a lookup table to map floating point values to category symbols for Categorical dimensions. Attribute dimensions represent values such as color or selection that are used in views such as scatterplot matrices, but are ignored by purely data-oriented operators such as variance culling or dimension reduction.

5.2 Operators

Operators are functions that map an $n \times m$ table to an $n' \times m'$ table. That is, operators may add or delete points or dimensions, or change the value of any existing cell in the table. In the parlance of Section 2, they are the edges that connect two nodes in table space. For example, the `Cull:Variance` operator removes dimensions with low variance from an $n \times m$ table. If any of the m table dimensions has variance below the user-controllable threshold, then the application of this operator would result in a new $n \times m'$ table where $m' < m$.

Every operator may have an associated control and/or an associated view, although neither is mandatory. Operator *controls* are



Workflows: Reduce: PCA, Reduce: Glimmer, Cluster Verify

Steps: Cull: Variance, Data: Normalize, Collect: Pearsons Correlation, Reduce: PCA, View: Splom

```
Expressions
  E1: [Input:File] -> [Cull:Variance] -> [Collect:Pearson] -> [View:SPLOM]
    [Input:File] p=100 d=8 (../../../datasets/dimstillerwide.csv)
    S1:[Cull:Variance] p=100 d=7 (threshold = 0.001)
      Dim_5 :  0.0
    S2:[Collect:Pearson] p=100 d=4 (threshold = 0.85)
      S2.D1 : ( Dim_1,Dim_3 )
      S2.D2 : ( Dim_2,Dim_4,Dim_6 )
    S3:[View:SPLOM] p=100 d=4
```

Figure 1: **Left:** The anatomy of a simple DimStiller expression. Input data is fed into a pipeline of operators that alter the dimensionality of the data. Each operator may or may not have controls or views. An Operator's control is displayed when it is selected in the Expression Tree. An operator's view is shown in a separate window, allowing side by side comparison between multiple views. Changing an operator's parameter in the control may produce a change to its output that is propagated across the expression using events that may travel both upstream and downstream from the operator. **Right:** The DimStiller interface for this expression, showing the `Collect:Pearson` and `View:SPLOM` views. The `Cull:Variance` expression is selected in the Expression Tree, so its control is visible.

GUI elements that permit users to modify operator parameters. Operator controls afford the user command over local search in table space. For example, Figure 2 shows the control panel for the above `Cull:Variance` operator, which has an interactive plot of the variance for each input dimension. The threshold parameter for culling is adjusted by clicking directly on the plot, and this operator does not have a separate view. Operator *views* are visualizations of the data table at that stage of the pipeline. Some operators are purely view-oriented, and do not transform the data table at all. For example, the view for the `View:SPLOM` is a scatterplot matrix, and its control panel only affects this visual display. In contrast, the `Collect:Pearson` operator has both a control panel with a slider to change the threshold, and a separate view with a matrix of colored boxes to show the pairwise correlations encoded with a blue-yellow-red diverging colormap.

The Operator namespace has a two-level structure, where a **family** has specific **instances**. The set of Operator families and instances, with notation `Family:Instance`, is:

- `Attrib:Color` Adds attribute color dim for views
- `Collect:Pearson` Join highly correlated dims
- `Cull:Variance` Identify/remove low-variance dims
- `Cull:Name` Identify/remove dim by name string match
- `Data:Norm` Normalize input dims
- `Input:File` Load comma separated value (CSV) file
- `Reduce:PCA` Estimate/reduce dimensionality with PCA
- `Reduce:MDS` Estimate/reduce dimensionality with MDS
- `View:SPLOM` Plot *n*-dim table using *nxn* scatterplot matrix
- `View:Histo` Plot dim distribution with histogram

The current families and operators serve to illustrate the potential of our approach to system architecture; the set of families is not exhaustive, nor is the set of operators within any family. The built-in set of operator families and instances can be extended by implementing new operators.

