Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

Sliceplorer: 1D slices for multi-dimensional continuous functions

T. Torsney-Weir[1], M. Sedlmair[1], and T. Möller[1]

[1]University of Vienna, Vienna, Austria

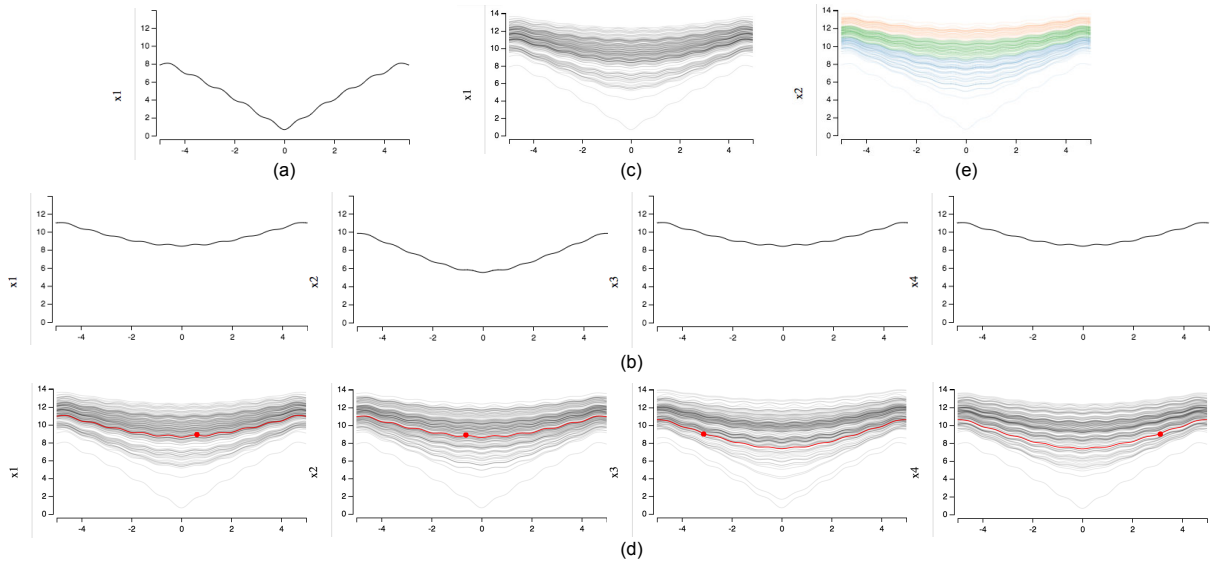


Figure 1: The evolution of our technique. We adapt the commonly known technique of 1D function plots (a) to multiple dimensions by taking a small multiples approach and repeating each plot for each dimension (b). We address the focus point selection problem by sampling over the parameter space and then projecting the slices in the corresponding plot (c). The user can mouse over a particular slice in one plot and the corresponding slices are highlighted in the other dimension plots (d). This allows one to see the corresponding function behaviors in the other dimensions. Finally, we can cluster the function slices (e), to show groups of similar behavior in the manifold.

Abstract

Multi-dimensional continuous functions are commonly visualized with 2D slices or topological views. Here, we explore 1D slices as an alternative approach to show such functions. Our goal with 1D slices is to combine the benefits of topological views, that is, screen space efficiency, with those of slices, that is a close resemblance of the underlying function. We compare 1D slices to 2D slices and topological views, first, by looking at their performance with respect to common function analysis tasks. We also demonstrate 3 usage scenarios: the 2D sinc function, neural network regression, and optimization traces. Based on this evaluation, we characterize the advantages and drawbacks of each of these approaches, and show how interaction can be used to overcome some of the shortcomings.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Motivation

The visual analysis of multi-dimensional scalar functions is a fundamental aspect in many areas, from computational sciences (based on computer simulations) to data sciences (based on machine learning techniques) and general optimization algorithms. Neural networks, for instance, have shown to produce very good results that come in the form of highly non-linear response manifolds. While understanding these manifolds would greatly help to verify the re-

sulting models, there is currently no established way to inspect these multi-dimensional manifolds. In case of an optimization algorithm, a good visualization of response manifolds could, for instance, help to see where possible "holes" (local extrema) are that the algorithm may get trapped in and fail to find the global optimum. By visually examining the functions that we are trying to optimize, one can develop new insights on how to improve the optimization algorithm. Our work is based on the assumption that visual analyses of these functions will help to increase understanding of what they are doing, as argued by Gleicher [Gle16].

To examine these functions visually we must reduce them to be displayed on a 2D screen. Currently, the two major approaches used to visualize these spaces are either two-dimensional slices, a technique known as HyperSlice [vWvL93] or dimension transformation techniques. This includes dimensionality reduction and topological methods [CLB11, CSA03]. HyperSlice shows two-dimensional slices (using either a heatmap or contour view) of the function directly around a particular focus point. It clearly shows the behavior of the function with respect to the parameters. However, one can only view one focus point at a time and the approach does not scale well to many dimensions. With each additional dimension one must substantially shrink the subplots so less detail can be seen just like in scatterplot matrices. Topological and dimensionality reduction techniques take the opposite approach. They morph the space to produce a 2D global view. However, the morphing process is rather complex and it is unclear what the resulting layout means. This reduces comprehension. Ideally we would like to somehow combine the global view with local detail.

In this paper, we explore the idea of 1D slices to fill this gap. We focus on multi-dimensional continuous scalar functions which we define as functions that take two or more scalar parameters as input and produce a single scalar as output. Towards illustrating the benefits, we propose a concrete technique using projections of 1D slices, which we call Sliceplorer (Figure 1). 1D slices are traditional function line plots familiar to anyone with basic mathematics knowledge: one dimensional curves with respect to changes in a single parameter. Like HyperSlice, we show a separate subplot for each input dimension. For 1D slices, the number of subplots scales linearly with the number of dimensions, not quadratically as with 2D slices. We address the issue of having to choose a focus point by sampling focus points and then showing all slices as a projection. Therefore, our 1D approach can be seen as a hybrid method of slicing and projection techniques.

In order to evaluate any new technique, we need to consider what data characteristics and tasks each method is good for. As of yet, there has not been a comprehensive listing of the tasks a user would want to perform when looking at multi-dimensional continuous functions. To this end, we begin development of this task summary by extending the task classification of Amar, Eagan, and Stasko [AES05]. We use this classification to evaluate our 1D approach and compare it to 2D slices and different topological approaches. This comparison allows us to characterize what technique is best for which tasks and reveals that 1D slices is the most flexible of the current approaches. That is, it supports the broadest range of different tasks.

We also provide three usage scenarios comparing our technique with other state of the art techniques. These scenarios illustrate that 1D slices can reveal structure in the functions that could not have been seen before. For example, we discuss how one can use our method to compare the *global* prediction manifold of a neural network algorithm against a support vector machine to guide and better understand the architecture of a neural network. This is currently an open research question in the machine learning community.

In summary our contributions are:

- exploring the idea of 1D slices with a concrete software prototype called Sliceplorer,
- a task-based analysis of multi-dimensional functions, comparing 1D slices to 2D HyperSlices and topological techniques, and
- three usage scenarios illustrating the value of 1D slices in common function visualization scenarios.

