Journal of Neural Engineering

PAPER

DataHigh: graphical user interface for visualizing and interacting with high-dimensional neural activity

To cite this article: Benjamin R Cowley *et al* 2013 *J. Neural Eng.* **10** 066012

J. Neural Eng. **10** (2013) 066012 (19pp) doi:10.1088/1741-2560/10/6/066012

DataHigh: graphical user interface for visualizing and interacting with high-dimensional neural activity

Benjamin R Cowley[1,2], **Matthew T Kaufman**[3,4,5], **Zachary S Butler**[6,7], **Mark M Churchland**[3,4,8], **Stephen I Ryu**[4,9], **Krishna V Shenoy**[3,4,10,11] **and Byron M Yu**[2,12,13]

[1] Department of Machine Learning, Carnegie Mellon University, Pittsburgh, PA, USA
[2] Center for Neural Basis of Cognition, Carnegie Mellon University, Pittsburgh, PA, USA
[3] Neurosciences Program, Stanford University, Stanford, CA, USA
[4] Department of Electrical Engineering, Stanford University, Stanford, CA, USA
[5] Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, USA
[6] Department of Electrical Engineering, Grinnell College, Grinnell, IA, USA
[7] Department of Computer Science, University of California-Irvine, Irvine, CA, USA
[8] Department of Neuroscience, Columbia University Medical School, New York, NY, USA
[9] Department of Neurosurgery, Palo Alto Medical Foundation, Palo Alto, CA, USA
[10] Department of Bioengineering, Stanford University, Stanford, CA, USA
[11] Department of Neurobiology, Stanford University, Stanford, CA, USA
[12] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
[13] Department of Biomedical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

E-mail: byronyu@cmu.edu

Received 24 June 2013
Accepted for publication 2 October 2013
Published 12 November 2013
Online at stacks.iop.org/JNE/10/066012

Abstract

Objective. Analyzing and interpreting the activity of a heterogeneous population of neurons can be challenging, especially as the number of neurons, experimental trials, and experimental conditions increases. One approach is to extract a set of latent variables that succinctly captures the prominent co-fluctuation patterns across the neural population. A key problem is that the number of latent variables needed to adequately describe the population activity is often greater than 3, thereby preventing direct visualization of the latent space. By visualizing a small number of 2-d projections of the latent space or each latent variable individually, it is easy to miss salient features of the population activity. *Approach.* To address this limitation, we developed a Matlab graphical user interface (called DataHigh) that allows the user to quickly and smoothly navigate through a continuum of different 2-d projections of the latent space. We also implemented a suite of additional visualization tools (including playing out population activity timecourses as a movie and displaying summary statistics, such as covariance ellipses and average timecourses) and an optional tool for performing dimensionality reduction. *Main results.* To demonstrate the utility and versatility of DataHigh, we used it to analyze single-trial spike count and single-trial timecourse population activity recorded using a multi-electrode array, as well as trial-averaged population activity recorded using single electrodes. *Significance.* DataHigh was developed to fulfil a need for visualization in exploratory neural data analysis, which can provide intuition that is critical for building scientific hypotheses and models of population activity.

(Some figures may appear in colour only in the online journal)

1. Introduction

A major challenge in systems neuroscience is to interpret the activity of large populations of neurons, which may be recorded either simultaneously or sequentially (Stevenson and Kording 2011). During exploratory data analysis, there are several key benefits to analyzing the activity of a population of neurons together. First, instead of averaging across experimental trials, we can leverage the statistical power of the recorded population to denoise and analyze the neural activity on a single-trial basis (Yu *et al* 2009, Churchland *et al* 2007). Second, salient structure in the neural population dynamics may be more easily discernible when considering the activity of many neurons at once rather than the activity of one neuron at a time (Churchland *et al* 2012, Mante *et al* 2013, Stopfer *et al* 2003). Third, this allows us to embrace the heterogeneity of the activity of different neurons (Churchland and Shenoy 2007, Machens *et al* 2010), in contrast to selectively analyzing a subset of the recorded neurons that appear to be most interpretable.

To understand how population activity differs across individual experimental trials, one might display the raster plot for each trial, where a tick mark represents a neuron's action potential (figure 1(A)). As the number of neurons and trials grows, it can be difficult to pick out key features in the raster plots that differentiate one trial from another (Churchland *et al* 2007). In addition, one may seek to understand how population activity differs across experimental conditions. A common approach is to average the spike trains across trials to create a peri-stimulus time histogram (PSTH) for each neuron and experimental condition (figure 1(B)). As the number of neurons and conditions increases, the task of comparing population dynamics across different conditions can be challenging due to the heterogeneity of the PSTHs (Churchland and Shenoy 2007, Machens *et al* 2010, Mante *et al* 2013, Rigotti *et al* 2013).

To overcome these difficulties, we can extract a smaller number of *latent variables* that succinctly summarize the population activity for each experimental trial (figure 1(C)) or for each experimental condition (figure 1(D)). There are two complementary ways of understanding the relationship between the latent variables and the recorded neural activity. First, the latent variables can be viewed as 'readouts' of the population activity, where each latent variable captures a prominent co-fluctuation pattern among the recorded neurons. The latent variables can be obtained by simply adding and subtracting the activity of different neurons, while possibly incorporating smoothing in time. Second, because these latent variables capture the most prominent co-fluctuation patterns, the population activity can be 'reconstructed' by adding and subtracting the patterns in different ways for different trials or conditions. This approach has been applied to study the motor system (Yu *et al* 2009, Churchland *et al* 2010, 2012, Afshar *et al* 2011, Shenoy *et al* 2013), olfactory system (Stopfer *et al* 2003, Mazor and Laurent 2005, Broome *et al* 2006), visual system (Churchland *et al* 2010), auditory system (Luczak *et al* 2009, Bartho *et al* 2009, Bouchard *et al* 2013), working memory and decision making (Machens *et al* 2010, Harvey *et al* 2012), visual attention (Cohen and Maunsell 2010), and rule learning in prefrontal cortex (Durstewitz *et al* 2010). Dimensionality reduction methods to extract these latent variables include principal component analysis (PCA) (Machens *et al* 2010, Stopfer *et al* 2003, Mazor and Laurent 2005, Briggman *et al* 2005), factor analysis (FA) (Santhanam *et al* 2009), Gaussian-process factor analysis (GPFA) (Yu *et al* 2009), and locally-linear embedding (LLE) (Stopfer *et al* 2003).

The basic setup is to first define an n-dimensional space, where each axis represents the firing rate of one of the n neurons in the population. A dimensionality reduction method is then applied to the n-dimensional population activity to determine the number of latent variables, k, needed to adequately describe the population activity ($k < n$), as well as the relationship between the latent variables and the population activity (figure 2). These latent variables define a reduced k-dimensional (k-d) *latent space* in which we can study how the population activity varies over time, across trials, and across experimental conditions. Ideally, we would like to visualize the latent variables directly in the k-dimensional space. However, the number of latent variables, k, is typically greater than 3 (Yu *et al* 2009, Santhanam *et al* 2009, Machens *et al* 2010) and direct plotting can only provide a two-dimensional (2-d) or three-dimensional (3-d) view. If specific features of interest of the population activity are known, one approach is to specify a cost function to find a 2-d projection of the k-d space that illustrates those features (Churchland *et al* 2012). However, in exploratory data analysis, such features may be unknown in advance, and viewing a single 2-d projection can be misleading. As illustrated in figure 2, the same 6-d latent space can yield rather different looking 2-d projections. This underscores the need to look at many 2-d projections to obtain a more complete picture of the high-dimensional structure of population activity.

To quickly and smoothly view many 2-d projections, we developed an interactive graphical user interface (GUI), called *DataHigh*, in Matlab for visualization of the k-d latent space. The user uploads raw spike trains to DataHigh, which then guides the user to perform dimensionality reduction and to quickly visualize a continuum of 2-d projections. We found DataHigh to be a valuable tool for building intuition about population activity, for hypothesis generation, and for development of models of population activity. Although high-dimensional visualization is a challenge across many scientific fields, DataHigh has tools tailored to neural data analysis that are currently not present in general-purpose high-dimensional visualization software (Swayne *et al* 2003). DataHigh is versatile and can be used to study population activity recorded either simultaneously (using multi-electrode arrays) or sequentially (using conventional single electrodes). The population activity may be in the form of single-trial spike count vectors (taken in a single time bin on each experimental trial), single-trial timecourses (where spike counts are taken in small, non-overlapping time bins), or trial-averaged timecourses (PSTHs). For each class of population activity, DataHigh can extract the corresponding latent variables, termed *neural states*, single-trial *neural trajectories*, and

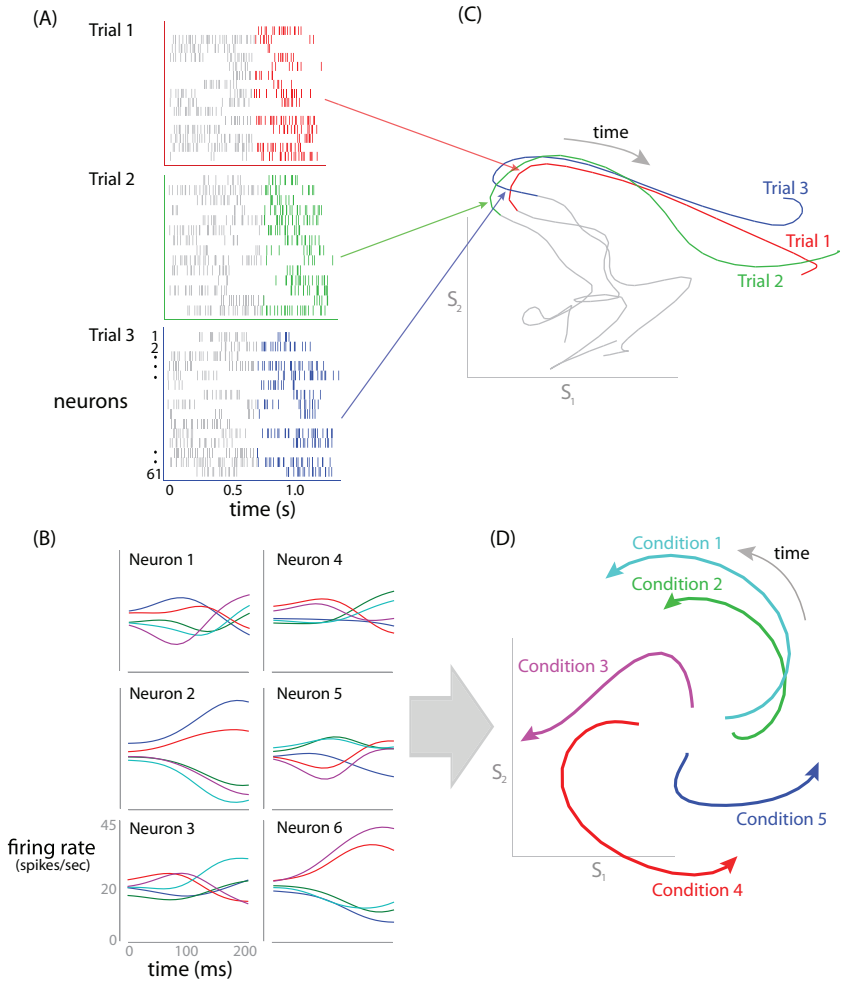

Figure 1. Conceptual illustration of applying dimensionality reduction to neural population activity. (A) Comparing population activity across repeated trials of the same experimental condition. Each raster plot corresponds to an individual experimental trial. (B) Comparing trial-averaged population activity across different experimental conditions. The peri-stimulus time histograms (PSTHs) of six neurons with five different experimental conditions are shown. (C) A dimensionality reduction method (GPFA) was applied to single-trial population activity (three trials are shown in panel (A)) to extract 15-d single-trial neural trajectories. Each trajectory corresponds to a different experimental trial. S_1 and S_2 define a 2-d projection of the extracted 15-d latent space. (D) A dimensionality reduction method (PCA) was applied to trial-averaged population activity (five experimental conditions are shown in panel (B)) to extract 6-d trial-averaged neural trajectories. Each trajectory corresponds to a different experimental condition. S_1 and S_2 define a 2-d projection of the extracted 6-d latent space.

trial-averaged neural trajectories, respectively. We previously presented a preliminary version of this work in Cowley *et al* (2012). Section 2 describes how to use DataHigh for neural data analysis and the tools available in DataHigh. We then apply DataHigh to experimental data in section 3, and compare DataHigh to a general-purpose visualization tool in section 4.