5.2.1 Cull and Collect Operator Families

Operators in the Cull family compute a specific criterion for each dimension and remove those dimensions that do not satisfy it. The criterion for the `Cull:Variance` operator is variance, and dimensions that fall beneath a user-specified threshold are culled. It can help users locate and eliminate dimensions whose variability is zero, or is small but non-zero because of noise. The `Cull:Name` operator allows users to selectively remove dimensions manually, for example to analyze only a subset of the input dimensions.

While the Cull family acts on individual dimensions, operators in the Collection family use pairwise criteria like covariance and Pearson's correlation coefficient. Rather than removing dimensions whose pairwise measures do not satisfy the threshold, these operators replace them with a single representative dimension for the collection, for example the average.

These operators can help users who may need to refine the processes used to generate their input dataset, as we discuss in Section 4.2. They are also useful for those whose analysis needs preclude the creation of synthetic dimensions.

5.2.2 Reduce Operator Family

A critical design choice in the Dimstiller architecture is that the `Reduce` operator family includes estimation of the intrinsic dimensionality of the space in addition to actually performing the reduction. The control for the operator, shown in the lower left of Figure 4, has a scree plot: a bar chart with the number of dimensions on the horizontal axis, and an estimate of the variability that would not be accounted for if the dataset were reduced to a space of that size on the vertical axis. The user then can make an informed choice when selecting the target dimensionality, by clicking on the plot at the desired threshold. (The definition of "intrinsic dimensionality" that we espouse is the smallest dimensionality of the set of spaces in which the data can be embedded with distortion *less than some noise tolerance*, rather than *zero* distortion.) DimStiller supports users in experimentally determining the correct noise threshold, which differs between datasets, by interactive threshold adjustment.

Although scree plots are far from new, most previous toolkits do not explicitly couple them to the use of a reduction algorithm: users are simply expected to provide a number as input, with no guidance. Users who are not experts or are dealing with unfamiliar datasets will often have no idea of what a reasonable number might be. Worse yet, a significant number of reduction technique implementations are hardwired to blindly reduce to two (or three) dimensions, with no hint to the user that this choice might be inappropriate or misleading. Even the relatively sophisticated analyst who knows to run an estimator is often provided with the black-box output of a single number, rather than the detailed information for each possible number of cumulative dimensions shown in a scree plot [5]. Our design also has the benefit that users can see different estimates of intrinsic dimensionality in a lightweight and fast way with the scree plots, rather than the more heavyweight approach of reducing and then viewing the results in a scatterplot matrix. A related design choice is that the `View:SPLOM` view for showing a table is a scatterplot matrix rather than a single scatterplot. When the table has only two dimensions this view does of course show the case of only a single scatterplot, but when it has more the user is guided to see all of the information rather than an arbitrary subset.

While designing the architecture, we considered whether to have the estimation step separate from the reduction step. We ultimately decided that they should be coupled together into one module in service of the goal of providing guidance for the non-expert user. Understanding which estimators are appropriate for which reduction algorithms requires significant knowledge of dimensionality reduction: for example, nonlinear reduction methods should not be used in conjunction with linear estimators. Thus, we do not expect the middle-ground user to make that choice, reserving it for the designer of new operators.

The exact measure shown on the vertical axis of the scree plot depends on the operator instance. The `Reduce:PCA` operator shows the eigenvalues, and the `Reduce:MDS` shows the *stress* values of the embedding in each dimension. We use the CPU implementation of Glimmer [7] for both the MDS reduction and estimation.

5.2.3 Attribute and View Operator Families

Attribute operators add attribute dimensions to the output data table of the operator. Attribute dimensions are interpreted by view and attribute operators and ignored by other operator families. The Color attribute operator creates an attribute dimension used for coloring to which it assigns values based on the values of numeric or categorical dimensions. The assignment of colors is performed either by linearly interpolating between two endpoint colors or by assigning colors to individual dimension values. The default colormap for categorical data, inspired by the work of Stone [15], has 10 bins; colors are repeated if there are more than 10 values.

View operators provide visualizations of their input data. The two built-in View operators are `View:Hist` for showing the distributions of individual dimensions, and `View:SPLOM` for showing pairwise relationships between dimensions. Both the SPLOM and the Histogram views provide global linked selection by creating an attribute dimension for selection, and displaying points with a nonzero selection attribute value in a default selection color. SPLOM views also use the color attribute dimension to color points.