2. Related Work

The question of how to comprehend multi-dimensional data is a heavily researched area in visualization. There are two principal approaches, projection techniques (such as scatterplots) and slicing techniques (such as HyperSlice). Projection techniques generally show all of the data and, therefore, represent a more global view. On the other hand, slicing techniques present more of a local view around a point of interest (which we call focus point). The lion's share of previous work is concerned with projection approaches of discrete data. In contrast, we are focusing on continuous multi-dimensional scalar functions and seek to combine the strengths of projection and slicing. Given our focus on continuous data, our options for a visual exploration are limited and could be categorized into three areas (a) discretization, (b) local methods, (c) global methods.

2.1. Discretization

There is a number of different approaches to display discretized multi-dimensional functions. The typical approaches are scatterplot matrices [Har75], parallel coordinates [Ins85], star coordinates [Kan00], and RadVis [HGM*97]. Star coordinates and RadVis were generalized into one framework by Lehmann and Theisel [LT16]. These can all be combined with a variety of dimensionality reduction techniques [Hol06]. However, all of these seem inappropriate if the mental model of the function we are studying is a continuous one. In such a case all of these projection techniques would fail to properly convey the complexity of the underlying continuous phenomenon. Hence, while discretization seems like an easy way out, it is not a proper alternative for studying *continuous* multi-dimensional functions, such as regression functions or classification boundaries. Here, one of the main concerns is understanding a continuous phenomenon and a good visualization design should thus respect this mental model [TM04, SMM12, LS10].

2.2. Local methods

The idea of a local technique is to focus on a part of the function. Interaction is used to explore other parts of the function. One of the oldest approaches here is the HyperSlice technique by van Wijk and

van Liere [vWvL93]. HyperSlice is a technique where the function is shown directly but in multiple 2D slices laid out similar to a scatterplot matrix. In many ways, our work was inspired by this work. One of the drawbacks of HyperSlice is that one has to choose a focus point — a point common to all 2D slices. Exploring the full data set then shifts over to exploring all possible focus points. Although not created for HyperSlice specifically, techniques like the grand tour [Asi85], projection pursuit [Hub85], and optimal sets of projections [LT15] might be appropriate to tackle this issue. All of these approaches are still local though. A mental image of the global function can only be built up over time and with mental effort by browsing through the focus points. Our approach seeks to overcome this limitation while keeping the benefits of ease of understanding.

Note that for some tasks a local view might in fact suffice, such as when one is interested in the robustness of an extrema value. For example, Tuner [TWSM*11] used 2D continuous slices, letting the user navigate them via selecting Pareto optimal focus points in a separate view. Berger et al. [BPFG11] use coordinated views of scatterplots and parallel coordinates to show additional (continuous) prediction uncertainty. Having a discrete approach provided extra space to display information about the prediction uncertainty for the currently selected point.

2.3. Global methods

The visualization community has developed many global views of multi-dimensional continuous functions. Continuous scatterplots [BW08] and continuous parallel coordinates [HW09] can encode a multi-dimensional density field into either two dimensions or more than two dimensions respectively. Here, we are not as concerned with the density field, rather we are concerned with the manifold created from a continuous multi-dimensional scalar function.

Topological methods like topological spines [CLB11], the work by Gerber et al. [GBPW10], and contour trees [CSA03] extract extrema and saddle points from a function and then show these. These methods are good for seeing the relative relation of extrema in a function. However, they do not work for important tasks like robust optimization. Here, one does not necessarily want to find the global optimum but wants an optimum in a relatively "flat" area of the parameter space.

In our case, we use line plots together with the widely-used technique of projection to overdraw 1D slices. This approach is similar to the work by Hall et al. [HKC14] but differs in two major ways. (1) They showed 2 primary dimensions with slices and then used the third for color limiting their view to three dimensions and (2) they were concerned with isosurface extraction. Our technique can scale to any number of dimensions and we evaluate based on a much broader set of tasks and applications, such as parameter space analysis.

3. Sliceplorer

When developing our technique, we first identified design requirements with respect to tasks that a user would perform when analyzing multi-dimensional scalar functions. We continuously evaluated how our technique fits with these requirements and iteratively adapted it to encompass as many tasks as possible. A static slice view itself does not address many of the tasks required so we use interaction methods to address these and create a comprehensive technique.

3.1. Design requirements

When analyzing a function visually, there are a number of features that the user wants to see.

R-peaks: The most obvious feature is identifying peaks and valleys. This is primarily done to find the global optimum of a function.

R-robust: The relative height around each optima is important to detecting the robustness of that optimum. In some cases, one may prefer a local optimum over a global one if it is more stable. This is very common in simulations of manufacturing processes where variations in manufacturing tolerances should not affect the performance of a part too much [BPFG11].

R-bowl: Similarly, we may be interested in how "bowl"-shaped the area around an optimum is. The goal is similar to robust optimization but rather than looking for "flat" areas of the function we are looking for areas with smooth gradients. The amount of this smoothness is important for correctly parameterizing optimization algorithms [Bäc96]. An incorrect parameterization can either cause them to get "stuck" at some local minimum or make unreasonably slow progress towards the global minimum.

R-overall: Finally, we want to view the overall "shape" of the function. It is important to understand if it is smooth everywhere and how much variance there is in the function. When building a surrogate regression model we need to know if the function has consistent variance and we need to choose a model that captures this behavior.

All of these requirements mean that we need to view more than just the maxima and minima of a function.

3.2. Intuition

If we were analyzing one-dimensional continuous functions then the choice would be obvious: a line plot like the one in Figure 1a. The x-axis is used for the independent variable or parameter setting and the y-axis is used for the dependent variable or scalar value of the function. The function response is shown with a line. This is a metaphor that anyone who has taken high-school algebra can comprehend. The vertical and horizontal location channels visually encode the primary values of interest. These are the "best" visual channels to use in terms of accuracy and sensitivity to differences according to Bertin and others [Ber67, Mac86].

We show the evolution of our technique from a one-dimensional function to the full multi-dimensional Sliceplorer view in Figure 1. To extend this simple technique to multi-dimensional functions we simply repeat the one-dimensional function plot for each parameter (Figure 1b). We will have d plots where d is the number of dimensions in the function. In the same way that HyperSlice [vWvL93] is inspired by the SPLOM layout, we can use any layout technique for multiple histograms. 2D slices scale as $\mathcal{O}(d^2)$ which is worse than

the $\mathcal{O}(d)$ for 1D slices. 1D slices also give us separable channels remaining for encoding of additional information such as uncertainty or optimization traces (see Sec. 5.3).

Slicing offers a number of advantages over projection-based views like scatterplots and histograms. Slices give context around a particular focus point. We can see the precise shape of the function at this point. For example, peaks and valleys (**R-peaks**), flat areas (**R-robust**), and variance in the function can all be seen directly. While scatterplots and histograms can be used to see general trends, they suffer from "false distances" where points that are visibly close to each other are not actually close to each other.