2. DataHigh

In section 2.1, we first outline a step-by-step procedure to illustrate how DataHigh can be used for exploratory neural data analysis. We then lay out the mathematical formulation for finding 2-d projections in DataHigh (section 2.2), describe a suite of neural data analysis tools included in DataHigh (section 2.3), and specify how the data should be preprocessed and formatted for input into DataHigh (section 2.4). In section 2.5, we introduce a dimensionality-reduction tool that takes, as input, raw neural spike trains, computes neural states or neural trajectories, and uploads the results to DataHigh for visualization.

2.1. Data analysis procedure

We describe and motivate a neural data analysis procedure that incorporates DataHigh and involves four steps: pre-processing, dimensionality reduction, visualization, and testing (figure 3).

In the pre-processing step, the user first inputs single-trial raw spike trains into *DimReduce*, a plug-in tool of DataHigh (left-hand side of figure 3). DimReduce first takes spike counts in non-overlapping time bins, where the bin width is specified by the user in the GUI and determines if the neural activity will be represented as neural states or single-trial neural trajectories (section 2.4). For single-trial neural trajectories, DimReduce presents two optional actions: to average across trials (for trial-averaged neural trajectories) and to apply kernel smoothing.

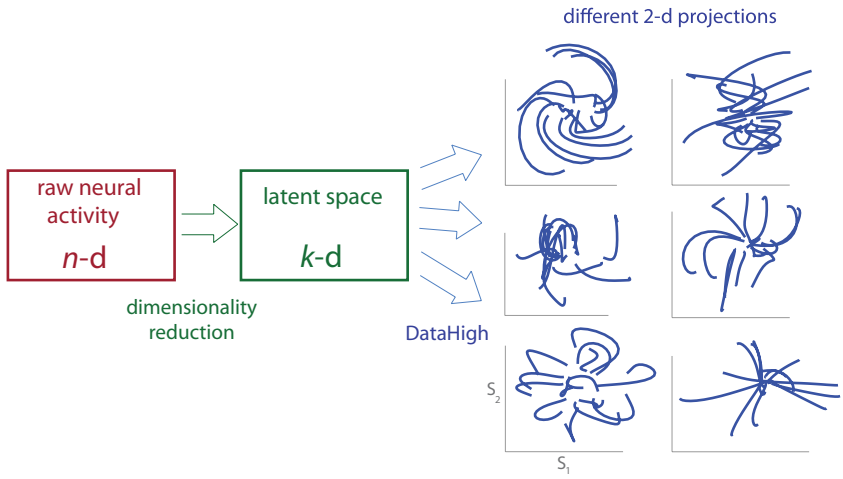

Figure 2. Flow diagram for visualization of population activity. Dimensionality reduction is performed on high-dimensional population activity (n-d, where n is the number of neurons) to extract a latent space (k-d, where k is the number of latent variables). Typically, k is less than n but greater than 3. We can then use DataHigh to visualize many 2-d projections of the same latent space. Shown here are six different 2-d projections of the same 6-d ($k = 6$) latent space described in section 3.3.

In the dimensionality reduction step, the user selects a dimensionality reduction method to extract latent variables from the neural population activity (section 2.5.1). After DimReduce has either performed cross-validation or extracted a large number of latent variables, the user can choose a latent dimensionality for visualization (section 2.5.2). These latent variables are automatically uploaded to DataHigh for visualization. Alternatively, dimensionality reduction can be performed outside of the DataHigh environment (right-hand side of figure 3). The first steps are to take binned spike counts and to choose whether to average across trials and whether to apply kernel smoothing. Then, a dimensionality reduction method that has not yet been implemented in DataHigh can be applied to the data. Finally, the extracted latent variables need to be formatted correctly and input into DataHigh (section 2.4).

The next step in the data analysis procedure is to visualize the extracted neural states and trajectories, whose dimensionality is typically greater than 3. By including visualization as a step in exploratory data analysis, the experimenter can potentially save a substantial amount of time in filtering out hypotheses about features that are not salient in the population activity and guiding the experimenter toward building hypotheses and intuition about features that are salient. This can help guide the development of algorithms and models that attempt to extract statistical structure from the population activity (section 3).

A standard way to incorporate visualization is to first hypothesize about a feature of the data and then define a cost function to search for a 2-d or 3-d projection that shows the existence of such a feature. If the hypothesized feature appears to be present, statistical tests can then be applied. However, there are two drawbacks to this approach. First, individual 2-d projections of high-dimensional data can be misleading. For instance, two points that are close together in 2-d visualization may not be close together in the high-dimensional space. Second, this 'guess-and-check' approach may require the application of many cost functions before salient features are found, and can potentially miss features that were not hypothesized. Instead of limiting visualization to a small number of projections found by cost functions, the user can interactively view many 2-d projections of the latent space with DataHigh (section 2.2), and utilize a suite of built-in analysis tools that assist in the visualization process (section 2.3). The user can then use DataHigh to visually investigate existing hypotheses while building intuition and new hypotheses.

After hypothesis-building from visualization, the user can perform statistical tests on the hypotheses. If the user has an existing hypothesis about the population activity (either from previous studies or from analyses of other datasets), the user can visually inspect whether the hypothesis holds and apply statistical tests to the data (left-hand side of *testing* in figure 3). These tests typically produce quantitative metrics (e.g., p-values, decoding accuracy), which can be complemented with visualization to add qualitative intuition about why a test succeeds or fails. However, if visualization suggests a *new* scientific hypothesis, statistical testing should be done on held-out datasets (either different datasets collected from the same subject or from different subjects), which avoids bias that comes from testing hypotheses suggested by the data (Berk *et al* 2010) (right-hand side of *testing* in figure 3). Testing can either be done in the k-d latent space or the original n-dimensional space. The advantage of testing in the latent space is that the data are 'denoised' and effects may be more pronounced. However, the user needs to carefully consider whether the dimensionality reduction method can build the effect in question into the neural states or neural trajectories, either via simulations or analytical reasoning. It is often 'safer' to use the latent space for hypothesis generation, and then test the hypothesis in the original high-dimensional space (Afshar *et al* 2011).

Another benefit of DataHigh is that it can be used to quickly view and triage the large amounts of data produced by a neuroscience experiment. With single-electrode recordings, it is common to listen to each spike as it streams in and to plot a neuron's PSTHs and tuning curve during an experiment.

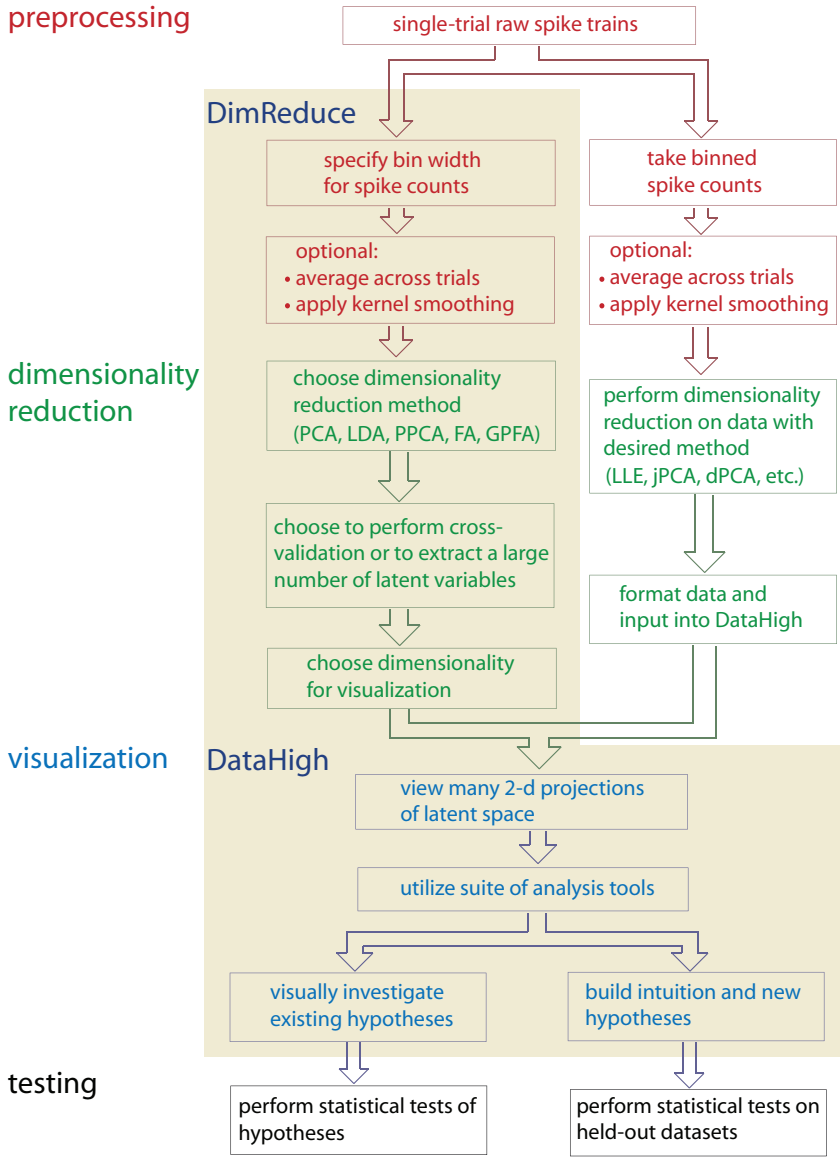

Figure 3. Flowchart for a data analysis procedure that utilizes visualization. The user may input raw spike trains into DataHigh, perform dimensionality reduction using the DimReduce tool (left-hand side of *dimensionality reduction*) and visualize many 2-d projections of the extracted latent space using DataHigh. The user may also perform dimensionality reduction outside the DataHigh environment (right-hand side of *dimensionality reduction*) and input the identified latent variables into DataHigh for visualization.

With the advent of multi-electrode recordings, it has become less common to listen to and visualize neural activity as it streams in, simply due to the sheer quantity of the neural data (e.g., if there are 100 recording electrodes). Using dimensionality reduction in tandem with visualization allows experimenters to quickly inspect a large amount of data and perform error checking between each recording session of an experiment. This can help to guide changes in the design of an experiment, to build intuition about the population activity, and to detect any potential problems with the recording apparatus. For example, this approach led experimenters to identify individual, outlying trials (Churchland *et al* 2010, Yu *et al* 2009) and electrodes that contained cross-talk (Yu *et al* 2009). As the number of sequentially- or simultaneously-recorded neurons increases, having methods for quickly building intuition about the population activity and assaying large datasets will become increasingly essential.

2.2. *Rotating a 2-d projection plane*

The main interface of DataHigh (figure 4) allows the user to quickly and smoothly rotate a 2-d projection plane in the k-d space, where k is the number of identified latent variables. The goal is to provide the minimum set of 'knobs' that allow the user to achieve all possible rotations within the k-d space. We first describe the mathematical idea of our approach, then the implementation. The mathematical details presented in this section are not necessary to use DataHigh, and may be skipped without loss of intuition for the tool. We begin with two arbitrary orthonormal k-d vectors, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, which define the horizontal and vertical axes, respectively, of a 2-d