5.3 Expressions

A DimStiller expression is the instantiation of an ordered list of operators applied to an input table. Figure 1 Left illustrates the elements of a sample expression, showing the associated views and controls for each operator. As the expression progresses, the data entries change value and the output table changes shape as it is progressively refined. Figure 1 Left also shows the relevant pathways for how information about input, view, and parameter changes moves across the expression. In our informal description of the table space graph of Section 2, the nodes are data tables and the edges are the transformation operators. However, in the DimStiller user interface, the more natural representation uses the dual graph, where operators are the nodes, and the edges represent the data flowing between them. The user is thus encouraged to focus on manipulating and understanding the transformations of the data.

5.4 Workflows

Workflows are templates for entire expressions that can be immediately created with a few clicks. DimStiller has a base set of workflows built in, and users can create their own by saving the list of operators in any active expression as a workflow.

A workflow contains a sequence of individual operator *steps*, and saved parameters associated with each operator. When a user instantiates a workflow, a new expression is produced unique to a given input table. Because many operators may result in time-consuming computations, only the first operator in a workflow computes its output upon workflow instantiation, with subsequent operators greyed out in the user interface. Users choose when to progress to activating the next step, possibly after adjusting parameters at the current step, with a `Step Operator` button. Heavyweight operators downstream will thus only initiate their computations on data tables that may be much more compact than the input table due to reduction at upstream stages.

The built-in workflows are designed to help users begin to answer the set of questions that we identified in Section 4.1, as a proof of concept that this style of guidance can help middle-ground users. We do not claim that they are the only way, or even the best way in all cases, to answer these questions. Workflows provide optional global guidance; they are not mandatory. Power users have the flexibility to build up new expressions directly in DimStiller by choosing individual operators from the currently loaded set.

The set of workflows built into DimStiller are:

- **Reduce:PCA.** `Cull:Variance`→`Data:Normalize`→`Collect:Pearson`→`Reduce:PCA`→`View:SPLOM`
- **Reduce:MDS.** `Cull:Variance`→`Data:Normalize`→`Collect:Pearson`→`Reduce:MDS`→`View:SPLOM`
- **Cluster Verify.** `Attrib:Color`→`Data:Normalize`→`Reduce:PCA`→`View:SPLOM`

5.5 DimStiller Interface

Figure 1 Right shows a screenshot of the DimStiller session containing the expression diagrammed in Figure 1 Left. The visual structure of the DimStiller interface, with views and controls for each operator, encourages the user to examine the individual operators, adjust their parameters, and observe the effects on the resulting transformations in the visual representations.

The `DimStiller` window on the left contains the Workflow Selector at the top, with the Expression Tree underneath and an operator control panel on the bottom. Two view windows are visible on the right, a scatterplot matrix for the `View:SPLOM` operator and the correlation matrix for the `Collect:Pearson` operator. The Expression Tree shows that the input file `dimstillerwide.csv` contained a table with 100 rows of points and 8 columns of dimensions. In this example dataset, `Dim_1` and `Dim_2` are independently sampled from a uniform distribution between 0 and 1, `Dim_3` is a scalar multiple of `Dim_1`, `Dim_4` and `Dim_6` are scalar multiples of `Dim_2`, `Dim_5` is set to all zeros, `Dim_7` and `Dim_8` are linear combinations of both `Dim_1` and `Dim_2` with a uniform noise term.

The first `E1` operator `S1` is `Cull:Variance`. The user has clicked on the first nonzero dimension in the scree plot, resulting in a threshold value of 0.0007 (rounded to 0.001 for display in the tree). The summary line for `S1` in the Expression Tree shows that the output table of S1 has 7 dimensions as opposed to the 8 dimensions that were input to the operator, and the expanded details beneath show that Dim_5 is the one that was culled. The S2 operator collects dimensions that are correlated with the threshold of 0.85, resulting in a table of 4 dimensions whose pairwise correlations are shown with color in the top view. The expanded details shows Dim_1 and Dim_3 are now represented by new synthetic dimension S2.D1, and the remaining three are now represented by S2.D2. The last operator is View:SPLOM, and the bottom view shows the scatterplot matrix. The user selected some points in one plot, and they are colored red in all of the linked plots.