3.3. Focus point projection

Showing a local 1D slice requires the selection of a single focus point, i.e. a point in multi-D through where all 1D slices intersect. Once this focus point is selected, we can use an off-the-shelf 1D function plot drawing method to draw the slice itself. Rather than only showing one focus point (i.e. one slice line per dimension) at a time and having the user choose a focus point we select multiple focus points automatically. This enables a more global view of the function (**R-overall**). Now, all 1D slices (one per focus point and dimension) are projected onto the same plot (see Figure 1c). In doing so, users do not need to memorize the previously seen slices, they can look among them to see general trends. This approach combines the ideas of slicing and projection, and fosters one of the core strengths of visualization: "perception beats recall" [Mun14].

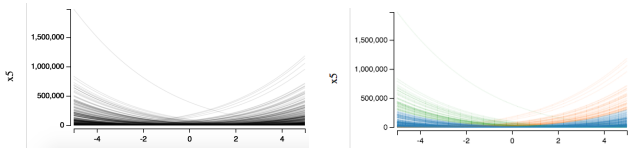
We are using a Sobol sequence [Sob67] to select the focus points themselves. The Sobol sequence is a space-filling, quasi-random, low-discrepancy sequence that is designed for sampling in high dimensional spaces. The Sobol sequence will give us a sampling of the multi-dimensional parameter space with an economy of samples. This will maximize the chances that we will see extrema (**R-peaks**), plateaus (**R-robust**), and bowls (**R-bowl**) in our 1D slices. In addition, using a Sobol sequence makes it easy to adjust the number of 1D slices shown (i.e. focus points) on the fly. Specifically, it avoids a complete resampling of the parameter space like we would need with a Latin hypercube sampling.

3.4. Linked selection

One disadvantage with 1D views over 2D views is that we cannot see the two-dimensional interactions anymore. We compensate for this with interaction. The user can mouse over a particular slice which will highlight all slices corresponding to that focus point. That is, one slice in each view is highlighted. We also superimpose the focus point itself on these lines. In doing so, the user can see the behavior of the function with respect to the other parameters around that focus point (see Figure 1d).

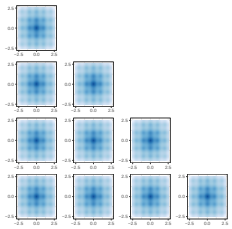
3.5. Clustering

Similar to visual encoding techniques such as parallel coordinate plots, projected 1D slices might mask certain patterns due to overdrawing. Figure 2a is an example where it is difficult to tell if the slices are monotonic or bowl-shaped (**R-bowl**). The interactive slice highlighting can give some insight into how individual slices

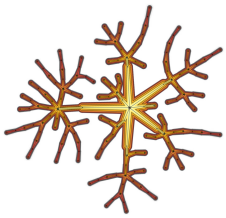


(a) (b)

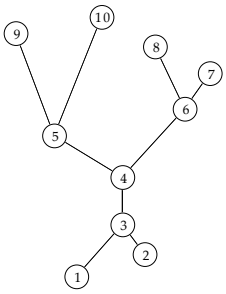
Figure 2: 500 projected slices of the 5th dimension of the 5D Zakharov [Bäc96] function. It is difficult to see if the slices are bowl shaped or two sets of monotonically decreasing and increasing curves. It is much clearer in the cluster view that there are actually three sets of curves: there is a set of monotonically decreasing curves and a set of monotonically increasing curves. The very low-value curves form a third set.



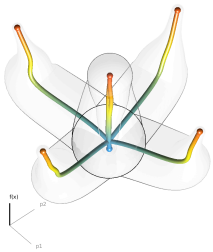
(a) HyperSlice [vWvL93]



(b) Topological spine [CLB11]



(c) Contour tree [CS03]



(d) Gerber et al. [GBPW10]

Figure 3: The four techniques we used to compare with 1D slices. With the exception of HyperSlice, the images are from the respective papers and show different datasets used in their context.

are behaving but lacks a global method to distinguish groups. We offer a clustered slice view to address this (Figure 2b). The clustering is done with a k-nearest neighbor algorithm using the L^2 distance between two slices as the distance metric. This allows us to group the slices into distinct groups of behavior and color-code these groups to distinguish them.

4. Task-based evaluation

We first evaluate 1D slices in terms of their flexibility to deal with a broad set of different low-level tasks. Task taxonomies give a basis for comparing visualization techniques to each other [Mun14]. If a technique addresses a large number of tasks, that is usually a good indicator of its flexibility. Over the last years, many different taxonomies have been proposed [AES05, BM13, HS12, SHB*14]. However, to the best of our knowledge none of these taxonomies

Task	Task description for discrete data items from [AES05]	Our adaption to continuous scalar functions	QRI results					Expert study results				
			HyperSlice	Topological spines	Contour tree	Gerber et al.	1D slices	HyperSlice	Topological spines	Contour tree	Gerber et al.	1D slices
Retrieve value	"Given a set of specific cases, find attributes of those cases"	Given an x, what is the function value?										
Filter	"Given some concrete conditions attribute values, find data cases satisfying those conditions."	For what parameter values is the function equal or over x?										
Compute derived value	"Given a set of data cases, compute an aggregate numeric representation of those data cases"	Summary statistics: variance, mean, SA										
Find extremum	"Find data cases possessing an extreme value of an attribute over its range within the data set"	Find local/global min/max										
Determine range	"Given a set of data cases and an attribute of interest, find the span of values within the set"	What is the range of possible outputs?										
Characterize distribution	"Given a set of data cases and a quantitative attribute of interest, characterize the distribution of that attribute's values over the set"	What types of shapes do the manifolds have										
Find anomalies	"Identify any anomalies within a given set of data cases with respect to a given relationship or expectation, e.g. statistical outliers"	Do areas of the manifold have shapes unlike any others										
Cluster	"Given a set of data cases, find clusters of similar attribute values"	Areas of the manifold have similar shapes										
Correlate	"Given a set of data cases and two attributes, determine useful relationships between the values of those attributes"	1D vs 2D relationships										

Table 1: We summarize our task-based evaluation here. We extended the discrete data-focused tasks of Amar, Eagan, and Stasko [AES05] to directly address continuous data, with the exception of "sort" (see section 4). We show the scores from our qualitative results inspection as well as our expert study on a scale from "none" , "partially" , "mostly" , to "fully" where "none" means that the task is not addressed at all and "fully" means that this task is directly supported by this view. We provide quotes of the general description from Amar, Eagan, and Stasko's paper in the "discrete" column for convenience.

thus far had a dedicated focus on the visual analysis of multi-dimensional *continuous* data. We thus took a popular taxonomy for tasks on discrete data, by Amar, Eagan, and Stasko [AES05], and extended each of their task categories to directly address continuous data. We see this as an initial step towards more consideration of multi-dimensional continuous data as a first class citizen when developing task hierarchies.