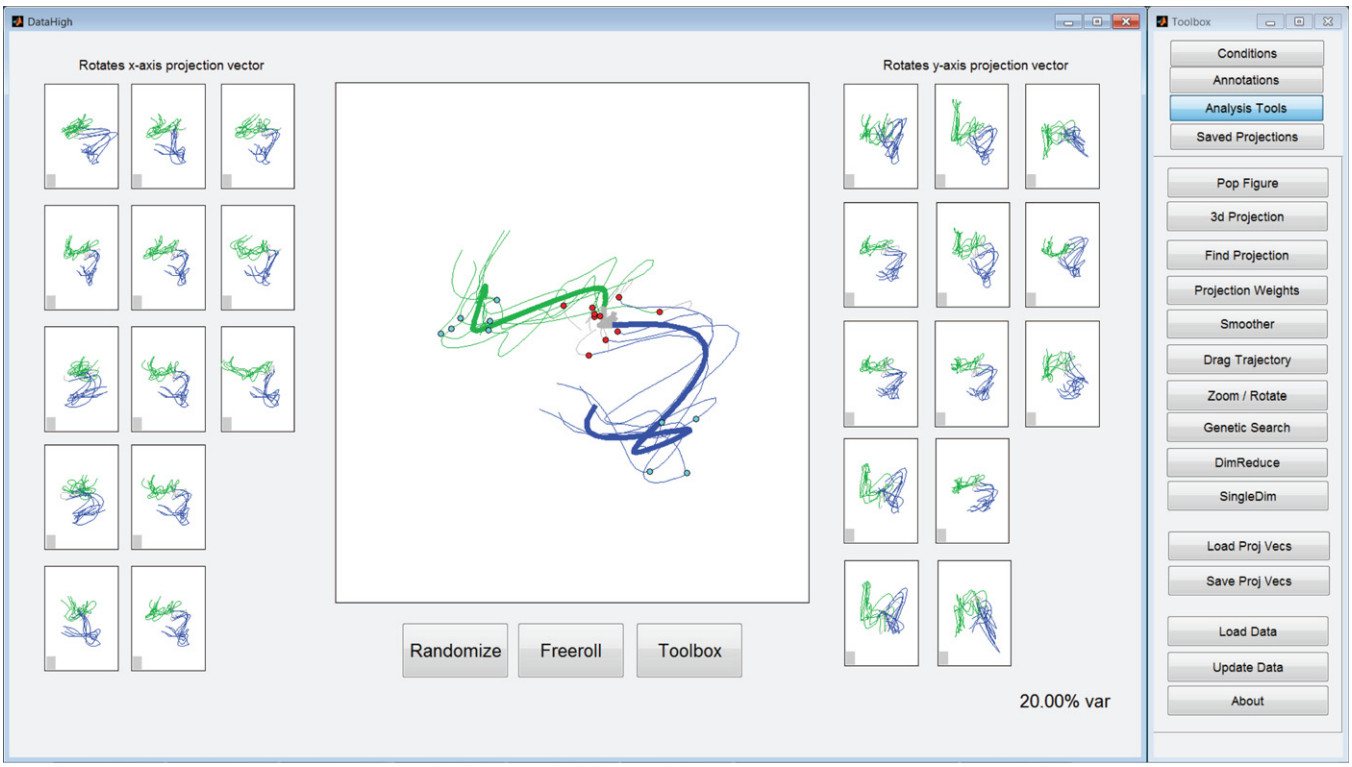

Figure 4. Main interface for DataHigh. Central panel: 2-d projection of 15-d single-trial neural trajectories extracted using GPFA from population activity recorded in premotor cortex during a standard delayed-reaching task for two different reach targets (green and blue) (section 3.2). Dots indicate time of target onset (red) and the go cue (cyan). Gray indicates baseline activity before stimulus onset. Preview panels (left and right of central panel): clicking and holding on a preview panel instantly rotates one of the two projection vectors that make up the central 2-d projection. The bottom right corner shows the per cent variance of the latent space that is captured by the central 2-d projection. The Toolbar (far right) allows the user to access analysis tools described in section 2.3.

projection plane. To rotate the projection plane, we keep one vector $\boldsymbol{v}_1$ fixed, while rotating the other vector $\boldsymbol{v}_2$. To maintain orthogonality, $\boldsymbol{v}_2$ must rotate in the $(k-1)$-d orthogonal space of $\boldsymbol{v}_1$. In this space, any rotation of $\boldsymbol{v}_2$ can be fully specified by $(k-2)$ angles. Thus, we provide the user with $(k-2)$ 'knobs' (right-hand panels in figure 4) to rotate $\boldsymbol{v}_2$ while keeping $\boldsymbol{v}_1$ fixed. Each panel shows a preview of the resulting 2-d projection if $\boldsymbol{v}_2$ were rotated by 180° in a particular rotation plane. The user can click and hold on a particular preview panel, which continuously updates the central panel as $\boldsymbol{v}_2$ is rotated smoothly in that plane. Similarly, we can fix $\boldsymbol{v}_2$ and rotate $\boldsymbol{v}_1$, which yields an additional $(k-2)$ preview panels (left-hand panels in figure 4). Thus, $2 \cdot (k-2)$ 'knobs' are the least required to choose any possible 2-d projection of the k-d space.

Explicitly, we first use the Gram–Schmidt process to find a set of $(k-1)$ orthonormal vectors spanning the orthogonal space of $\boldsymbol{v}_1$; these vectors define the columns of $Q \in \mathbb{R}^{k\times(k-1)}$. We also define a rotation matrix $R_i(\theta) \in \mathbb{R}^{(k-1)\times(k-1)}$, which rotates a $(k-1)$-d vector by an angle θ in the ith rotation plane:

$$R_i(\theta) = \begin{bmatrix} I_{i-1} & & & \\ & \cos(\theta) & -\sin(\theta) & \\ & \sin(\theta) & \cos(\theta) & \\ & & & I_{k-i-2} \end{bmatrix}, \qquad (1)$$

where I_p is a $p \times p$ identity matrix and $i = 1, \ldots, k-2$. To rotate $\boldsymbol{v}_2$ by an angle θ in the ith rotation plane, we compute

$$\boldsymbol{v}_2^{\text{new}} = QR_i(\theta)Q^{\mathrm{T}}\boldsymbol{v}_2^{\text{old}}. \qquad (2)$$

The neural trajectories shown in figure 4 are 15-dimensional ($k = 15$), leading to the use of 13 preview panels for $\boldsymbol{v}_1$ (the x-axis projection vector) and 13 preview panels for $\boldsymbol{v}_2$ (the y-axis projection vector). At present, DataHigh can support dimensionalities up to $k = 17$ (30 preview panels), which we found to be large enough for most current analyses, yet small enough to have all preview panels displayed simultaneously on a standard monitor. For $k > 17$, DataHigh applies PCA to the neural states or neural trajectories and retains the top 17 PCA dimensions for visualization. Alternatively, the user may implement a larger number of preview panels in DataHigh.

2.3. DataHigh analysis tools

In addition to the continuous rotation of a 2-d projection plane, DataHigh offers a suite of additional analysis tools that are useful for exploratory data analysis. We provide motivation and a description for each tool here, and list implementation details in appendix B.

- *Find projection.* The user might want to compare the 2-d projections found in DataHigh with static 2-d projections found by PCA, LDA, or cluster PCA (figure 5). A random projection can also be chosen. Find projection smoothly rotates the current 2-d projection to the desired static 2-d projection. Cluster PCA is a variant of PCA whereby the datapoints (either neural states or neural trajectories) are

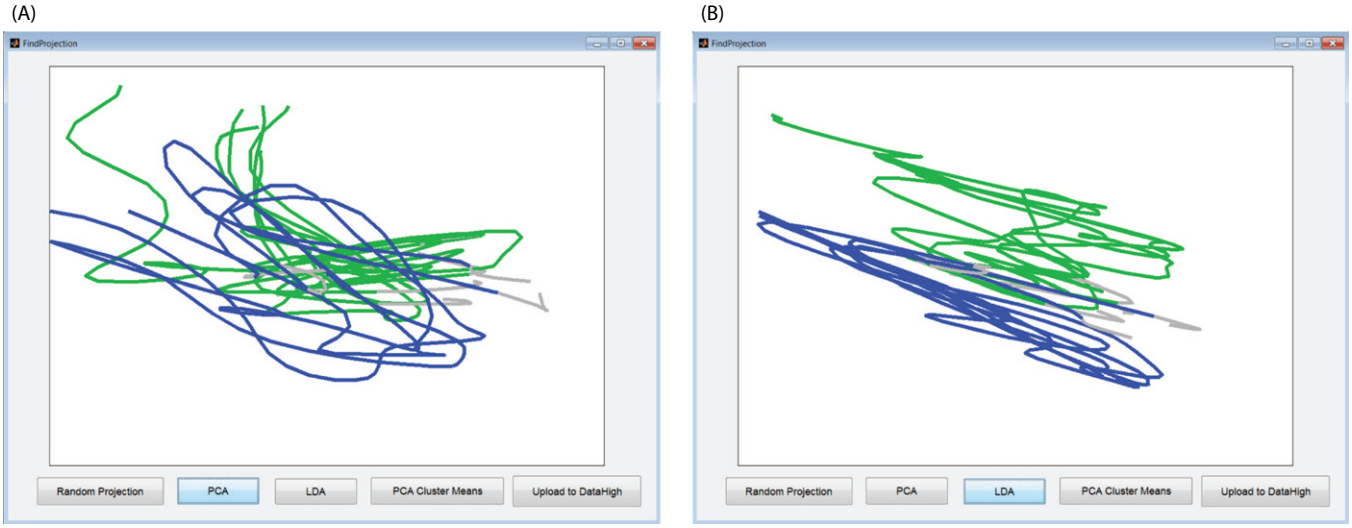

Figure 5. Find Projection tool. Static 2-d projections found by (A) PCA and (B) LDA, when applied to the data shown in figure 4. The colors (green and blue) denote different reach targets, and for each reach target, five neural trajectories are shown.

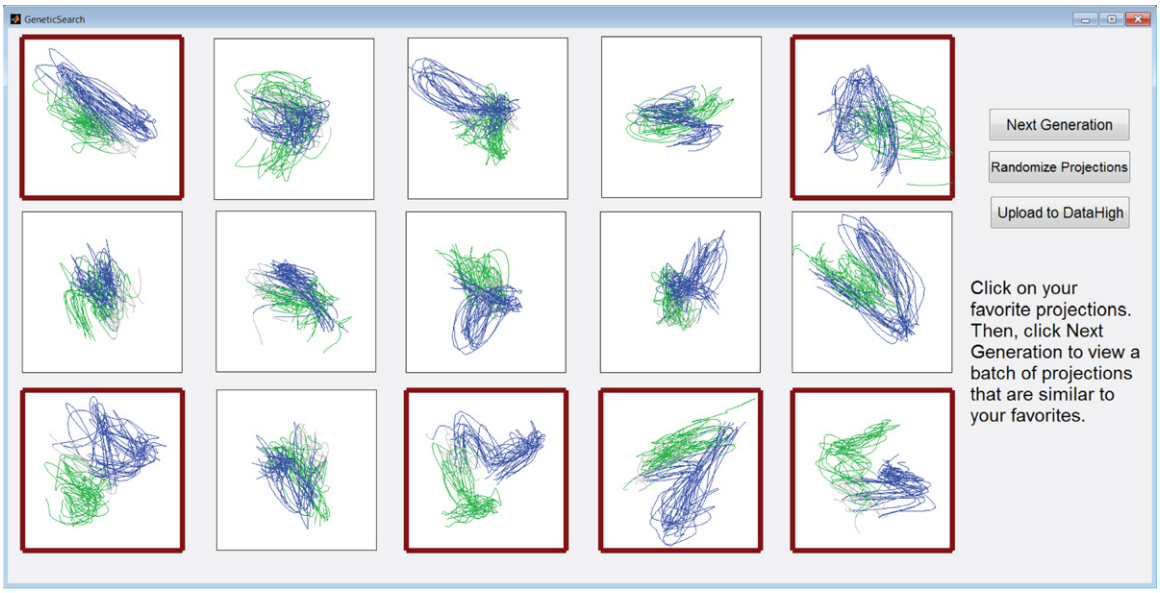

Figure 6. Genetic search tool. The user selects projections of interest (highlighted in red) and the tool transforms the current projections to be more similar to the selected ones for the next generation. Each panel displays a 2-d projection of single-trial neural trajectories for two reach targets (green and blue). These are all projections of the same 15-d latent space shown in figure 4.

first averaged within each condition, then PCA is applied to the condition means. This yields a 2-d projection which separates one condition from another, while ignoring trial-to-trial variability.

- *Genetic search.* Although DataHigh can in principle achieve any 2-d projection of the k-d space, it does so using a series of 'local' rotations. In other words, the preview panels in the main DataHigh interface show only local perturbations of the 2-d projection plane (figure 4). Thus, we created the Genetic search tool, which allows the user to first see a more global view of the population activity and then to home in on features of interest (figure 6). This global view is provided in the form of 15 projections that are randomly sampled from the space of all possible 2-d projections. The user can then select the most 'interesting' projections, and a new iteration occurs in which 15 new 2-d projections are generated to be similar to the selected projections. The search continues until the user decides to upload a projection to the main display. This tool was inspired by genetic algorithms, where the key innovation is that the user controls each evolution step.
- *3-d projection.* Rotating a 2-d projection plane in the latent space can sometimes be difficult to intuit because a small rotation may lead to considerably different 2-d projections. Instead, the user can rotate a 3-d projection in Matlab's built-in 3-d viewer, where rotation may be more intuitive. The 3-d Projection tool uses the current 2-d projection, along with a randomly-chosen orthonormal third projection vector, to allow visualization in Matlab's 3-d viewer (figure 7). However, switching

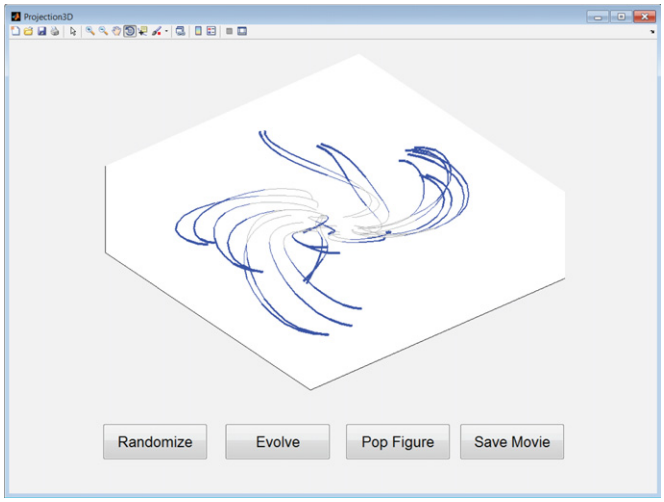

Figure 7. A 3-d projection of trial-averaged neural trajectories from section 3.3 during reaching movements. Each neural trajectory corresponds to one reach condition.

the third projection vector to a different randomly-chosen projection vector can give noticeably different 3-d projections, which underscores the need to look at many projections of the latent space. Like the Pop figure tool in section 2.3, the 3-d figure may be opened in a new Matlab figure for further editing.