5.5.1 Workflow Selector

The Workflow Selector displays the available workflows and allows the user to select one and create a new expression from it. Selecting a workflow fills the adjacent list box with the sequence of steps for the user to inspect. If the user chooses to activate that workflow by clicking the Add button, DimStiller applies the workflow steps to the currently selected expression, making those operators visible in the Expression Tree.

5.5.2 Expression Tree

The Expression Tree is a three-level tree widget that lists all open expressions. At the top level, expressions are described by a short text summary where each new operator X is appended on the right of the text string as → [X], where X is a very terse label. When the user drills down to the next level, the individual operators that comprise the expression are listed with a concise yet complete text summary that includes the size of the output table produced by the operator in terms of points and dimensions, as well as any operator parameters that are set to non-default values. The third and final level of detail is only added if an operator modifies the output dimensionality, namely the list of the dimensions modified by that operator and any details that relate the input and output dimensions. For example, in Figure 1 the expansion of the Collect:Pearson operator shows two of the synthetic dimensions and names of the original dimensions collected together.

5.5.3 Operator Control Panel

Each operator may have a control panel that lets the user adjust its parameters. Called operator controls, they afford the user with the means to locally search for a meaningful region in table space. When an operator is selected in the Expression Tree, its control populates the operator control panel region at the bottom of the main DimStiller window. Only one operator can be selected at a time. Operator controls visible in this paper include the Cull:Variance control shown in Figure 1 and the Reduce:PCA control shown in Figures 4 and 5.

5.5.4 View Windows

Each operator also may have an associated view. When an expression is loaded or created, the associated views open up as individual windows to support side-by-side comparison across operators within the same expression or even across different expressions. All the views are created using the Processing language, but new operator view plugins could be created using any graphical toolkit that can interface with Java. The built-in operators that have views are the View:Histo, the View:SPLOM shown in Figures 1 and 2, and the box matrix showing pairwise relationships for the Collect:Variance operator in Figures 1 and 3.

6 Case Studies

We now describe how the DimStiller architecture facilitates the task of dimensionality reduction and analysis through case studies on real-world data. We use the built-in workflows to construct expressions that inform users about the character and relationships of the dimensions and clusters of the datasets[2].

6.1 Sustainability Simulation: Measuring Variability

Our first case study focuses on a sustainability simulation dataset containing a large collection of simulated results of government policy decision scenarios [11]. The 294 dataset dimensions represent the environmental and societal indicators affected by the policy decisions. A first attempt to analyze this data using preexisting tools fell prey to the *reduce to 2 and plot* pitfall discussed in Section 1 [11]. The simulation is agglomerated from many subpieces originally designed for varying purposes, rather than being carefully constructed from custom components that would dovetail seamlessly. The simulation designers thus did not have a clear idea of the intrinsic dimensionality of this dataset. They did know what their dimensions were, with meaningful labels for each such as Cost of Living and Air Quality. However, they thought it was possible that some indicators always had the same value across the entire dataset. They were also interested in learning about how the dimensions related to each other: they suspected that many indicators were highly correlated, but did not know the number of equivalence classes or which indicators were in each group. They were also curious whether automatically computed correlation groups would match with their intuitions about indicator relationships.

The first choice to make when using DimStiller is whether to construct our own expression from scratch by individually choosing operators, or to instantiate a workflow from the existing list. Since we are interested in finding the intrinsic dimensionality of the space as well as any correlations, a workflow in the Reduce family seems to be a good match, and we start with Reduce:PCA.

Figure 2 shows the control of the Cull:Variance operator that plots the sorted variances of the dimensions, with the log-scale option selected to emphasize small values. We notice that there are indeed many zero-variance dimensions, and click the scree control to remove these 34 dimensions, leaving 260 in the output table. The researchers could now drill down in the Expression Tree to see the names of the potentially problematic culled dimensions. They now know that either the input policy choices used in this run of the simulator did not effectively span the indicator space, or that there are unforeseen interactions between simulator components.