Using this list of tasks, we compare 1D slices to other state of the art techniques for multi-dimensional continuous data: HyperSlice [vWvL93], topological spines [CLB11], contour trees [CS03], and the technique by Gerber et al. [GBPW10] (see Figure 3). We refer to topological spines, contour trees, and the work by Gerber et al. as topological techniques when it makes sense to compare them as a group. We evaluate based on all tasks, except for "sort", for which we could not find a suitable extension to continuous functions. The guiding theme in our extensions is that users want to view the relationship of independent variables to the dependent variable and to see how the dependent variable changes with respect to the independent values. The extensions are shown in Table 1 along with the results of two investigations we conducted based on them, as detailed in the following section.

4.1. Study design

To perform a task-based evaluation, we investigated the different techniques in two different ways. First, we used a *qualitative result inspection* approach [IIC*13]. We (the authors) iteratively analyzed the techniques with different datasets and summarized our discussion and analysis on a four-point scale: "None" means that it is not possible to perform the task with the technique, "partly" means that it requires major interaction with the view to accomplish the task, "mostly" means that one can accomplish the task with little interaction, and "fully" means that this task is directly addressed by the technique.

Second, in order to get a more objective judgment we also asked *four visualization experts* familiar with examining multi-dimensional spaces like parameter space exploration to examine the eight datasets with different techniques and rate how well each task can be accomplished with each technique on the same scale. We averaged these results and show them along with the results of our qualitative result inspection in Figure 3.

For the techniques, we use our own implementation of HyperSlice and topological spines since no code was available. We used the `msr` R package [GP12] which implements the algorithm of Gerber et al. [GBPW10]. For the contour tree we used the `libtourtre` library and then rendered the trees using GraphViz using the Sugiyama [GKNV93] layout. As datasets we chose the 2D sinc function, 5D Rosenbrock [Ros60] function, 6D Ackley function, a 26 node hidden layer neural network built on the Boston housing dataset [Lic13], a support vector machine with Gaussian kernel built on the housing dataset, the fuel 3D volume dataset [Roe17], and the neghip 3D volume dataset [Roe17]. Not all datasets could be rendered with all techniques due to software errors.

4.2. Results

In this section, we summarize our discussion about the strengths and weaknesses of each technique in terms of performing the task. For more details, we also include as supplemental material a website that contains details of how each visualization technique can solve each task. The website is available at `http://sliceplorer.cs.univie.ac.at`.

Retrieve value: In the discrete case, the user should be able to look at a point and get the detailed values of it. In the continuous case we are interested in what the function value is for a certain input parameter setting. All the techniques support this although with the topological methods this is only possible for the extrema and saddle points as all other points are filtered out. For example,

there could be many points between node 4 and 5 in the contour tree (Figure 3c). With slicing techniques, both 1D and 2D, the values can be read directly off the chart. Of course, for all techniques the adding of interaction, such as a tooltip, can make retrieving concrete values even easier.

Filter: Amar, Eagan, and Stasko describe this task as a general filtering query on data points. In the continuous case, the user wants to understand the outputs of the function. This is a query as to where the function value is in a certain range. With continuous data this is the domain of isosurface extraction. This is possible with slicing techniques by visual examination. With HyperSlice, though, one must be careful to view sufficient focus points to get a general idea of where the function equals certain values. Topological spines also shows this directly and they use concentric areas (Figure 3b) to give a general idea of the area that a particular value range takes up. The other topological techniques allow one to see if a certain value is possible, for example, we can see that the function represented by the contour tree in Figure 3c takes the values greater than 4 somewhere by seeing that there are edges from node 4 to nodes 5 and 6. However, there is no relation back to the parameter settings that will produce these values.

Compute derived value: The direct interpretation of this task to continuous data is to compute derived value results about the curves like mean and variance. Many of these values can also be perceived visually. Topological methods compute the persistence value between the function to determine what to show but with the exception of topological spines this is hidden from the user. Topoplogical spines show a graph of the persistence and "saturated persistence" which allows the user to select which nodes to filter. Projections of 1D curves allows us to see the distribution. In Figure 1c we can see that there are very few function values around the global minimum and the function has two types of behavior: a periodic sine wave across the domain and a general parabola shape.

Find extremum: All the topological techniques we evaluated support this in some way. With HyperSlice one needs substantial guidance on setting the focus point to find extrema (like a histogram of function outputs). 1D slices is a global technique showing all slices at once, one can find extrema by inspecting the graphs. As previously mentioned, topological methods are purpose built to extract extrema from continuous data. For example, it is easy to see that the function using the method by Gerber et al. (Figure 3d) has five maxima and the function of the contour tree (Figure 3c) has four maxima.

Determine range: Amar, Eagan, and Stasko describe this as finding the range of possible values for a particular attribute. We really have only one attribute of interest: the values of the multi-dimensional scalar function. Any view from which we can read the global minimum or the global maximum allows us to do this. Contour trees, Gerber et al., and 1D slices all allow us to read these off the view. Topological spines either show the global maximum *or* the global minimum, but not both. HyperSlice has no way to do this directly by adjusting the focus point. However, one expert noticed that they could simply read the range of the function off of the color legend.

Characterize Distribution: Here again there is one key value of interest that we want to characterize: the function value. This requires a global view. Projections of slices directly show how the function slices are distributed. We can see in Figure 1d that there are very few function values around the global minimum but many around high values. It would be difficult to use HyperSlice to truly understand the distribution of values. The user would somehow have to browse around the focus points and then memorize the function values. Topology throws away the spatial element and just shows the relationships between extrema and saddles.

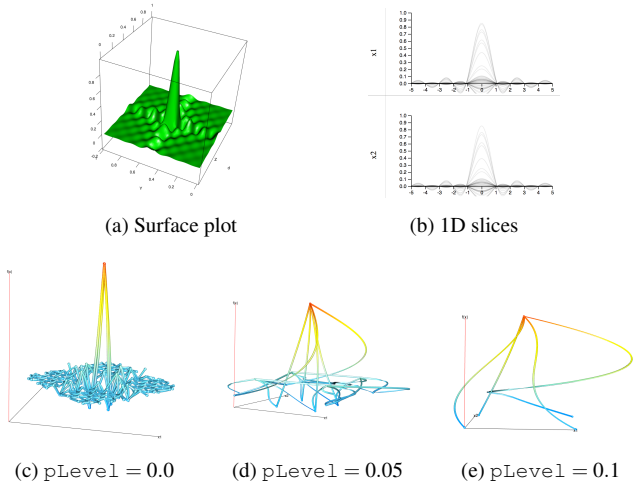
Find anomalies: Anomalies in the discrete case are single point outliers. While that is also possible in the continuous case, we may also have entire parts of the function that are unlike any other part. These should also be identified. In a global view like projected 1D slices these will show up visually. The slices will stand out from the rest similar to other projection-based techniques like scatterplots. With HyperSlice we must browse around until we can see one directly. However, we will see it if we can find it. Topological methods will only show extrema in terms of maxima or minima values but not shape and hence mask anomalies and outliers.

Cluster: Since we are looking at manifold behaviors, we want to be able to group the functions into areas of similar behavior. For example, are they monotonically increasing or decreasing? Furthermore, can we find areas where the variance changes? The topological technique of Gerber et al. [GBPW10] was created to address just this. They split the function into areas of monotonic behavior and then show a line indicating how those monotonic regions are related to each other. However, the way they reconstruct the function between extrema and saddle points does not allow us to view the variance between these points as the 1D slice view allows. Clustering the 1D slices tries to split the slices into groups of similar behavior (see Figure 2b).