- *Evolve.* Different neural trajectories may sometimes take a similar stereotyped path in latent space, but the paths are traced out at different speeds. For example, it is possible for a motor plan or decision to form more quickly on some trials than others. To highlight the timing differences between neural trajectories, we implemented the Evolve tool to display a movie of the neural trajectories playing out together over time. The movie may also be saved for external viewing.
- *Single dimension.* One might want to view each of the k latent variables separately to see how the population activity varies in each latent dimension. For neural states, Single dimension computes a histogram of latent values for each condition and each latent variable. Each of the k plots shows an overlay of all of the conditions' histograms for the corresponding latent variable. For neural trajectories, each latent variable is plotted versus time (figure 8). The number of plots k equals the number of latent variables.
- *Depth perception.* Distances in a 2-d projection may be misleading because two points that are far away in the k-d latent space may appear close together in a 2-d projection. To help intuit the shape of a cluster, infer distances between clusters, and identify outlying points, Depth perception orders neural states by their projected values onto a vector that points in the direction of greatest variance in the current projection vectors' orthogonal space. Datapoints that have a similar projected value will be similar in size and color (figure 9). This tool is not available for neural trajectories.
- *Freeroll.* To passively visualize the latent space, one can watch the main 2-d projection display while the projection plane randomly rotates in the latent space. Akin to a drifting screensaver, Freeroll continuously rotates the 2-d projection plane in a random fashion without user intervention. The tool repeatedly 'clicks and holds' on a random preview panel (the preview panels flank the main display in figure 4) for a random period of time.
- *Conditions.* For datasets with many experimental conditions, the user might want to first examine the neural states or trajectories of a small number of experimental conditions before viewing all conditions at once. The Conditions tool allows the user to selectively display any subset of the experimental conditions in the dataset. All subsequent analyses within DataHigh use only the selected conditions. The user can choose to recenter the data based on the updated set of experimental conditions.
- *Update colors.* For visual reference, the user can color-code neural states and segments of neural trajectories based on experimental condition and epoch. The Update colors tool allows the user to change color attributes, which are initially supplied by the user as part of the input structure (section 2.4). The specified color for each condition's corresponding neural states may be changed. For each neural trajectory, the start index and color may be changed for any experimental epoch (e.g., baseline period, stimulus period, delay period, etc.). The resulting changes may be saved and updated in DataHigh.
- *Capture.* It is often useful to save particular 2-d projections and keep them on hand for later viewing. The Capture tool adds the current 2-d projection to a queue of saved projections, which are shown as thumbnail images (figure 10(A)). A projection can be uploaded to the central panel simply by clicking on its thumbnail image.
- *Drag trajectory.* The user might want to understand how each latent variable contributes to a neural trajectory by altering the latent variable's timecourse and visualizing the changes in the latent space. Drag trajectory plots each of the k latent variables of a neural trajectory versus time (figure 10(B)). The user can 'perturb' the trajectory by dragging the latent-variable-versus-time curves and see its effect in the current 2-d projection. The perturbed trajectory can then be sent to the main projection display (figure 4) for further inspection.
- *Zoom/rotate* allows the user to scale the main 2-d projection axes to zoom in or out, change the rotation speed, and rotate the current 2-d projection within its projection plane around the center of the projection.
- *Pop figure* replicates the main projection with a pop-up Matlab figure. The user may proceed to edit the figure as a regular Matlab plot, adding axis labels, a title, and annotations. The figure can then be saved as `.fig`, `.jpg`, `.eps`, and `.pdf`.
- *Weights* displays how each latent variable contributes to the main 2-d projection. The Weights tool plots the elements of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ as bar graphs. The projection vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ can be saved for use outside the DataHigh environment. The user can also manually specify the values of $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in a `.mat` file, and load the projection vectors into DataHigh.

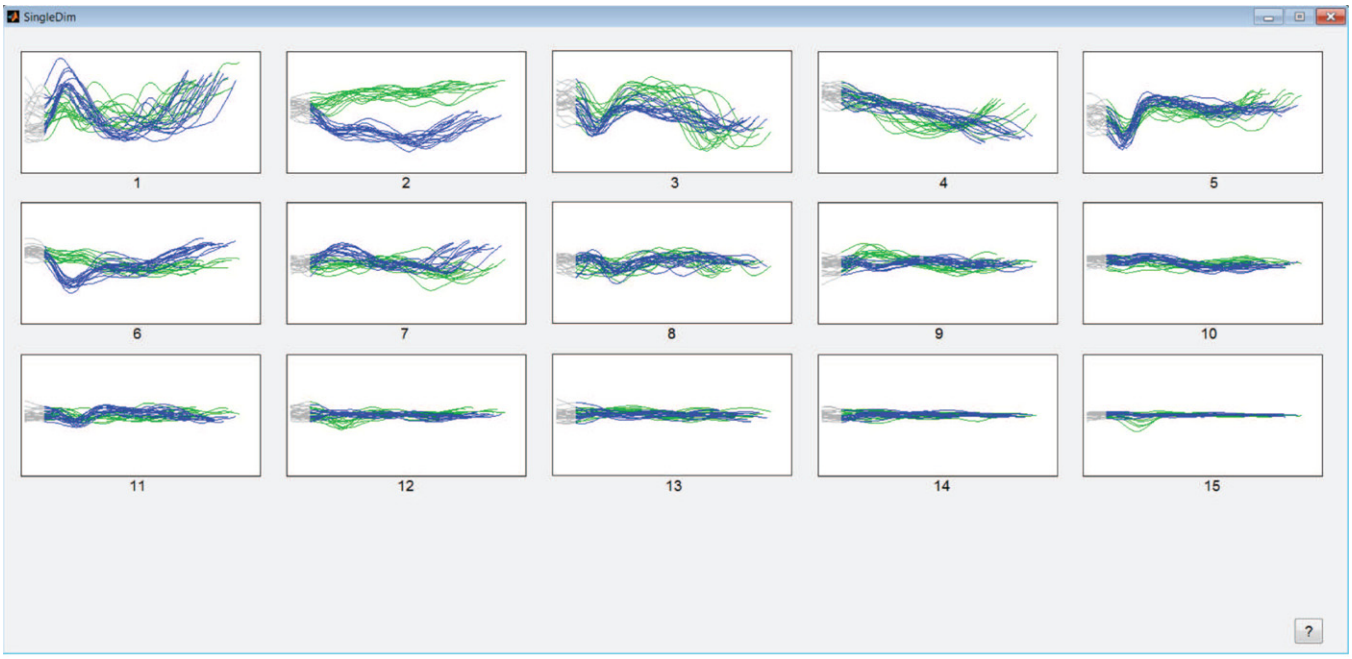

Figure 8. Single dimension tool. Each panel shows one of the 15 latent variables versus time for the neural trajectories shown in figure 4. In this case, the latent variables are ordered such that the first latent variable (panel '1') captures the greatest covariability while the the last latent variable (panel '15') captures the least covariability in the population activity.

Figure 9. Depth perception tool. The size of each point corresponds to how close that point is to the user in the projection plane's orthogonal space. The green and blue neural states correspond to two different reach targets, as described in section 3.1. The red neural states are error trials that were incorrectly classified by a Poisson Naïve Bayes classifier.

- Additional annotations for neural states: for each experimental condition, display cluster mean, covariance ellipse, origin of latent space, and first principal component direction (figure 11). The cluster mean and covariance ellipse summarize the location and scatter of neural states. The origin of the latent spaces provides a visual reference point, which is helpful for orienting the user during rotation of the 2-d projection plane. DataHigh can also display a line that indicates the direction of greatest trial-to-trial variability in the

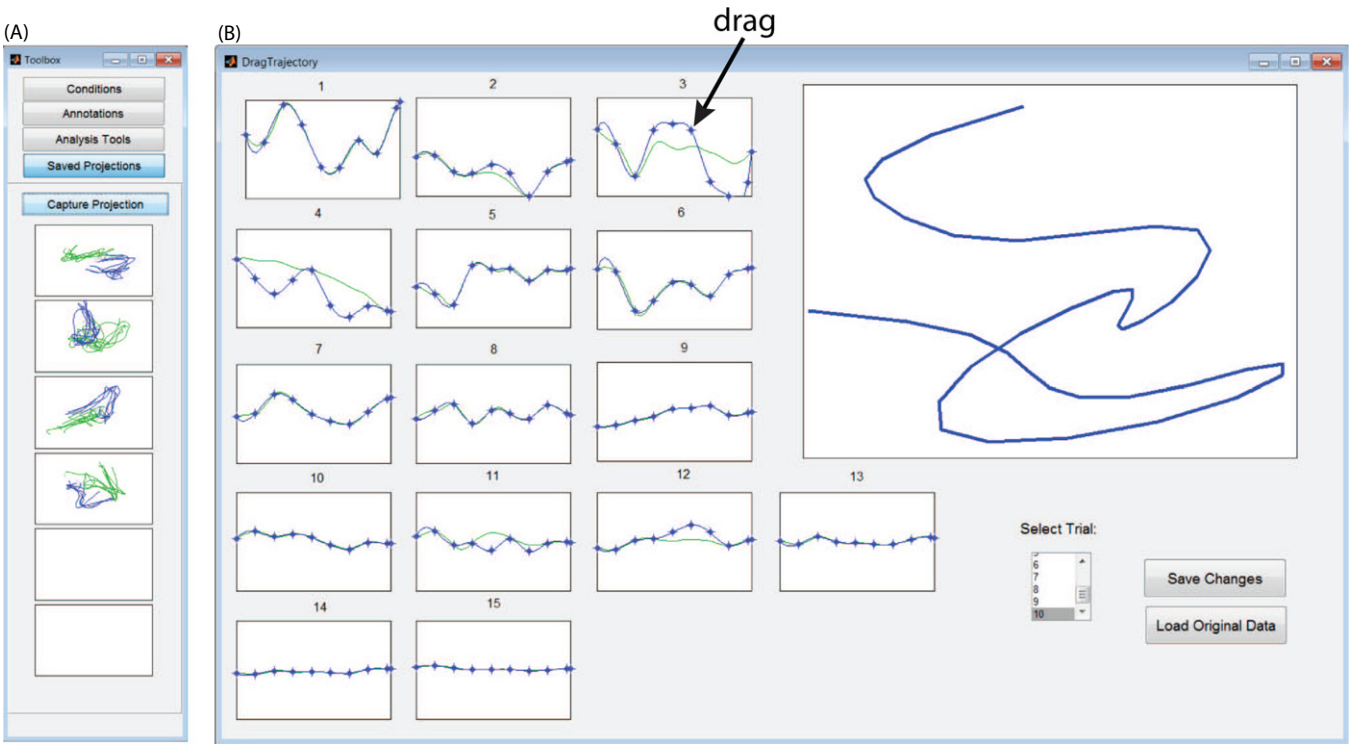

Figure 10. (A) The capture tool saves selected projections in a queue. A saved projection can be loaded by clicking on its thumbnail. (B) Drag trajectory allows the user to manipulate a neural trajectory by dragging one of the points placed along the trajectory. Each panel plots one of the latent variables versus time (left panels). The result is immediately updated in the 2-d projection display (right panel).