We click the Step operator button to activate the next workflow step, the Data:Norm operator. Both reduction workflows include a normalization step to guide users who may be unaware of the effects of transforming dimensions with differing scales of variation. The Expression Tree in Figure 4 shows that we chose to normalize using Z scores, so the operator subtracts the mean and divides by the standard deviation. We then step to activate the Collect:Pearson operator, which gathers highly correlated dimensions. Even the most stringent possible threshold setting of 1.0 for perfect correlation results in a drastic reduction of the number of dimensions: from 260 to 147. Figure 3 Left shows the correlation matrix view, where only a small fraction of the boxes are visible without scrolling. Relaxing the threshold to more reasonable value of 0.8 results in the view shown in Figure 3 Right, where the number of dimensions in the table is reduced to 22. Again, the simulation designers could now drill down in the Expression Tree to see the names of which dimensions were collected together, in order to check whether the automatic computations match their intuitions about the expected behaviour of the simulator.

Finally, we would like to determine whether the intrinsic dimensionality of this space is even smaller than the 22 dimensions of the table after culling and collecting, and if so reduce to that space. The

[2] The accompanying video shows the look and feel of interactive sessions with these datasets.

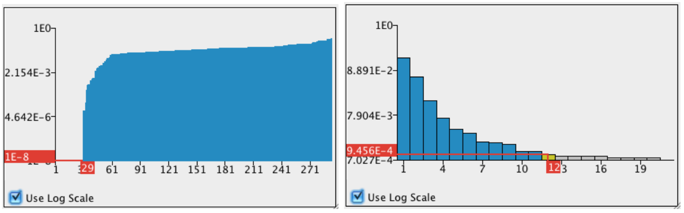


Figure 2: **Left:** The `Cull:Variance` operator control displays a sorted list of the dimension variances. Many have zero variance, indicating potential problems in the choice of input variables to the simulator or the operation of the simulator itself. Log-scaling of the variances emphasizes small values. **Right:** The scree plot for the nonlinear `Reduce:MDS` shows an intrinsic dimensionality of 12 dimensions, versus the 16 dimensions found by linear methods shown in Figure 4.

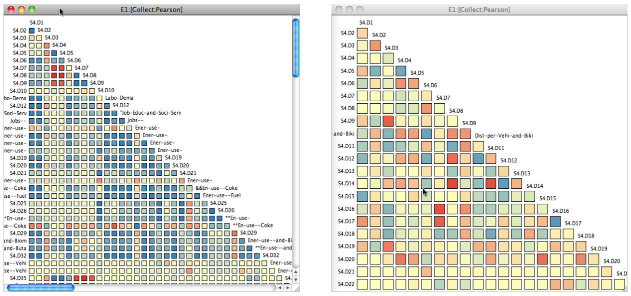


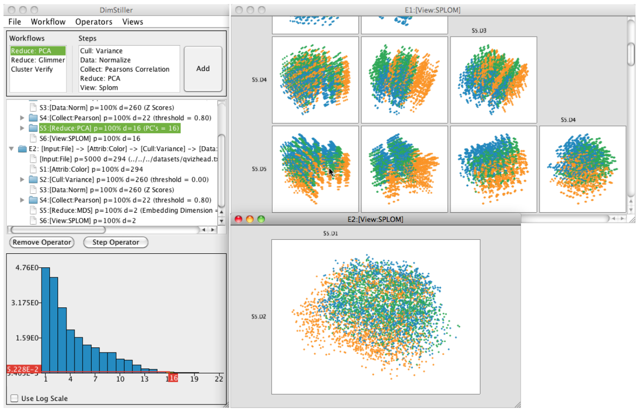
Figure 3: The `Collect:Pearson` operator view shows the correlation matrix with a diverging color scale ranging from blue at the positive end, through yellow for independent pairs, to red for negative correlation. **Left:** The perfect correlation threshold of 1.0 reduces the table from 260 to 147 dimensions. **Right:** Relaxing to a more reasonable threshold of 0.8 reduces the number of dimensions to just 22, all visible without scrolling.