Correlate: Finally, we consider correlation. In the discrete data case the goal is to find correlation between attributes. With continuous data, we already have a dependency between the independent and dependent variables. What we would like to learn is how many variables have an influence on the function. With 2D views (that only HyperSlice provides) one can see both 1D and 2D interactions with the function. We can see that the function in Figure 3a has radial behavior so the function value depends on both 1D and 2D interactions. None of the other techniques are capable of showing 2D interactions between parameters.

4.3. Summary

From the summary in Table 1 we can see that the 1D slices technique addresses more of the tasks than any other technique. It is not always the highest performing view though. HyperSlice is the only technique we evaluated that could show more than one-dimensional interactions but it does not do well on global tasks like extrema detection. The various topological techniques directly address tasks related to extrema detection and comparison but do not perform as well on others. The experts often commented that they felt they needed more knowledge about what exactly the topological techniques were doing in order to interpret the results. Thus, the ratings for these techniques may be artificially low. We conclude that 1D slices are a very flexible technique indeed.



(a) Surface plot (b) 1D slices

(c) pLevel = 0.0 (d) pLevel = 0.05 (e) pLevel = 0.1

Figure 4: Different views of the 2D sinc function. We show the surface plot in (a) for reference. Our 1D slice view is shown in (b). The central peak as well as the sub peaks are prominent. For comparison we show the method of Gerber et al. [GBPW10] in (c)–(e) at different levels of persistence filtering. With no filtering (c) the graph looks much like the original function. The plot is very sensitive to the filtering level. (d) and (e) are all very different from each other.

5. Usage scenarios

In addition to evaluating 1D slices with a low-level task hierarchy, we also provide usage scenarios to understand their value in real-world applications. We begin with an illustrative example using the 2D sinc function. We then use our 1D slices approach to illustrate how it can help better understand neural network architectures for a regression problem. Finally, we use 1D slices to investigate the effect of initial position on optimization algorithm performance.

The purpose of these evaluations is a proof of concept that 1D slices can be used for real-world problems. In particular, it is not meant as a comparative evaluation as provided in the previous section. To the best of our knowledge, neither HyperSlices nor topological techniques have been applied to understanding neural networks nor optimization algorithms so far. A full adaption of, and comparison to, these techniques for the provided use cases are beyond the scope of this paper, and are left for future work.

5.1. 2D sinc function

Imagining how a function in more than 3-dimensions looks is difficult if not impossible. In order to illustrate how Sliceplorer visualizes functions we show the 2D sinc function. We are using the formulation where $y(x_1,x_2) = \frac{sin(\pi x_1)}{\pi x_1}\frac{sin(\pi x_2)}{\pi x_2}$. In Figure 4a we show a 3D surface plot of this function. The global maximum is at $x_1,x_2 = 0,0$ where $y = 1$. There are a number of local maxima and minima of decreasing value radiating out from the origin.

We show the 1D slice view using Sliceplorer in Figure 4b. We are showing 50 slices in each of the 2 plots. We can clearly see that the maximum value occurs when $x_1,x_2 = 0,0$ in the graph at around $y = 1$. We can also see the decreasing extrema radiating out from the origin. We can also precisely measure the height and x-location of the extrema. If we want to examine a particular trace then we can highlight it in the view and see the full slice highlighted on screen.

For comparison, we show visualizations of the 2D sinc function rendered using the `msr` package [GP12] in R. This package implements the visualization of the Morse-Smale complex from Gerber et al. [GBPW10]. We sampled the function with 2000 sample points using a Sobol sequence. The 1D slices view is showing 50 focus points with 21 samples for each slice so this was done to use a similar sampling method and number of samples to the Sliceplorer method. The function can do persistence-based filtering of the graph before rendering. This is controlled by the `pLevel` parameter which filters all persistences less than a certain value. In Figure 4c we show the view with the filtering level set to 0, i.e. no filtering. The view does a very good job showing the critical points of the graph. It looks very similar to the surface plot (Figure 4a). However, the visualization is very sensitive to the filtering level. In Figure 4d and Figure 4e we show the sinc function with the filtering level set to 0.05 and 0.1 respectively. The 1D slice view does not suffer from this issue of parameterization.

5.2. Neural networks

Artificial neural networks are currently gaining a lot of attention in machine learning. The goal of these algorithms is to produce a multi-dimensional function fitted to the training points. Neural networks, in particular, have proven to be very good at producing accurate, generalizable predictions. One of the major challenges for designers of such models, however, is to properly architect these networks. For instance, how many hidden layers does one need and how many nodes should be put into each layer? These architectural choices can drastically change the predictions. While these choices are crucial, currently, there is only little guidance available for designers. A typical rule of thumb is to use a hidden layer two times the size of the input dimensions. There are also some general proofs regarding what type of functions neural networks can represent [HSW89, ES16]. However, there are no formal guidelines for designing these networks [GBC16] and the way these models make predictions is still obtuse.

One of the ways we can increase the understandability of neural network regression models is by viewing the response function directly [Gle16]. If we want to understand how the network architecture affects the prediction we could compare the prediction manifold to one produced by a "simpler" machine learning model [RSG16], for example a support vector machine [SS04]. Support vector machines have known guarantees on error rate with the number of training samples. With this comparison we may be able to get some better insight about how the neural network learning algorithms are performing.

To compare, we chose the Boston housing dataset [Lic13] from the UCI repository. This dataset contains median home prices given 13 factors including crime rate, age of the house, and proximity to highways. We then trained a neural network with a single hidden layer of 26 nodes, a neural network with 2 hidden layers: one of 5 and one of 3 nodes, a support vector machine with a polynomial