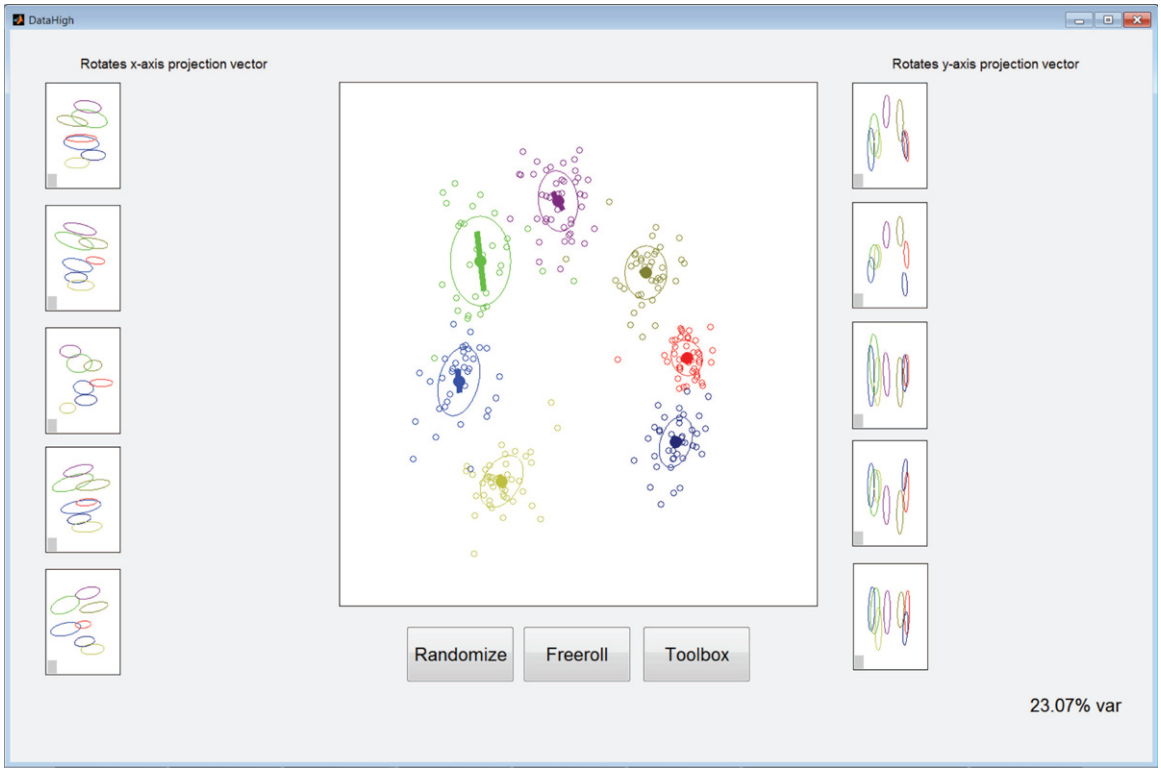

Figure 11. Trial-to-trial variability of neural states during reach planning. Each point corresponds to one trial and is colored according to reach target. For each cluster, the user can choose to display the one-standard-deviation ellipse, cluster mean, and/or the direction of greatest trial-to-trial variability (thick lines). The data corresponds to the example in section 3.1.

k-d space, and how well that direction aligns with the current 2-d projection plane. The length and direction of this line dynamically updates as the user rotates the 2-d projection plane, and the user can compare how the lines differ in length and direction between experimental conditions.

- Additional annotations for neural trajectories: display average trajectories for each experimental condition and plot task epoch boundaries with colored dots (figure 4). An average trajectory provides a visual reference when examining many single-trial neural trajectories. Epoch boundaries provide reference points when neural trajectories transition between epochs in an experiment (e.g., a task cue).

2.4. Data preprocessing and format

The first pre-processing step is to choose the bin width in DimReduce, which takes spike counts starting at the first timepoint of every trial. For neural states, the user should choose the bin width to be the length of the shortest trial, which is the maximum bin width allowed in DimReduce. If the user wants to analyze the neural states for a particular experimental epoch of each trial (e.g., stimulus period, delay period, or movement period), each trial should be truncated to include only timepoints that are within that epoch before the trials are input into DimReduce. For single-trial and trial-averaged neural trajectories, choosing the bin width is a tradeoff between the desired time resolution and the ability of dimensionality reduction methods to accurately estimate underlying changes in firing rate. Using a smaller time bin gives better time resolution. However, the spike counts will tend to be smaller (in the limit, the counts will be all zeros and ones), and it can become difficult to detect co-fluctuations in firing rate (Cohen and Kohn 2011). We suggest starting off with a default bin width of 20 ms, which we found to be a good compromise between the time resolution and mean spike counts in analyses of population activity in the motor and visual cortices (Churchland *et al* 2010).

Before performing dimensionality reduction, we also suggest the following two pre-processing steps. First, the user should compute the rate of coincident spikes (with 1–2 millisecond precision) between each pair of electrodes. It is possible for different electrodes to become (partially) electrically coupled (termed *electrode cross-talk*), leading to spikes that occur at the same time on multiple electrodes (Yu *et al* 2009). A dimensionality reduction algorithm would then dedicate a latent variable to capturing this strong co-fluctuation, possibly leading to 'jagged' neural trajectories. Thus, we recommend removing all but one electrode in each group of electrodes that show an abnormally large percentage of coincident spikes. Second, we suggest removing neurons with a low mean firing rate (<1 spike per second). These neurons tend to not contribute much to the extracted neural states or trajectories, and can cause dimensionality reduction algorithms to become unstable due to the prevalence of zero spike counts.

The input format depends on whether the user wants to input raw spike trains into DimReduce or extracted latent variables into DataHigh (which correspond to the left-hand and right-hand sides of *pre-processing* in figure 3, respectively). The input structure is a Matlab `struct`, `D`, and was designed for simple formatting while maintaining a high level of flexibility. For input into the dimensionality reduction tool DimReduce, the simplest and suggested form of `D` has a `data` field, where `D(i).data` is a matrix (number of neurons $\times$ number of milliseconds) that contains the `i`th trial's spike trains in 1 ms time bins.

For input into DataHigh, the simplest form of `D` has two fields, `data` and `type`. The `type` field takes a string and denotes which type of data: `'state'` or `'traj'`. For neural states, each `D(i)` specifies the `i`th condition, and `D(i).data` is a matrix of size $k \times N$, where k is the dimensionality of the latent space and N is the number of trials for that condition. For neural trajectories, `D(i)` specifies the `i`th trajectory, and `D(i).data` is a $k \times T$ matrix specifying a k-dimensional trajectory that is T timesteps long. Optional fields may also be included for neural states and trajectories to color-code condition and epoch information. Details about optional fields are listed in appendix A, and examples are included in the DataHigh software package.

2.5. DimReduce

To facilitate the application of dimensionality reduction methods to neural data, we developed an optional dimensionality reduction tool to DataHigh, called DimReduce (figure 12). The user inputs raw spike trains, and DimReduce extracts the corresponding latent variables, which are automatically uploaded to DataHigh (left-hand side of *dimensionality reduction* in figure 3). DimReduce gives step-by-step instructions, embedded in the GUI with large red numbers and the 'Next Step' button (figure 12), to perform dimensionality reduction, and guides the user to choose parameters that specify how to pre-process the data (e.g., time bin width, mean firing rate threshold, and smoothing kernel width), to select a dimensionality reduction method, and to specify a set of candidate dimensionalities for cross-validation. To identify the optimal number of latent dimensions, DimReduce computes a cross-validated data likelihood and a cross-validated leave-neuron-out prediction error (Yu *et al* 2009) for each user-specified candidate dimensionality. DimReduce plots these cross-validated metrics versus latent dimensionality, and the user can inspect these plots to find the latent dimensionality which maximizes the cross-validated data likelihood or minimizes the leave-neuron-out prediction error. Using a slider in the GUI, the user can then specify the number of latent variables DimReduce should extract and upload to DataHigh. To give the user a preview of the extracted latent variables, the tool displays a random 2-d projection of the latent space and a heat map of the loading matrix, which describes how each latent variable contributes to each neuron's activity. The tool also includes a viewer that plots the timecourse for each latent variable. The tool provides help buttons for how to choose a dimensionality reduction method and select pre-processing parameters, as well as how to interpret the cross-validated metrics.

DimReduce includes five linear dimensionality reduction methods that are computationally fast and have been fruitfully applied to analyze neural population activity. The five methods included are PCA, Fisher's linear discriminant analysis (LDA), probabilistic PCA (PPCA), FA, and GPFA (Bishop 2006, Yu *et al* 2009). If the user would like to use a dimensionality

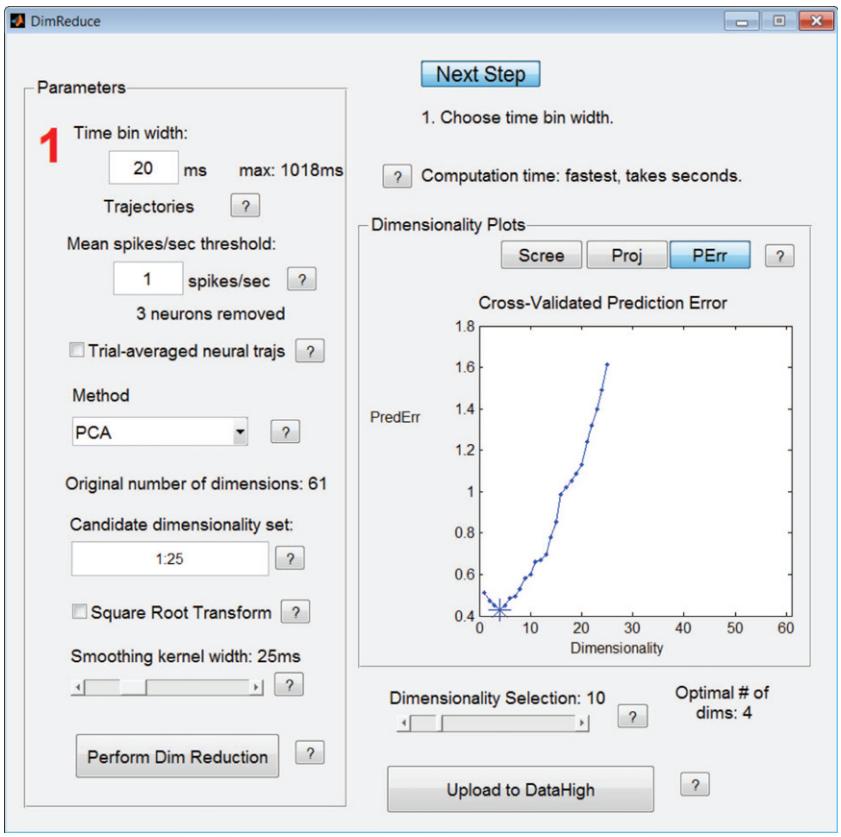

Figure 12. DimReduce allows the user to input raw spike trains, perform dimensionality reduction, choose the latent dimensionality, and upload the extracted latent variables to DataHigh. The large red '1' instructs the user where to complete the first step, which is to choose a bin width. Clicking the 'Next Step' button increments the red step number and moves it to the next step. The example here shows a plot of leave-neuron-out prediction error versus candidate latent dimensionality. Using this metric, the optimal latent dimensionality is the dimensionality with the minimum cross-validated prediction error (starred on the plot).

reduction method that is not currently available in DimReduce, the user may extract the latent variables outside the DataHigh environment and input them directly into DataHigh for visualization (right-hand side of *dimensionality reduction* in figure 3). For example, latent variables can be extracted by locally-linear embedding (Stopfer *et al* 2003), jPCA (Churchland *et al* 2012), demixed PCA (dPCA) (Brendel *et al* 2011), and linear dynamical systems (Yu *et al* 2009, Macke *et al* 2011). These methods may be added to future releases of DataHigh.

2.5.1. Selecting a dimensionality reduction method. The choice of dimensionality reduction method depends on the scientific questions of interest and the properties of the population activity to be visualized. Key considerations include whether the neurons were sequentially or simultaneously recorded, whether the user is interested in comparing single-trial or trial-averaged population activity, and whether the user is interested in the timecourse of the population activity. For each class of neural data, we shall recommend a corresponding dimensionality reduction method that we believe is appropriate for visualization purposes.

If the user is interested in trial-to-trial variability, where there is a single spike count vector (i.e., a point in multi-dimensional firing rate space) for each trial, we suggest FA. FA attempts to remove the independent Poisson-like spiking variability to identify a set of shared factors that describes the co-fluctuations of the activity across the neural population. FA has advantages over PCA, which assumes no observation noise and is thus less effective at removing Poisson-like spiking variability, and PPCA, which assumes that each neuron has the same observation noise variance. In contrast, FA allows neurons with different mean firing rates to have different levels of observation noise, which better agrees with Poisson-like spiking variability that depends on mean firing rate.