`Reduce:PCA` operator constructs a linear projection of the data into a subspace that minimizes distortion. Our motivation for doing the reduction is to observe the structure of the major axes of the simulation output in a subsequent scree plot. The control view in Figure 4 shows the scree plot in the PCA control, and we see that the eigenvalues approach zero between 12 and 18 dimensions. Mousing over dimension 16 shows that it corresponds to a very small noise threshold of 0.001, and we click to select that value. We then click the `Step Operator` button and activate the `View:SPLOM` operator which brings up the scatterplot matrix view.

In order to facilitate viewing the original structure of the data in the input space, we insert an `Attrib:Color` operator into the expression after the `Input:File` and select the initial dimension as the dimension to color by. The resulting colored SPLOM is shown in Figure 4 in the top right view window, labeled `E1:[View:SPLOM]`.

The original analysis of the dataset was done by projecting the data down to two dimensions using multidimensional scaling. We quickly replicate this analysis in DimStiller so that we can compare the results directly. We reload the same data into a new expression `E2`, adding a `Attrib:Color` Operator and using the `Reduce MDS` workflow. The operators in this workflow are the same as in the previous analysis except for the Reduce operator, and we use the same settings as before. We check the scree plot for this case, and find that only 12 dimensions suffice with this nonlinear reduction, as shown in Figure 2. Then, to illustrate the *reduce to 2 and plot* pitfall, we use the `Reduce:MDS` Operator to select 2 as the output dimensionality. The SPLOM view at the bottom right of Figure 4 shows only the single scatterplot. The structure visible in the SPLOM above is completely hidden, and we see only an undifferentiated blob.

This analysis session with DimStiller shows that although the simulator produces hundreds of outputs, dozens have zero variance, most of the remainder are highly correlated, and the full structure of the data can be represented with only around a dozen dimensions. Although in theory this full analysis could have been carried out with existing tools like MATLAB, and bits and pieces of it were done over the course of a few years, in practice we did not have a complete picture of this messy real-world dataset until we could analyze it with the DimStiller system.



Workflows: Reduce: PCA, Reduce: Glimmer, Cluster Verify

Steps: Cull: Variance, Data: Normalize, Collect: Pearsons Correlation, Reduce: PCA, View: Splom

```
    S3:[Data:Norm] p=100% d=260 (Z Scores)
  ▶ S4:[Collect:Pearson] p=100% d=22 (threshold = 0.80)
  ▶ S5:[Reduce:PCA] p=100% d=16 (PC's = 16)
    S6:[View:SPLOM] p=100% d=16
▼ E2: [Input:File] -> [Attrib:Color] -> [Cull:Variance] -> [Data:
    [Input:File] p=5000 d=294 (../../../datasets/qvizhead.t
    S1:[Attrib:Color] p=100% d=294
  ▶ S2:[Cull:Variance] p=100% d=260 (threshold = 0.00)
    S3:[Data:Norm] p=100% d=260 (Z Scores)
  ▶ S4:[Collect:Pearson] p=100% d=22 (threshold = 0.80)
    S5:[Reduce:MDS] p=100% d=2 (Embedding Dimension =
    S6:[View:SPLOM] p=100% d=2
```

Figure 4: The scree plot in the `Reduce:PCA` control shows an estimate of the intrinsic dimensionality as between 12 and 18 dimensions, and we have interactively selected 16 as the threshold. The top scatterplot matrix shows a result with visible structure. In contrast, the bottom view shows the pitfall of reducing to just two dimensions using MDS, where an undifferentiated blob gives a misleading impression of no structure.