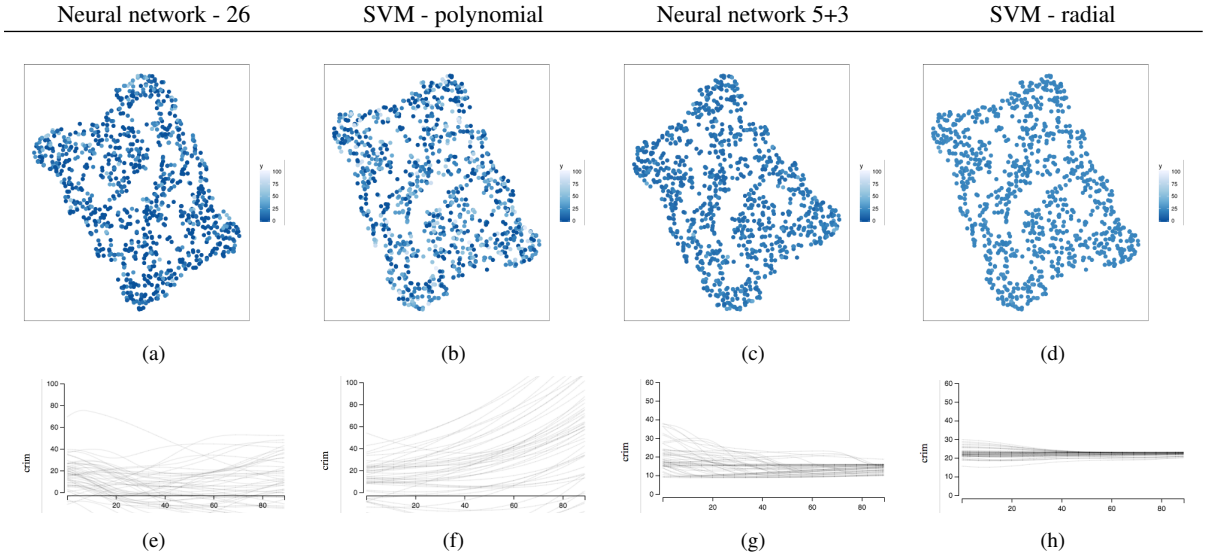


Figure 5: Two different views of the predictions of four different machine learning regression models on the Boston housing dataset. The top row (a – d) shows predictions by each model compressed to two dimensions with t-SNE. We colored the points on a continuous color scale with dark blue being 0 and light blue the highest value. The bottom row (e – h) shows a 1D slice view of the first dimension of the dataset, crime rate. We show the slices of the remaining dimensions in the supplemental material. The 1D slices reveal interesting information about how the models perform and can assist with model selection. We may not want to use the SVM with polynomial kernel (f), for example, since it predicts that home price will go up with higher crime rates.

kernel, and a support vector machine with a radial (Gaussian) kernel.

We compare 1D slices with an adaption of the common way of viewing classification algorithms to continuous data. The results of *classification* models are commonly visualized by using MDS [Kru64] or t-SNE [vdMH08] to reduce the input dimensions to two and then present a scatterplot with the predictions colored by class. We extended this by sampling the prediction model with 1,000 samples from a Latin hypercube [Tan93] (a space-filling design) converting the points to two dimensions with t-SNE and then coloring the points on a continuous scale which we show in Figure 5 (top row). The bottom row of the figure shows the 1D slice view of the same four prediction models. We only show the first dimension due to space reasons. The full 13 dimension slice view image is in the supplemental material.

Showing the changes in home price as it corresponds directly to the crime rate can help to increase confidence in a model. From the prediction lines, one may not want to use the SVM with a polynomial kernel. By and large the prediction lines are increasing. This means that the home price is increasing as the crime rate goes up. This does not really make sense. The model is not generalizing well. Similarly, the neural network with a single hidden layer (left column) also has a number of curves that increase as crime increases. The neural network with two hidden layers does not have this problem. Maybe this is the best model to use in this case.

In summary, this usage scenario illustrated that a direct inspection of a model's response surfaces can give intuition of its behavior, and can lead to a better model selection and a better intuition of the modeling process. 1D slices can help to gain important insights in this process.

5.3. Optimization algorithm

General purpose optimization algorithms try to find the global minimum (or equivalently, the global maximum) of a function of arbitrary dimension. Many optimization algorithms such as Nelder-Mead [NM65] work by starting at a particular parameter setting and evaluating the "shape" of the function around that point. The algorithm then determines where the function is decreasing the greatest, and "jumps" a certain distance in that direction. The "jump", starting position, and termination tolerance parameters are user-settable parameters. Depending on how they are set, the algorithm can get stuck in a local minima or take unreasonably long to finish.

When one is trying to parameterize or build optimization algorithms then one wants to evaluate the trace of the optimization on an easy function that is fast to compute first. This analysis helps to better understand how to parameterize for more complex problems but are too computationally expensive to analyze directly. Visual inspection of the easy function before running the optimization algorithm, as well as viewing the trace of the optimization algorithm (the sequence of steps it took) is a good way to ensure that the algorithm is converging towards the global minimum.

We compare 1D slices with HyperSlice [vWvL93] as this is the only technique that also directly visualizes the parameter space. We ran the Nelder-Mead optimization algorithm on the 5D Ackley function [Ack87], a popular optimization algorithm testing function. To examine the effect of starting position, we tried different

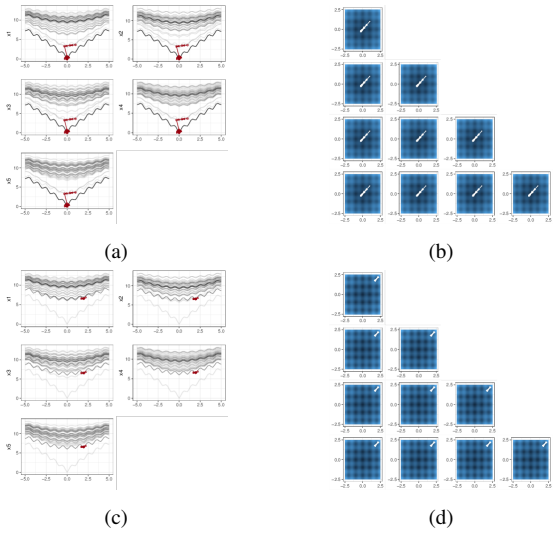


Figure 6: 1D slice and HyperSlice views showing traces of an optimization algorithm searching for the global minimum of a 5D Ackley function. (a) and (b) show the trace starting at the point $(1,1,1,1,1)$ while (c) and (d) show the trace starting at the point $(2,2,2,2,2)$.

starting positions: $x = (1,1,1,1,1)$ and $x = (2,2,2,2,2)$. We overlay the optimization trace on top of both 1D slices and HyperSlice and the results are shown in Figure 6.

The 1D slice view allows us to see the path that the algorithm took and the general shape of the function simultaneously. In addition, the 1D slice view shows that the distribution of values around the global minimum is small. Most of the slices are clustered around $y = 10$ with only one slice descending close to $y = 0$. Since the sampling is uniform in the parameter space this means that it is very difficult to select slices around the global minimum. In fact, this is a known property of the Ackley function. It is easy to see that the optimization algorithm got stuck at a local minimum when started at $x = (2,2,2,2,2)$. However, with the HyperSlice view it is difficult to see the difference in value and steepness of the function at $x = (1,1,1,1,1)$ versus $x = (2,2,2,2,2)$. Humans are not good at perceiving fine differences in color [Mun14], but is required for this task. We learn a lot more about the behavior of the optimization algorithm from the 1D slice views (see Figure 6a and Figure 6c) than the HyperSlice view. However, the HyperSlice view does clearly show that that optimization algorithm is moving in multiple directions at once. This is not clear in the 1D slice views.

6. Discussion

The above examples illustrated that the technique of 1D slices as presented in our paper is quite flexible and useful for various low- and high-level tasks. However, we do not intend to claim that it is the only and best method for all problems out there. We rather would like to argue that it is a valuable (and thus far overlooked) technique in a toolbox of visual inspection methods for multi-dimensional functions. We hope that this paper inspires a discussion and exploration of guidelines for tasks, proper visual encoding, and interaction techniques for various tasks. Along these lines we would like to put forth our current experience with various techniques.