If the user is interested in single-trial neural trajectories, we recommend using GPFA (Yu *et al* 2009). GPFA extends FA to include temporal smoothing of the latent variables, where the level of smoothness is determined by the data. Since the publication of Yu *et al* (2009), we have accelerated the running time of the GPFA Matlab code by two orders of magnitude for the same hardware. For example, applying GPFA (i.e., running the EM algorithm once to convergence) to 100 neurons and 200 experimental trials now takes less than a minute on a standard single-processor laptop computer.

If the user is interested in comparing trial-averaged population activity across different experimental conditions, we suggest using PCA, which identifies latent variables (i.e., principal components) that correspond to axes of greatest variance. In contrast to PPCA and FA, PCA does not have an explicit observation noise model to help remove the Poisson-like spiking variability. The use of PCA is well-justified here

because trial-averaging (to produce the PSTHs that are input to PCA) likely removes much of the Poisson-like spiking variability, especially as the number of trials that are averaged together increases. If averaging over a small number of trials, the user can choose to apply kernel smoothing to the PSTHs.

2.5.2. Choosing the number of latent variables for visualization. One typically identifies the optimal latent dimensionality from the data by performing cross-validation on a range of candidate dimensionalities. Cross-validation can be time-consuming because each candidate dimensionality needs a separate set of model parameters to be fit for each cross-validation fold. For visualization purposes, finding the optimal dimensionality is not crucial, and choosing a number of latent variables that capture most of the richness in the data usually suffices. To do this, we suggest first performing dimensionality reduction with a large number of latent variables (e.g., a 50-d latent space for 100 neurons) without cross-validation. Then, choose a small subset of top latent variables that explain most of the variability. For PCA and PPCA, common methods for choosing this subset involve viewing the eigenspectrum of the sample covariance matrix and looking for a 'bend at the elbow,' or choosing a dimensionality that explains greater than 90% of the variance. Likewise for FA and GPFA, one may view the eigenspectrum of the shared covariance matrix and again look for an elbow. The shared covariance matrix is defined by CC^{T}, where C (number of neurons $\times$ number of latent variables) is the loading matrix. A complementary view is provided by plotting the timecourse separately for each orthonormalized latent variable (Yu *et al* 2009). Orthonormalization has two key benefits: (i) it provides an ordering of the latent variables from greatest to least covariance explained (similar to PCA), and (ii) the units of the latent variables will be the same as the units of the observed variables (in this case, spike counts). After orthonormalization, the user can focus on visualizing the top dimensions. As one proceeds to the lower dimensions, one typically sees less variability in the latent variables over time, across trials, or across conditions (figure 8). The user can estimate visually how many of the top dimensions are needed to capture the interesting structure in the population activity.

3. Data analysis examples

We demonstrate here three examples of the utility of DataHigh for exploratory neural data analysis. The datasets discussed in this section are included in the DataHigh software package.

3.1. Trial-to-trial variability of neural states

One possible use of DataHigh is to examine how the population activity binned within a fixed time window varies from trial-to-trial. Visualizing the size and shape of trial-to-trial variability has important links to the study of spike count correlations (Averbeck *et al* 2006, Cohen and Kohn 2011), and can motivate the development of a more accurate brain-computer interface (BCI) classifier (Santhanam *et al* 2009). The specific example here comes from multi-electrode array recordings in premotor cortex while a monkey is planning to reach to different visual targets. Previous studies have shown that the population activity in premotor cortex reflects the identity of the upcoming reach (Santhanam *et al* 2006). We first took spike counts across 61 neurons in a 400 ms window during the plan period of a delayed reaching task (dataset G20040123) (Yu *et al* 2009). We then applied FA to extract a 7-d neural state of each trial using DimReduce. The latent dimensionality seven was obtained by finding the peak of the cross-validated log-likelihood plot.

We then used DataHigh to visualize the 7-d latent space (figure 11). The most salient feature is the clustering of the neural states according to reach target. In other words, the neural states on repeated trials for the same reach target tend to be closer together than neural states on trials for different reach targets. Furthermore, the clusters appear in the same order around a circle as the corresponding reach targets appear in the monkey's workspace. This is consistent with the finding that tuning curves tend to vary smoothly with reach direction. We can also study the shape of the clusters, as well as the directions of greatest trial-to-trial variability. For example, if there were one dominant direction of trial-to-trial variability, then the cluster would look like a 'pencil.' We can then ask what direction the 'pencil' is pointing in, and whether that helps or hurts discriminability among the reach targets. For each cluster, the thick line represents the 2-d projection of a unit vector pointing in the direction of greatest trial-to-trial variability *in the 7-d latent space*. The length of the line indicates the extent to which the direction of greatest variability (i.e., the 'pencil') in the 7-d space aligns with the 2-d projection plane. Note that the lines need not align with the direction of greatest variability as seen in the 2-d projection (i.e., each line need not align with the major axis of the corresponding ellipse). For this example, we found considerable trial-to-trial variability for each cluster orthogonal to the projection shown in figure 11. This underscores the danger of drawing conclusions from any individual 2-d projection, as well as the need for looking at many 2-d projections using DataHigh.

To assess the discriminability of different targets based on the neural activity recorded on a single-trial basis, we applied a Poisson Naïve Bayes classifier to the 61-d spike counts. Consistent with the clear separation of the clusters shown in figure 11, we found that 98% of the trials were correctly classified (chance level: 14%). A key benefit of DataHigh is that it allows us to go beyond a simple per cent correct value, and visualize in what regions of the corresponding latent space the classifier succeeds and fails. Figure 9 shows a 2-d projection of the 7-d neural states described above for two of the seven reach targets. Correctly-classified neural states are colored based on their corresponding reach target (green or blue). Incorrectly-classified neural states are colored red. This dataset had highly-separable clusters of neural states, and the classifier incorrectly classified only a few outlying datapoints. By visualizing how error trials (i.e., red dots) differ from correctly-classified trials, we can gain insight about how to improve the classifier to increase classification accuracy. Likewise, this approach can be applied to behavioral tasks to understand how the population activity differs on trials with successful behavior versus unsuccessful behavior.

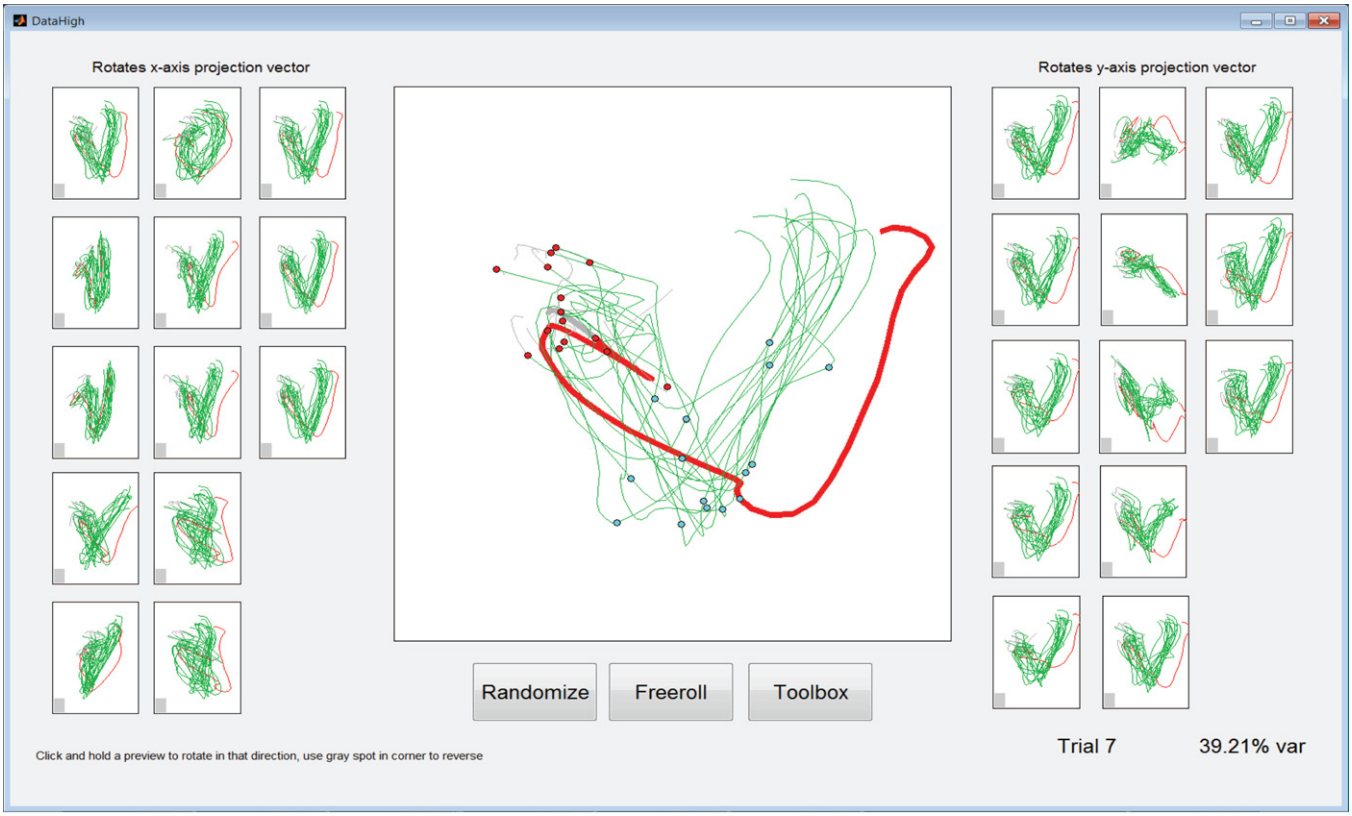

Figure 13. An outlying trial (red trace) can be easily identified by rotating the 2-d projection plane. Green trajectories represent single-trial population activity for a single reach target. A subset of these trials is shown in figure 4.

3.2. Single-trial neural trajectories

Another use for DataHigh is to examine how the population activity evolves over time within a single trial, as well as compare how the timecourse of population activity varies across trials. The benefits of visualization from section 3.1 also hold here; the key difference is that we now consider timecourses of neural activity. Instead of comparing raster plots across trials to understand how one trial differs from another (figure 1), we can apply dimensionality reduction and compare the extracted neural trajectories. With the same 61 simultaneously-recorded neurons in section 3.1 and a 20 ms bin width, we used GPFA to extract 56 single-trial neural trajectories for each of two reach conditions (five trials for each condition are shown in figure 4, dataset G20040123). Our previous work has shown that neural activity goes from a baseline region along a movement planning path, then along a movement execution path (Yu *et al* 2009). Furthermore, we expect trajectories for different reach targets to take on different paths based on our ability to decode the reach target from population activity, as described in section 3.1 (Santhanam *et al* 2006). If we use PCA or LDA to visualize the trajectories, they look like 'spaghetti' and are hard to interpret (figure 5). Using DataHigh's analysis tools (such as Freeroll, Find projection, and Genetic search), we can quickly search a large number of 2-d projections to find projections in which the trajectories splay out over time and are grouped by reach target (figures 4 and 6). We can study the size and direction of trial-to-trial variability and how the trial-to-trial variability changes over time (Churchland *et al* 2010). We also find DataHigh to be useful for identifying outlying trials and studying their relationship with the non-outlying trials (figure 13). DataHigh's interface allows the user to highlight any trajectory by clicking on it, change the color of the trajectory on the fly, and save this change to the Matlab data structure.

3.3. Trial-averaged neural trajectories

We can use DataHigh to analyze population activity in which the neurons are recorded sequentially using single electrodes, and ask how the timecourse of the trial-averaged population activity varies across different experimental conditions. The particular example we will consider here is 118 neurons recorded sequentially during a delayed reaching task in which there are 27 different reach conditions (monkey N) (Churchland *et al* 2012). One way to analyze this activity is to view 118 plots, each with 27 overlayed PSTHs. Because of the large number of plots and the temporal complexity and heterogeneity of the PSTHs (Churchland *et al* 2012), it is difficult to succinctly summarize the richness of the population activity simply by viewing the PSTHs. Instead, we can apply dimensionality reduction (in this case, PCA with $k = 6$ latent dimensions) using DimReduce and visualize the latent variables in DataHigh (figure 7). We analyzed neural activity for 200 ms starting 50 ms after the go cue. By viewing many 2-d projections, we found two salient features. First, the trial-averaged trajectories start off at different

locations before spiraling outwards. This is a consequence of preparatory activity differing for these different reach conditions (Churchland *et al* 2010). Second, there is rotational structure, and the rotation direction is consistent with where the trajectory starts (Churchland *et al* 2012). The key benefit of DataHigh here is that it allows the user to explore the rich structure of the population activity without needing to know *a priori* what structure is present. Without visualization, one would need to sequentially test whether different features are present in the population activity, which is a time-consuming process. Even if this process were carried out, salient features may be missed simply because one did not explicitly test for them.

