MatClassRSA v2 release: A MATLAB toolbox for M/EEG classification, proximity matrix construction, and visualization

Bernard C. Wang[1], **Raymond Gifford**[1], **Nathan C. L. Kong**[2], **Feng Ruan**[3], **Anthony M. Norcia**[2,4], **and Blair Kaneshiro**[1,5,6,✉]

[1]Center for Computer Research in Music and Acoustics, Stanford University, Stanford, California, 94305, USA
[2]Department of Psychology, Stanford University, Stanford, California, 94305, USA
[3]Department of Statistics, Stanford University, Stanford, California, 94305, USA
[4]Wu Tsai Neurosciences Institute, Stanford University, Stanford, California, 94305, USA
[5]Center for the Study of Language and Information, Stanford University, Stanford, California, 94305, USA
[6]Graduate School of Education, Stanford University, Stanford, California, 94305, USA

MatClassRSA is a MATLAB toolbox that performs magnetoencephalography and electroencephalography (M/EEG) classification and other analyses related to Representational Similarity Analysis (RSA). The toolbox is designed for cognitive neuroscience researchers who wish to perform classification-based decoding analyses of their data—often repeated trials of evoked responses—or derive Representational Dissimilarity Matrices (RDMs) as input to RSA. Organized as a collection of user-called functions, MatClassRSA v2 comprises five main modules: Preprocessing, Reliability, Classification, RDM Computation, and Visualization. The Classification module supports multiple classifiers (LDA, RF, SVM) and classification schemes (e.g., multiclass, pairwise, cross-validated, train-test, hyperparameter optimization) and offers basic statistical analyses via permutation testing. Functions in other modules include, for example, trial-averaging and noise normalization; reliability estimation; non-classification RDM construction; and hierarchical and non-hierarchical clustering visualizations. The toolbox is freely available on GitHub under an MIT license and includes the main codebase, a User Manual, example function calls, and illustrative analyses. This preprint provides a general background of M/EEG classification for RSA as well as a narrative overview of the v2 MatClassRSA release, its updated functionalities, and illustrative analyses performed using the toolbox.

Representational Similarity Analysis | magnetoencephalography (MEG) | electroencephalography (EEG) | Representational Dissimilarity Matrix (RDM) | MATLAB | classification | machine learning | brain decoding | split-half reliability | multidimensional scaling (MDS) | hierarchical clustering

Correspondence: ***blairbo@ccrma.stanford.edu***

PREPRINT

Introduction

Classification involves the construction of a statistical model from categorically labeled data observations, which is then used to predict labels of new observations [1]. In the case of magnetoencephalography (MEG) or electroencephalography (EEG) data, observations can be single or group-averaged trials, and categorical labels can be, for example, descriptors of stimuli (e.g., 'face', 'cat', expected, unexpected), tasks (e.g., difficulty, attentional target), or participants (e.g., expert versus novice). M/EEG classification is considered a *decoding* approach, in that 'stimulus features' (broadly defined, and encompassing the above labeling examples) are predicted from the brain data [2]. Classification offers several advantages for analyzing multivariate neuroimaging data, including data-driven selection and weighting of features, predictive capabilities, and the ability to provide interpretable results across large stimulus sets.

Both MEG and EEG are considered suitable data modalities for multivariate analyses, including classification [3]. EEG classification has a long history in brain-computer interfaces (BCI), where accurate decoding—often from single data trials—is crucial for system performance [4]. In recent decades, however, M/EEG classification has also gained traction in cognitive neuroscience research and has been used to study, for example, behavioral performance [5] as well as representation of object categories [6–11], language categories [12], and stimulus modality [13]. In these contexts, research goals may shift away from optimizing decoding accuracy, and classification may rather be used to identify salient data features or as a means of obtaining neural measures of distance or similarity across a stimulus set through classifier accuracies or confusions, respectively.

This latter treatment of classifier accuracies as neural proximities supports the use of M/EEG data in Representational Similarity Analysis (RSA). As explained in seminal RSA papers by Kriegeskorte and colleagues [14, 15], the core tenet of this approach is the abstraction of multiple models, response data, or other representations of a stimulus set away from their native formats (e.g., M/EEG trials, image statistics, perceptual ratings) to a Representational Dissimilarity Matrix (RDM) for each, which summarizes all pairwise distances across the set. There exist various methods for computing RDMs, including correlations, difference vectors, or—as noted above—classifier outputs, and pairs of RDMs can be rank-correlated to assess their representational similarity. In all, RDM representations enable data that may have been difficult to compare directly in disparate original modalities—for example, a set of speech waveforms with a set of EEG responses—to be quantitatively compared in a common similarity space.