Topological techniques are helpful for a global overview: Topological techniques allow us to compare *between* optima but are not as good at evaluating the area *around* an optimum since these areas are typically abstracted away. Topological spines attempts to compensate for this by showing the area covered by a particular optimum as an area around the node. However, many of the tasks like "correlate" and "cluster" are best served by viewing the response manifold directly. In a larger system, the topological techniques could be used effectively as a global overview of the function with a HyperSlice or 1D slice showing local context. Selecting a point in the topological view would change the focus point in the local view.

HyperSlice is good when you need to show 2D interactions: HyperSlice is the only technique that can display more than one dimension of data interaction. So, if this is a requirement then HyperSlice is the best option. However, one can use 1D slices to get a general overview of the dependence of the function on each dimension. The dimensions that are not interesting because, for example, the function is not sensitive to them could easily be eliminated from further consideration. This would reduce the number of subplots that we need to view in the HyperSlice plot.

1D slices should be used for a "first pass" visualization: 1D slices addresses many of the tasks that a user wants to perform. The technique does a very good job on a wide variety of tasks. 1D slices are easy to implement, easy to understand, and the static view provides a lot of information.

7. Limitations and future work

Our 1D slice view consists of a projection of many lines. We are showing the distribution of slices through direct projection. Techniques like contour boxplots [WMK13] and curve boxplots [MWK14] build a distribution model of curves which could help to address the "characterize distribution" task in Table 1. However, neither of these or any of the other time curve visualization techniques have been applied to multi-dimensional functions. Evaluating these techniques for this purpose is an exciting topic for future work.

When developing the 1D continuous slicing technique we only considered multi-dimensional continuous scalar functions in terms of requirements, tasks, and comparisons. We do not consider multi-field (i.e. functions with multiple outputs) or complex-valued functions in our analysis. There are multi-field topology techniques to address this [DCK*12, HHG14, CGT*15] which we do not consider but our technique and analysis would need to be extended to this domain. We leave this for future work.

The x-axis of each 1D slice is independent of the x-axes of the other 1D slices. This allows each plot to scale individually if the range of inputs have different values. The x-axis and y-axis automatically change to incorporate their respective minimum and

maximum ranges. While the x-axis scales itself independently, the y-axis is the same for each plot. This is also the default behavior in many of-the-shelf plotting packages. The plots will adjust automatically to shifts. For the x-axis we use axis-aligned projections. Therefore, the views are sensitive to rotational transformations of the function.

Finally, our technique is also based on sampling, just like the other techniques we are comparing to. As with any technique based on sampling one must be careful to take an adequate number of samples in order to properly capture all desired behavior. If the function is not smooth we may see a slice that is an "outlier," i.e. one slice is much higher or lower than all the others. In this case all other slices will be compressed into either the top or bottom of the chart. This is often a problem with many common visualization techniques like bar graphs or scatterplots and can be addressed with log scaled axes, for example.

8. Conclusion

In this paper we have presented Sliceplorer, a visualization method for multi-dimensional functions based on one-dimensional slices. We defined a task taxonomy specific to multi-dimensional continuous functions and found that, while some state of the art techniques are very good at addressing specific tasks, our method supports a wide variety of tasks. Consequently, our technique may be a good first pass when visualizing multi-dimensional continuous functions. It is easy to implement, easy to understand, and addresses a greater variety of tasks than any other technique.

Acknowledgments

We wish to thank the members of the VDA lab for their helpful feedback and support with this project especially Peter Ruch, Michael Oppermann, and Patrick Wolf. We also wish to thank Harald Piringer, Eduard Gröller, Ivan Viola for their feedback on the project. The authors wish to especially thank Johanna Schlereth for her invaluable help with making the video. This work was partly supported by the Vienna Business Agency.

References

[Ack87] ACKLEY D. H.: *A Connectionist Machine for Genetic Hill-climbing*. The Kluwer International Series in Engineering and Computer Science. Springer, 1987. doi:10.1007/978-1-4613-1997-9. 8

[AES05] AMAR R., EAGAN J., STASKO J. T.: Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization (InfoVis)* (Oct. 2005), IEEE Computer Society, pp. 111–117. doi:10.1109/INFVIS.2005.1532136. 2, 4, 5

[Asi85] ASIMOV D.: The grand tour: A tool for viewing multidimensional data. *SIAM Journal on Scientific and Statistical Computing 6*, 1 (Jan. 1985), 128. doi:10.1137/0906011. 3

[Bäc96] BÄCK T.: *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, USA, 1996. 3, 4

[Ber67] BERTIN J.: *Sémiologie Graphique: Les diagrammes, les réseaux, les cartes*. Gauthier-Villars, 1967. 3

[BM13] BREHMER M., MUNZNER T.: A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2376–2385. doi:10.1109/tvcg.2013.124. 4

[BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. *Computer Graphics Forum 30*, 3 (June 2011), 911–920. doi:10.1111/j.1467-8659.2011.01940.x. 3

[BW08] BACHTHALER S., WEISKOPF D.: Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Dec. 2008), 1428–1435. doi:10.1109/TVCG.2008.119. 3

[CGT*15] CARR H., GENG Z., TIERNY J., CHATTOPADHYAY A., KNOLL A.: Fiber surfaces: Generalizing isosurfaces to bivariate data. *Computer Graphics Forum 34*, 3 (June 2015), 241–250. doi:10.1111/cgf.12636. 9

[CLB11] CORREA C. D., LINDSTROM P., BREMER P.-T.: Topological spines: A structure-preserving visual representation of scalar fields. *IEEE Transactions on Visualization and Computer Graphics 17*, 6 (Aug. 2011), 1842–1851. doi:10.1109/TVCG.2011.244. 2, 3, 4, 5

[CS03] CARR H., SNOEYINK J.: Path seeds and flexible isosurfaces using topology for exploratory visualization. In *Eurographics / IEEE VGTC Symposium on Visualization* (May 2003), Eurographics Association, pp. 49–58. doi:10.2312/VisSym/VisSym03/049-058. 4, 5

[CSA03] CARR H., SNOEYINK J., AXEN U.: Computing contour trees in all dimensions. *Computational Geometry 24*, 2 (Feb. 2003), 75–94. doi:10.1016/S0925-7721(02)00093-7. 2, 3

[DCK*12] DUKE D., CARR H., KNOLL A., SCHUNCK N., NAM H. A., STASZCZAK A.: Visualizing nuclear scission through a multifield extension of topological analysis. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Oct. 2012), 2033–2040. doi:10.1109/TVCG.2012.287. 9

[ES16] ELDAN R., SHAMIR O.: The power of depth for feedforward neural networks. In *Conference on Learning Theory* (June 2016), pp. 907–940. arXiv:1512.03965. 7

[GBC16] GOODFELLOW I., BENGIO Y., COURVILLE A.: *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org. 7

[GBPW10] GERBER S., BREMER P.-T., PASCUCCI V., WHITAKER R.: Visual exploration of high dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov./Dec. 2010), 1271–1280. doi:10.1109/TVCG.2010.213. 3, 4, 5, 6, 7

[GKNV93] GANSNER E. R., KOUTSOFIOS E., NORTH S. C., VO K.-P.: A technique for drawing directed graphs. *IEEE Transactions on Software Engineering 19*, 3 (Mar. 1993), 214–230. doi:10.1109/32.221135. 5