4. Comparison to GGobi

A popular visualization tool is GGobi (Swayne *et al* 2003), to which we compare DataHigh. The most prominent difference is that DataHigh, unlike GGobi, was almost entirely developed to visualize high-dimensional time series data (i.e., how neural trajectories play out over time). DataHigh also includes a suite of visualization tools that are tailored for neural data analysis. Fundamentally, the tools are designed for different classes of data analysis problems. GGobi displays relationships between an experiment's independent and dependent variables (i.e., supervised learning), while DataHigh is tailored for studying latent variables that describe many observed variables as they co-fluctuate together (i.e., unsupervised learning). For example, while GGobi finds individual relations between olive harvesting locations and protein levels in corresponding olive oils, DataHigh displays latent variables that capture the activity co-fluctuations across a neural population without requiring prior assumptions about their relationship to stimulus parameters or the subject's behavior. GGobi allows the user to toggle variables on and off, but this ability is less useful in DataHigh, where the user is interested in visualizing the latent space in which all latent variables are plotted together.

The interface of DataHigh, unlike GGobi's external software platform that requires XML-formatted input, is completely incorporated into the Matlab environment. Matlab was chosen for its hardware/platform independence and because it is currently the most commonly-used data analysis software in the neuroscience community. This allows users to modify DataHigh's code with Matlab's extensive library of functions and to input commonly-used data structures for raw spike trains.

One of the key advantages of DataHigh is the use of preview panels in the main DataHigh interface (figure 4). This allows the user to make an informed decision about which panel to click on and facilitates driving toward interesting projections. In contrast, GGobi provides an interface to change the coefficients of the projection vectors, but does not provide a preview of how changing the projection vectors would change the 2-d visualization (figure 14). Overall, we found it easier to navigate the k-d space using DataHigh. In addition, DataHigh provides the ability to save and load projections and to apply commonly-used dimensionality reduction methods to the data.

5. Discussion

One approach to studying the high-dimensional activity of a population of neurons is to perform dimensionality reduction. However, the number of identified latent variables needed to capture the dynamics and structure of population activity is often greater than 3 and, as a result, it is difficult to directly visualize neural states or neural trajectories in the extracted latent space. To address this limitation, we developed DataHigh, an interactive Matlab GUI that allows the user to smoothly rotate a 2-d projection plane in the latent space and quickly view a large number of 2-d projections. DataHigh is also equipped with a suite of visualization tools tailored to neural data analysis.

We found DataHigh to be particularly useful for exploratory neural data analysis, where the relationship between the activity of any individual neuron and externally-imposed (e.g., sensory stimulus) or externally-measurable (e.g., subject's behavior) quantities may be unknown. By first applying dimensionality reduction to neural population activity, we attempt to relate the activity of each neuron to the activity of other neurons (either recorded simultaneously or sequentially). These relationships are captured by the identified latent variables, which describe how the activity of the neurons covary. In principle, we would like the population activity to first 'speak for itself' via the identified latent variables. Then, we can attempt to relate the latent variables to externally-imposed or externally-measurable quantities. DataHigh facilitates the process of building intuition about how the latent variables (and, by extension, the population activity) vary over time, across trials, and across conditions. More generally, DataHigh can be used to visualize the latent variables of any model of population activity, including generalized linear models (Vidne *et al* 2012).

Visualization can be a valuable component of the data analysis procedure. For example, Bartho *et al* (2009) presented sustained auditory tones and, as a first step of analysis, visualized the corresponding trial-averaged neural trajectories to gain insight into the population activity recorded from the auditory cortex. The visualization of the neural trajectories suggested that the sustained population response differed in magnitude and direction from the transient population response directly after stimulus onset. They then formed hypotheses about what they had seen in the visualizations and conducted statistical tests to either confirm or reject the hypotheses. Visualization with DataHigh, in tandem with statistical techniques (including statistical tests, decoding algorithms, models of population activity, and simulations), provides a powerful analysis framework to build intuition and to test scientific hypotheses.

DataHigh's code is released as open source, so that users may modify and add to the suite of available tools. This may include adding more dimensionality reduction methods to DimReduce, implementing projection-finding algorithms based on some cost function, such as jPCA (Churchland *et al* 2012) or dPCA (Brendel *et al* 2011), and extending DataHigh for real-time visualization of population activity during experiments. The DataHigh

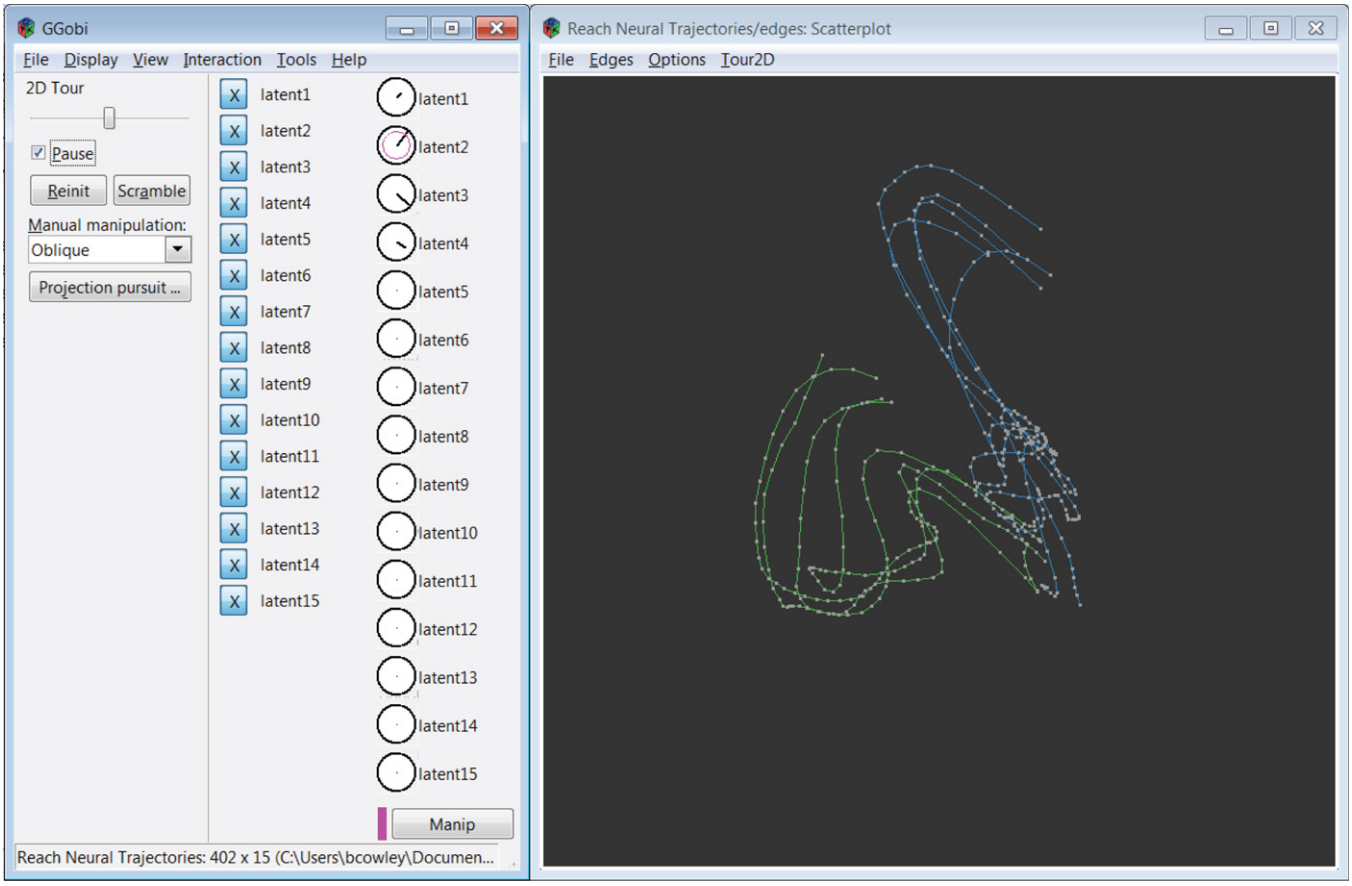

Figure 14. Screenshot of the GGobi graphical user interface while visualizing the same 15-d neural trajectories as shown in figure 4. The left panel allows the user to select which variables to display and which coefficients of the projection vectors to manipulate. The right panel displays a 2-d projection of the 15-d neural trajectories. Clicking and dragging in the right panel modifies the coefficients of the projection vectors selected in the left panel. (GGobi version 2.1.9)

software package for Matlab can be downloaded from `www.ece.cmu.edu/~byronyu/software.shtml`.

Acknowledgments

We thank Patrick Sadtler, William Bishop, and John Cunningham for valuable feedback about the DataHigh software. This work was supported by the National Science Foundation Graduate Research Fellowship Program under grant no. 0946825, Department of Defense through the National Defense Science and Engineering Graduate Fellowship Program, Center for the Neural Basis of Cognition (CNBC) and HHMI Undergraduate Research Fellowships, NIH NICHD R01-HD-071686, NIH NINDS R01-NS-054283, NIH Director's Pioneer Award (1DP1OD006409), DARPA REPAIR (N66001-10-C-2010), Burroughs Wellcome Fund, and the Christopher and Dana Reeve Foundation.

Appendix A. Data format

For input into DimReduce, there are three optional fields along with the required `data` field that contains raw single-trial spike trains. The `condition` field takes a string to label that trial's experimental condition. The `epochStarts` field is an s-dimensional vector that contains the starting time (in milliseconds) for each of the s epochs of the trial. The `epochColors` field is an $s \times 3$ matrix, where each row specifies the RGB color of the trajectory segment for the corresponding experimental epoch.

For input into DataHigh, there are two types of input (neural states and neural trajectories) that take three optional fields along with the required `data` and `type` fields. For neural states, the `condition` field takes a string to label the condition of the trials found in `data`. The `epochStarts` field takes the scalar 1, while `epochColors` is a 1×3 vector that lists the three RGB values for the condition's color. For single-trial and trial-averaged neural trajectories, the `condition`, `epochStarts`, and `epochColors` fields follow the same format as described for input into DimReduce, except that the values in `epochStarts` are indices to the starting timepoints of the epochs, not necessarily in milliseconds.

Appendix B. Implementation of analysis tools

In this section, we describe the implementation details of the DataHigh analysis tools discussed in section 2.3.

- *Find projection.* To allow for a smooth transition between projections, the current 2-d projection plane is gradually rotated until the desired static 2-d projection is achieved (Buja *et al* 2005). We first describe the intuition and then

the mathematical implementation. First, we compute the two principal angles necessary to rotate the current 2-d projection plane (defined by projection vectors $U_c : k \times 2$) to the desired projection plane (defined by projection vectors $U_d : k \times 2$) in the k-d latent space. The first principal angle is the smallest angle between two vectors (called principal vectors) in which one vector lies within the current 2-d projection plane and the other within the desired 2-d projection plane. The second principal angle follows how the first is defined except its principal vectors must be orthogonal to the first principal vectors. We aim to incrementally rotate the current projection plane until the desired projection plane is achieved, and for each incremented step, we display the 2-d projection defined by the rotated projection vectors. We ensure that at each step the rotated projection vectors are orthonormal, which is crucial for visualization. Mathematically, we first perform a singular value decomposition such that $S_c \Lambda S_d^{\mathrm{T}} = U_c^{\mathrm{T}} U_d$, where S_c and S_d are 2×2 matrices and Λ is a 2×2 diagonal matrix. Let $\gamma = \cos^{-1}(\mathtt{diag}(\Lambda))$ define the two principal angles, where γ_i refers to the ith principal angle for $i = \{1, 2\}$. We also define $P_c = U_c S_c$ and $P_d = U_d S_d$ as the principal vectors of the current and desired projection planes, respectively. Let $\boldsymbol{p}_{ci}$ refer to the ith column of P_c, which is the ith principal vector for $i = \{1, 2\}$. To ensure orthonormality between projection vectors, we perform Gram–Schmidt on P_d to yield $\check{P}_d$ such that $\check{\boldsymbol{p}}_{di}$ is orthonormal to the principal vectors in P_c and the other vector in $\check{P}_d$. Our aim is to rotate P_c until it is equal to P_d. We step through a loop, beginning at $t = 0$ and incrementally increasing t until $t = 1$. At each step, we display a 2-d projection. Let $P_t : k \times 2$ be the rotated vectors for each step, where t is the fractional scalar applied to the principal angles, $t = \{0, \frac{1}{100}, \frac{2}{100}, \ldots, \frac{99}{100}, 1\}$. For each t, compute $\boldsymbol{p}_{ti} = \cos(t\gamma_i)\boldsymbol{p}_{ci} + \sin(t\gamma_i)\check{\boldsymbol{p}}_{di}$ for $i = \{1, 2\}$. Note $P_t = P_c$ when $t = 0$ and $P_t = P_d$ when $t = 1$. The step's projection vectors, $V_t = P_t S_c^{\mathrm{T}}$, can then be computed and used to display the 2-d projection. The multiplication by S_c^{T} ensures that $V_t = U_c$ when $t = 0$, and V_t defines the same 2-d projection plane as U_d when $t = 1$.