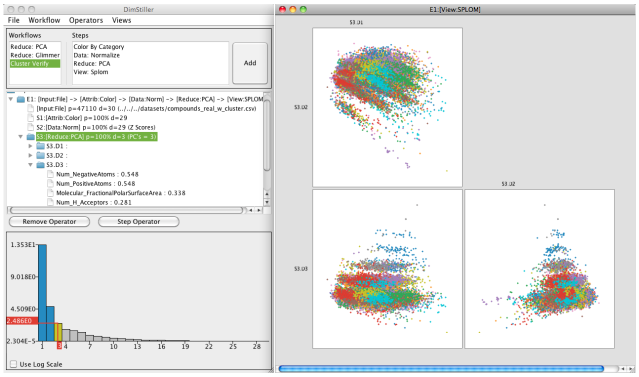
6.2 Computational Chemistry: Evaluating Clusters

We now examine a 30-dimensional computational chemistry dataset. The individual dimensions of this data measure physical properties of chemicals such as molecular weight and the number of bonds they possess. The dataset includes a cluster membership dimension with 236 clusters of the data produced by a commercial clustering package. The goal of the analysts who work with this dataset is to evaluate the quality of the clustering.

The `Cluster Verify` workflow is appropriate for this goal, so we instantiate a DimStiller expression from it. After loading the data, we use the `Color` operator control to choose which dimension we use for the categorical colormap. By default, the color operator culls the dimension by which it colors the points; including this cluster membership dimension in the downstream analysis would usually skew the results.

After adjusting color settings, we activate the `Data:Norm` operator which normalizes the dimensions to Z scores. We then activate the `Reduce:PCA` and observe its control. The scree plot of the eigenvalues, visible in Figure 5, shows an exponential drop off in magnitudes. This plot strongly suggests that the majority of the variance of the data resides in a lower dimensional space than the input dimensionality. Standard practice is to select the “knee” of the value drop-off curve as a good candidate for target dimensionality. We select 3 and then activate the `View:SPLOM` operator.

The resulting scatterplot matrix of the `View:SPLOM` Operator, also visible in Figure 5, reveals several interesting structures in the data. In the bottom row of two scatterplots, we observe clear separation of several clusters of points. In contrast to the spatial structure, some color labels appear to span gaps in the scatterplots. A single categorical color scheme of course cannot possibly show over 200 clusters with distinguishable colors, so DimStiller uses a repeating palette. To check whether some of the adjacent clusters with differing labels might have the same color by chance, we select the `Attrib:Color` Operator again in the Expression Tree to bring up its control. The `Permute Colors` button permutes the order in which colors are assigned to categories. After trying a few different permutations of the color scheme we conclude that the phenomenon we saw was not just an artifact; several cluster labels do indeed span these observed clusters. This result gives strong, albeit not conclusive, evidence that there may be better clusterings.



```
E1: [Input:File] -> [Attrib:Color] -> [Data:Norm] -> [Reduce:PCA] -> [View:SPLOM
    [Input:File] p=47110 d=30 (../../../datasets/compounds_real_w_cluster.csv)
    S1:[Attrib:Color] p=100% d=29
    S2:[Data:Norm] p=100% d=29 (Z Scores)
  S3:[Reduce:PCA] p=100% d=3 (PC's = 3)
    S3.D1 :
    S3.D2 :
    S3.D3 :
        Num_NegativeAtoms : 0.548
        Num_PositiveAtoms : 0.548
        Molecular_FractionalPolarSurfaceArea : 0.338
        Num_H_Acceptors : 0.281
```

Figure 5: Running the `Cluster Verify` workflow on a computational chemistry dataset. The scree plot in the lower left shows that most of the variability resides in a low dimensional subspace. We choose a threshold of 3 dimensions at the "knee" in the plot. We see in the colored SPLOM view that while the clusters are spatially coherent, they do not reflect the spatial structure in this projection, suggesting that this clustering is not the most appropriate.

7 Conclusions and Future Work

DimStiller uses a set of abstractions to structure and navigate dimensional analysis and reduction: data resides in tables, operators modify and visualize tables, expressions chain together operators, and workflows permit pattern re-use. DimStiller uses these mechanisms to provide both local and global guidance through the analysis space of possible data tables. In both of our case studies we showed how individual operators probe the input dimensions and produce values like variance, correlations, and principal components and visualizations such as scree plots and scatterplots. It is the analyst's task to turn these quantitative figures into answers to qualitative questions about the data. DimStiller brings target users to these answers quickly by providing a simple pipeline architecture that users can rapidly step through after making adjustments to their data.