RSA has been applied in numerous M/EEG studies. Classifier accuracies and confusions have been compared to perceptual, acoustical, descriptor-based, and computational representations of stimuli [10, 12, 16, 17]. Classification-based RDMs can be visualized not only as matrices but

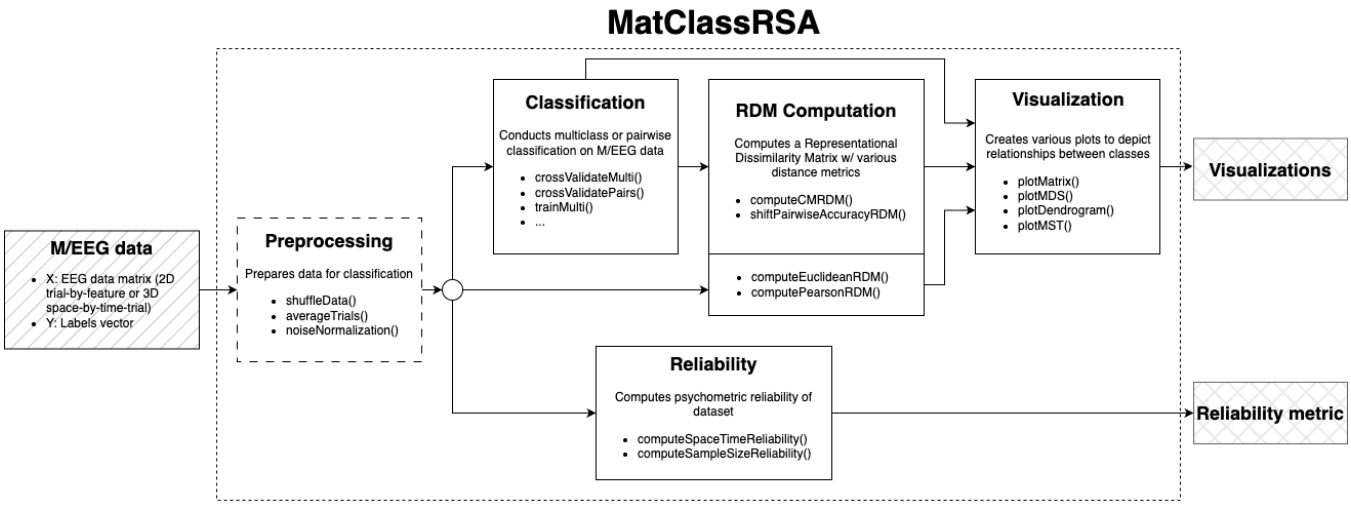

Fig. 1. Summary of MatClassRSA functionalities. The toolbox is organized into modules that each focus on different types of analyses. As shown in the flowchart, modules can be arranged according to the user's analysis goals. MatClassRSA operates on cleaned, ready-to-analyze data matrices and stimulus label vectors.

also via hierarchical or non-hierarchical clustering methods to convey the neural proximity structure of the stimuli [7, 9, 18]. Moreover, classifications can be performed over temporal and/or spatial subsets of multichannel M/EEG data [6, 7, 9, 19, 20] to highlight spatiotemporal features relevant to discriminating between classes. The high temporal resolution of M/EEG data also makes their RDMs a useful complement to fMRI in multimodal RSA analyses, for example to relate temporal dynamics (from M/EEG RDMs) to precise localization (from fMRI RDMs) by identifying similar RDMs across response modalities [8, 21].

In this preprint, we introduce the v2 release of MatClassRSA, a MATLAB toolbox centered around M/EEG classification for RSA analyses. As summarized in Fig. 1, the toolbox encompasses Preprocessing, Reliability, Classification, RDM Computation, and Visualization functionalities in a modular structure. MatClassRSA is geared toward cognitive neuroscience researchers working with M/EEG data and MATLAB—particularly those working in ERP paradigms, which are a common data type for classification. As MatClassRSA is a function-based toolbox (i.e., there is no graphical user interface (GUI) at this time), users should be comfortable writing code to load input data, call functions, and save output data and figures. In terms of scope, MatClassRSA is designed to work with data that have already undergone standard cleaning steps, and ends at the construction and visualization of RDMs. Users are encouraged to make use of complementary toolboxes such as Delorme and Makeig's EEGLAB [22] for data cleaning and Nili et al.'s RSA Toolbox [23] for downstream RSA analyses.

The MatClassRSA v2 release is a major update and expansion of the original 2017 release [24]. It includes additional functionalities, a restructuring of the codebase, expanded documentation, example function calls, and new illustrative analyses. The v2 release includes three main deliverables:

1. The publicly available GitHub repository,[1] which includes the codebase, User Manual, example function calls, and illustrative analyses.
2. This preprint, which provides a narrative overview of the toolbox.
3. A standalone dataset containing example data for the illustrative analyses [25].[2] The GitHub repository includes code to download these data files into the user's local copy of the toolbox.

The remainder of this preprint is structured as follows. In the next section, we describe the codebase and its contents. After that, we cover the functionalities of the toolbox, including the main user-called functions. Then, as a demonstration of how MatClassRSA functions can be combined for larger-scale analyses, we present a narrative overview of selected illustrative analyses. We close with a discussion of the toolbox's affordances and limitations.