- *Genetic search.* By clicking 'Next generation', each projection is updated using the following rule: a selected projection has a 10% chance of remaining the same, a 45% chance of rotating its projection plane by a small angle in any direction with equal probability, and a 45% chance of changing each of its projection vectors to be a random linear combination of all selected projection vectors while maintaining orthonormality. A non-selected projection has a 10% chance of re-sampling a new random projection and a 90% chance of rotating its projection plane toward a randomly-chosen selected projection, where the amount of rotation is half the angle between the two projection planes. The user may also randomize all current projections, re-starting the search.
- *Evolve.* Starting from the current timepoint, ten line segments that connect the eleven most recent timepoints are plotted with linearly diminishing line widths and make up a 'sliding trace' of the trajectory. The ten line segments are colored based on epoch colors. For reference, a thin gray line shows the remainder of each trajectory that the sliding trace has already passed. Evolve stops when all trajectories have finished playing out over time, leaving a colored line connecting the last eleven timepoints and a thin gray line for the rest of the trajectory. Alternatively, the user may pause Evolve at any point by clicking the Evolve button before the sequence stops. Execution may be resumed by re-clicking the Evolve button. The user may save the entire Evolve sequence as an `.avi`, `.mp4`, or `.mj2` video file.
- *Depth perception* projects the high-dimensional datapoints onto the first principal component of the current projection vectors' orthogonal space. The size of each datapoint reflects its principal component score, where the largest point corresponds to the datapoint that has the highest principal component score. Each point's RGB color, which corresponds to the neural state's experimental condition, is also linearly scaled based on its principal component score to control brightness. The marker of the datapoint with the highest principal component score will have the brightest color (i.e., its color will match the condition's given color, while other dots will have lesser RGB values and appear darker).
- *Freeroll* is comparable to a randomized grand tour (Asimov 1985). On each iteration, a random rotation plane (indexed by i in equation (1)) and a random angle (equivalent to clicking a preview panel without release for a random amount of time) are chosen. Freeroll proceeds to rotate the projection vectors in increments (θ in equation (1)). Once the incremental rotations sum to the chosen random angle, the next iteration occurs. Freeroll stops rotating on a user-click.
- *Conditions* instantly replots the panels and main projection display when conditions are selected or deselected. All other DataHigh analysis tools only have access to data from the selected conditions.
- *Drag trajectory.* The user selects which trajectory to analyze using a drop-down menu. The value of a latent variable can be perturbed by vertically dragging one of the points placed along the trajectory. Every fifth point along the trajectory is draggable. By dragging a point, the adjacent draggable points also move by half the amount in the direction of the dragged point. Then, the plot instantly displays a cubic spline interpolation of the draggable points to form a smooth trajectory for that latent variable. For visual reference, the original trajectory is plotted green in the background for each plot. The GUI also shows a 2-d projection of the selected trajectory (using the current projection vectors from the main projection display), which updates in real time as the trajectory is being dragged in the other panels. The user may also select other trials, upload all resulting dragged trajectories to the main DataHigh interface, and reload the original trajectory.
- *Covariance ellipses* plots the 2-d covariance ellipse for each neural state cluster. The covariance ellipse shows the

trial-to-trial variability in the 2-d projection. Specifically, the ellipse represents the projected covariance matrix $\mathbf{A} : 2 \times 2$ of the condition's covariance matrix $\Sigma : k \times k$, where k is the number of the latent variables. For the current projection vectors $\boldsymbol{v} : k \times 2$, $\mathbf{A} = \boldsymbol{v}^T \Sigma \boldsymbol{v}$. The first principal component, $\mathbf{u} : k \times 1$, of each condition's cluster is the top eigenvector of Σ and represents the direction of greatest variance. The unit vector $\mathbf{u}$ may be projected onto the 2-d projection as a line and positioned at the cluster mean. The 2-d projected principal component need not align with the 2-d covariance ellipse's major axis.

- *Average trajectories.* For visualization purposes, the single-trial neural trajectories for a given experimental condition may be averaged in the latent space. This provides a reference trajectory when examining single-trial neural trajectories. Note that this differs from trial-averaged neural trajectories, where averaging occurs before dimensionality reduction is performed. One difficulty with trial-averaging is that the duration of an experimental epoch for each trial may be different, and we want to ensure that the experimental epoch boundaries are aligned across trials for averaging. For ease of visualization, we also require a continuous average trajectory as opposed to separate trajectory segments, each aligned to an epoch boundary. Although there is not a single best method for trial-averaging (which highlights the dangers of interpreting trial-averaged quantities for non-identical trials), we opted for a method that linearly resamples the path represented by each neural trajectory in the latent space based on the distance traveled between epoch boundaries. This method ensures epoch boundary alignment, continuity, and simplicity in implementation, and works well for the population activity we have examined. The following describes how the averaging is performed. We first compute N_i, the average number of timepoints in the ith epoch. On each trial, we then resample the trajectory segment corresponding to the ith epoch by placing N_i markers. The location of each marker is determined by linearly resampling the distance along the path of the trajectory segment such that the distance along the path between two markers is the same for all pairs of consecutive markers. Finally, we average the N_i markers across trials to create the average trajectory segment for the ith epoch.

References

Afshar A, Santhanam G, Yu B M, Ryu S I, Sahani M and Shenoy K V 2011 Single-trial neural correlates of arm movement preparation *Neuron* **71** 555–64

Asimov D 1985 The grand tour: a tool for viewing multidimensional data *SIAM J. Sci. Stat. Comput.* **6** 128–43

Averbeck B B, Latham P E and Pouget A 2006 Neural correlations, population coding and computation *Nature Rev. Neurosci.* **7** 358–66

Bartho P, Curto C, Luczak A, Marguet S L and Harris K D 2009 Population coding of tone stimuli in auditory cortex: dynamic rate vector analysis *Eur. J. Neurosci.* **30** 1767–78

Berk R, Brown L and Zhao L 2010 Statistical inference after model selection *J. Quant. Criminol.* **26** 217–36

Bishop C M 2006 *Pattern Recognition and Machine Learning (Information Science and Statistics)* (New York: Springer) chapter 4 pp 191–2

Bouchard K E, Mesgarani N, Johnson K and Chang E F 2013 Functional organization of human sensorimotor cortex for speech articulation *Nature* **495** 327–32

Brendel W, Romo R and Machens C K 2011 Demixed principal component analysis *Advances in Neural Information Processing Systems* vol 24 ed J Shawe-Taylor *et al* (Cambridge, MA: MIT Press) pp 2654–62

Briggman K L, Abarbanel H D I and Kristan W B Jr 2005 Optical imaging of neuronal populations during decision-making *Science* **307** 896–901

Broome B M, Jayaraman V and Laurent G 2006 Encoding and decoding of overlapping odor sequences *Neuron* **51** 467–82

Buja A, Cook D, Asimov D and Hurley C 2005 Computational methods for high-dimensional rotations in data visualization *Handbook Statistics: Data Mining and Data Visualization* vol 24 (Amsterdam: Elsevier) pp 391–413

Churchland M M, Cunningham J P, Kaufman M T, Foster J D, Nuyujukian P, Ryu S I and Shenoy K V 2012 Neural population dynamics during reaching *Nature* **487** 51–6

Churchland M M, Cunningham J P, Kaufman M T, Ryu S I and Shenoy K V 2010 Cortical preparatory activity: representation of movement or first cog in a dynamical machine? *Neuron* **68** 387–400

Churchland M M and Shenoy K V 2007 Temporal complexity and heterogeneity of single-neuron activity in premotor and motor cortex *J. Neurophysiol.* **97** 4235–57

Churchland M M, Yu B M, Sahani M and Shenoy K V 2007 Techniques for extracting single-trial activity patterns from large-scale neural recordings *Curr. Opin. Neurobiol.* **17** 609–18

Churchland M M *et al* 2010 Stimulus onset quenches neural variability: a widespread cortical phenomenon *Nature Neurosci.* **13** 369–78

Cohen M R and Kohn A 2011 Measuring and interpreting neuronal correlations *Nature Neurosci.* **14** 811–9

Cohen M R and Maunsell J H R 2010 A neuronal population measure of attention predicts behavioral performance on individual trials *J. Neurosci.* **30** 15241–53

Cowley B R, Kaufman M T, Churchland M M, Ryu S I, Shenoy K V and Yu B M 2012 DataHigh: graphical user interface for visualizing and interacting with high-dimensional neural activity *EMBS'12: Proc. IEEE Conf. on Engineering in Medicine and Biology Society* vol 2012 pp 4607–10

Durstewitz D, Vittoz N M, Floresco S B and Seamans J K 2010 Abrupt transitions between prefrontal neural ensemble states accompany behavioral transitions during rule learning *Neuron* **66** 438–48

Harvey C D, Coen P and Tank D W 2012 Choice-specific sequences in parietal cortex during a virtual-navigation decision task *Nature* **484** 62–8

Luczak A, Bartho P and Harris K D 2009 Spontaneous events outline the realm of possible sensory responses in neocortical populations *Neuron* **62** 413–25

Machens C K, Romo R and Brody C D 2010 Functional, but not anatomical, separation of 'what' and 'when' in prefrontal cortex *J. Neurosci.* **30** 350–60

Macke J H, Buesing L, Cunningham J P, Yu B M, Shenoy K V and Sahani M 2011 Empirical models of spiking in neural populations *Advances in Neural Information Processing Systems* vol 24 ed J Shawe-Taylor *et al* (Cambridge, MA: MIT Press) pp 1350–8

Mante V, Sussillo D, Shenoy K V and Newsome W T 2013 Selective integration of sensory evidence by recurrent dynamics in prefrontal cortex *Nature* at press

Mazor O and Laurent G 2005 Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons *Neuron* **48** 661–73

Rigotti M, Barak O, Warden M R, Wang X-J, Daw N D, Miller E K and Fusi S 2013 The importance of mixed selectivity in complex cognitive tasks *Nature* **497** 585–90

Santhanam G, Ryu S I, Yu B M, Afshar A and Shenoy K V 2006 A high-performance brain–computer interface *Nature* **442** 195–8

Santhanam G, Yu B M, Gilja V, Ryu S I, Afshar A, Sahani M and Shenoy K V 2009 Factor-analysis methods for higher-performance neural prostheses *J. Neurophysiol.* **102** 1315–30

Shenoy K V, Sahani M and Churchland M M 2013 Cortical control of arm movements: a dynamical systems perspective *Ann. Rev. Neurosci.* **36** 337–59

Stevenson I H and Kording K P 2011 How advances in neural recording affect data analysis *Nature Neurosci.* **14** 139–42

Stopfer M, Jayaraman V and Laurent G 2003 Intensity versus identity coding in an olfactory system *Neuron* **39** 991–1004

Swayne D F, Temple Lang D, Buja A and Cook D 2003 GGobi: evolving from XGobi into an extensible framework for interactive data visualization *Comput. Stat. Data Anal.* **43** 423–44

Vidne M, Ahmadian Y, Shlens J, Pillow J W, Kulkarni J, Litke A M, Chichilnisky E J, Simoncelli E and Paninski L 2012 Modeling the impact of common noise inputs on the network activity of retinal ganglion cells *J. Comput. Neurosci.* **33** 97–121

Yu B M, Cunningham J P, Santhanam G, Ryu S I, Shenoy K V and Sahani M 2009 Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity *J. Neurophysiol.* **102** 614–35