A form of guidance not yet explicitly provided by DimStiller is to help the user fully understand the inner workings of the supported operators. For example, users could be given visual feedback highlighting relevant regions of scree plots in the reduce operator control panels. While the DimStiller architecture should in theory support this kind of targeted exploration, it would require significant future work to design a system that truly sheds light on all black-box algorithms.

The most obvious direction for future work is to observe how users in several different application domains use DimStiller, both to help us understand how to improve the tool, and to obtain further insight into the tasks and workflows that comprise real-world dimensional analysis. The first deployment has been to researchers at the Department of Fisheries and Oceans, who are using the system to analyze both simulated and measured high-dimensional data with preliminary positive results. Another direction of future work will be to add more encoding and interaction techniques previously shown to be effective for high-dimensional analysis, for example sorting the `Collect` operator matrix view for the rank by feature capability suggested by Seo and Shneiderman [14].

Acknowledgements

This work was supported by the NSERC Strategic Grant "Visually Enhanced Exploration of High-Dimensional Data".

References

[1] I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[2] D. Cook and D. F. Swayne. *Interactive and Dynamic Graphics for Data Analysis: With Examples Using R and GGobi*. Springer, 2007.

[3] D. Guo. Coordinating computational and visual approaches for interactive feature selection and multivariate clustering. *Information Visualization*, 2(4):232–246, 2003.

[4] J. Heer and M. Agrawala. Software design patterns for information visualization. *IEEE Transactions on Visualization and Computer Graphics (Proc. InfoVis 2006)*, 12(5):853–860, 2006.

[5] M. Hein and J.-Y. Audibert. Intrinsic dimensionality estimation of submanifolds in R^d. In *Proc. Intl. Conf Machine Learning (ICML)*, pages 289–296. ACM, 2005.

[6] R. Holbrey. Dimension reduction algorithms for data mining and visualization. `http://www.comp.leeds.ac.uk/richardh/astro`, 2006.

[7] S. Ingram, T. Munzner, and M. Olano. Glimmer: Multilevel MDS on the GPU. *IEEE Trans. Visualization and Computer Graphics (TVCG)*, 15(2):249–261, 2009.

[8] S. Johansson and J. Johansson. Interactive dimensionality reduction through user-defined combinations of quality metrics. *IEEE Trans. Visualization and Computer Graphics (Proc. InfoVis 09)*, 15(6):993–1000, 2009.

[9] I. T. Jolliffe. *Principal Component Analysis, 2nd ed.* Springer, 2002.

[10] F. Jourdan and G. Melancon. Multiscale hybrid MDS. In *Proc. Intl. Conf. on Information Visualization (IV'04)*, pages 388–393, 2004.

[11] T. Munzner, A. Barsky, and M. Williams. Reflections on QuestVis: A visualization system for an environmental sustainability model. Technical Report TR-2009-24, UBC Computer Science, Nov. 2009.

[12] R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2008.

[13] G. Ross and M. Chalmers. A visual workspace for hybrid multidimensional scaling algorithms. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 91–96, 2003.

[14] J. Seo and B. Shneiderman. A rank-by-feature framework for interactive exploration of multidimensional data. *Information Visualization*, 4(2):99–113, 2005.

[15] M. Stone. Color in information display. IEEE Visualization 2006 Course Notes. `http://www.stonesc.com/Vis06`, Oct 2006.

[16] The MathWorks Inc. *MATLAB*. Natick, Massachusetts, 2010.

[17] M. O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proc. IEEE Conf. Visualization (Vis)*, pages 326–333, 1994.

[18] J. Yang, A. Patro, S. Huang, N. Mehta, M. O. Ward, and E. A. Rundensteiner. Value and relation display for interactive exploration of high dimensional datasets. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 73–80, 2004.

[19] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. In *Proc. IEEE Symp. Information Visualization (InfoVis)*, pages 105–112, 2003.

[20] J. Yang, M. O. Ward, E. A. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. In *Proc. Eurographics/IEEE Symp. Visualization (VisSym)*, pages 19–28, 2003.