[Gle16] GLEICHER M.: A framework for considering comprehensibility in modeling. *Big Data 4*, 2 (June 2016), 75–88. doi:10.1089/big.2016.0007. 2, 7

[GP12] GERBER S., POTTER K.: Data analysis with the Morse-Smale complex: The msr package for R. *Journal of Statistical Software 50*, 2 (July 2012), 1–22. doi:10.18637/jss.v050.i02. 5, 7

[Har75] HARTIGAN J. A.: Printer graphics for clustering. *Journal of Statistical Computation and Simulation 4*, 3 (June 1975), 187–213. doi:10.1080/00949657508810123. 2

[HGM*97] HOFFMAN P., GRINSTEIN G., MARX K., GROSSE I., STANLEY E.: DNA visual and analytic data mining. In *Proceedings of the 8th conference on Visualization '97* (Oct. 1997), IEEE Computer Society, pp. 437–441. doi:10.1109/visual.1997.663916. 2

[HHG14] HUETTENBERGER L., HEINE C., GARTH C.: Decomposition and simplification of multivariate data using pareto sets. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec. 2014), 2684–2693. doi:10.1109/tvcg.2014.2346447. 9

[HKC14] HALL K. W., KUSALIK P. G., CARPENDALE S.: Profile contour plots: Alternate projections of 3D free energy surfaces. VisWeek 2014 Poster Compendium, 2014. 3

[Hol06] HOLBREY R.: *Dimension reduction algorithms for data mining and visualization*. Tech. rep., University of Leeds/Edinburgh, 2006. 2

[HS12] HEER J., SHNEIDERMAN B.: Interactive dynamics for visual analysis. *Queue 10*, 2 (Feb. 2012), 1–26. doi:10.1145/2133416.2146416. 4

[HSW89] HORNIK K., STINCHCOMBE M., WHITE H.: Multilayer feedforward networks are universal approximators. *Neural Networks 2*, 5 (1989), 359 – 366. doi:10.1016/0893-6080(89)90020-8. 7

[Hub85] HUBER P. J.: Projection pursuit. *The Annals of Statistics 13*, 2 (June 1985), 435–475. doi:10.1214/aos/1176349519. 3

[HW09] HEINRICH J., WEISKOPF D.: Continuous parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (Nov./Dec. 2009), 1531–1538. doi:10.1109/TVCG.2009.131. 3

[IIC*13] ISENBERG T., ISENBERG P., CHEN J., SEDLMAIR M., MÖLLER T.: A systematic review on the practice of evaluating visualization. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2818–2827. doi:10.1109/tvcg.2013.126. 5

[Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer 1*, 2 (Aug. 1985), 69–91. doi:10.1007/BF01898350. 2

[Kan00] KANDOGAN E.: Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions. In *IEEE Information Visualization Symposium, Late Breaking Hot Topics* (2000), IEEE Computer Society, pp. 9–12. 2

[Kru64] KRUSKAL J. B.: Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (Mar. 1964), 1–27. doi:10.1007/BF02289565. 8

[Lic13] LICHMAN M.: UCI machine learning repository, 2013. URL: http://archive.ics.uci.edu/ml. 5, 7

[LS10] LIU Z., STASKO J. T.: Mental models, visual reasoning and interaction in information visualization: A top-down perspective. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov. 2010), 999–1008. doi:10.1109/tvcg.2010.177. 2

[LT15] LEHMANN D. J., THEISEL H.: Optimal sets of projections of high-dimensional data. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (Jan. 2015), 609–618. doi:10.1109/tvcg.2015.2467132. 3

[LT16] LEHMANN D. J., THEISEL H.: General projective maps for multidimensional data projection. *Computer Graphics Forum 35*, 2 (May 2016), 443–453. doi:10.1111/cgf.12845. 2

[Mac86] MACKINLAY J.: Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics 5*, 2 (Apr. 1986), 110–141. doi:10.1145/22949.22950. 3

[Mun14] MUNZNER T.: *Visualization Analysis and Design*. AK Peters Visualization Series. CRC Press, 2014. 4, 9

[MWK14] MIRZARGAR M., WHITAKER R. T., KIRBY R. M.: Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec. 2014), 2654–2663. doi:10.1109/tvcg.2014.2346455. 9

[NM65] NELDER J. A., MEAD R.: A simplex method for function minimization. *The Computer Journal 7*, 4 (Jan. 1965), 308–313. doi:10.1093/comjnl/7.4.308. 8

[Roe17] ROETTGER S.: The volume library, 2017. URL: http://www9.informatik.uni-erlangen.de/External/vollib/ [cited March 10, 2017]. 5

[Ros60] ROSENBROCK H. H.: An automatic method for finding the greatest or least value of a function. *The Computer Journal 3*, 3 (Jan. 1960), 175. doi:10.1093/comjnl/3.3.175. 5

[RSG16] RIBEIRO M. T., SINGH S., GUESTRIN C.: "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug. 2016), ACM, pp. 1135–1144. doi:10.1145/2939672.2939778. 7

[SHB*14] SEDLMAIR M., HEINZL C., BRUCKNER S., PIRINGER H., MÖLLER T.: Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec. 2014), 2161–2170. doi:10.1109/tvcg.2014.2346321. 4

[SMM12] SEDLMAIR M., MEYER M., MUNZNER T.: Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Nov./Dec. 2012), 2431–2440. doi:10.1109/TVCG.2012.213. 2

[Sob67] SOBOL' I. M.: On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki 7*, 4 (1967), 784–802. doi:10.1016/0041-5553(67)90144-9. 4

[SS04] SMOLA A. J., SCHÖLKOPF B.: A tutorial on support vector regression. *Statistics and Computing 14*, 3 (Aug. 2004), 199–222. doi:10.1023/B:STCO.0000035301.49549.88. 7

[Tan93] TANG B.: Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association 88*, 424 (Dec. 1993), 1392–1397. doi:10.1080/01621459.1993.10476423. 8

[TM04] TORY M., MÖLLER T.: Rethinking visualization: A high-level taxonomy. In *IEEE Symposium on Information Visualization* (Oct. 2004), IEEE Computer Society, pp. 151–158. doi:10.1109/INFVIS.2004.59. 2

[TWSM*11] TORSNEY-WEIR T., SAAD A., MÖLLER T., WEBER B., HEGE H.-C., VERBAVATZ J.-M., BERGNER S.: Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (Nov./Dec. 2011), 1892–1901. doi:10.1109/TVCG.2011.248. 3

[vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9* (Nov. 2008), 2579–2605. 8

[vWvL93] VAN WIJK J. J., VAN LIERE R.: HyperSlice: Visualization of scalar functions of many variables. In *Proceedings of the 4th Conference on Visualization* (Oct. 1993), IEEE Computer Society, pp. 119–125. doi:10.1109/VISUAL.1993.398859. 2, 3, 4, 5, 8

[WMK13] WHITAKER R. T., MIRZARGAR M., KIRBY R. M.: Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (Dec. 2013), 2713–2722. doi:10.1109/TVCG.2013.143. 9