Codebase

Code access and license. MatClassRSA is freely available on GitHub (see footnote 1). MatClassRSA v2 is released under the MIT license.[3] If using any part of the codebase, we ask that you cite this preprint and to cite the related dataset if using any of those contents for outside projects [25].

Dependencies. Usage of MatClassRSA requires a MATLAB[4] license as well as MATLAB's Parallel Computing Toolbox[5] and Statistics and Machine Learning Toolbox.[6] A number of MatClassRSA illustrative analyses require the

[1] https://github.com/berneezy3/MatClassRSA
[2] https://purl.stanford.edu/kv831rr3606
[3] https://choosealicense.com/licenses/mit/
[4] https://www.mathworks.com/products/matlab.html
[5] https://www.mathworks.com/products/parallel-computing.html
[6] https://www.mathworks.com/products/statistics.html

PREPRINT

MATLAB Image Processing Toolbox.[7] The Support Vector Machine (SVM) classifications of the toolbox use the LIBSVM library, [26][8] which is included in the MatClassRSA repository; installation instructions are provided in the User Manual. Otherwise, the toolbox is self-contained, meaning that all helper functions are included in the MatClassRSA GitHub repository, and no other external codebases or toolboxes need to be added to the user's path.

The current release was developed and comprehensively tested on recent MacOS operating systems (12.* and higher) and is expected to work on Linux and Windows platforms as well. The code was developed with the aim of supporting older versions of MATLAB (2016b and higher, e.g., with older versions of input-parser functionality) and has been fully validated on MATLAB versions 2021a and higher.

Folders and files in the repository. The top level of the repository contains the following files and folders:

ExampleData. This folder stores example data required to run the illustrative analyses. Due to file size, the example data files are not directly included in the repository; rather, the illustrative analyses include a script that will download the necessary files into this folder.

ExampleFunctionCalls. This folder contains one script per user-called MatClassRSA function. Each script contains one or more runnable examples of the corresponding function.

IllustrativeAnalyses. This folder contains the script that downloads the example data into the ***ExampleData*** folder (required in order to run the illustrative analyses) as well as one script per illustrative analysis.

src. This folder is the main folder for the MatClassRSA codebase. It contains MATLAB package folders ('+' folders) for each of the five MatClassRSA modules as well as for the `+Utils` folder containing the helper functions. The `libsvm-3.24` folder is also a subfolder of ***src***.

.gitignore. This file lists any files or folders in the repository that should remain untracked by Git. Of special relevance to MatClassRSA users is that all contents of the ***ExampleData*** folder will be ignored.

LICENSE.md. License for the repository (MIT license).

MatClassRSA v2 User Manual.pdf. This file is the User Manual of the v2 release. It contains detailed information about getting up and running with the toolbox; comprehensive documentation of the user-called functions; supplemental overview and design rationales for the Classification module; details of the illustrative analyses; and brief documentation of helper functions.

README.md. README for the repository.

[7] https://www.mathworks.com/products/image-processing.html
[8] http://www.csie.ntu.edu.tw/~cjlin/libsvm

Code/data origins and past usage. Design factors of many MatClassRSA functions were drawn from the analyses reported by Kaneshiro et al. [9]. Certain Preprocessing and RDM Computation functions were adapted from or directly use code from Guggenmos et al. [27];[9] their specific attribution is provided here below, in the User Manual, and in MatClassRSA function docstrings. The MatClassRSA example data [25] include data files from other published datasets [28, 29], as attributed in the dataset documentation. Previous versions of MatClassRSA functions were used in published works by Kong et al. [10] and Losorelli et al. [18].

Functionalities of the toolbox

The functionalities of the toolbox are summarized in Fig. 1. As noted previously, the main MatClassRSA code module folders, helper functions folder, and LIBSVM folder can be found in the ***src*** folder of the GitHub repository.

Data specifications. MatClassRSA is focused specifically on classification and RSA-adjacent analyses, and does not perform data cleaning. However, the input data specifications are aligned with the 3D space × time × trial matrix format as output by EEGLAB [22]; hence, EEGLAB is encouraged for data cleaning. Many MatClassRSA functions involve the following core inputs:

1. Input data matrix `X`: This can be a 3D space × time × trial matrix as noted above, or a 2D trial × feature matrix (for single-sensor data, for example).
2. Labels vector `Y`: This vector contains numeric stimulus labels of each trial; its length should be the size of the trial dimension of `X`.

Functions also often involve numerous additional and/or optional inputs, which are detailed in function docstrings and in the User Manual. Alternatively, some MatClassRSA functions take in outputs of other functions; for example, Visualization functions take in matrices output by Classification and RDM Construction functions.

MatClassRSA functions output variables and/or figures. Users can specify in their own code how they would like these saved in e.g., .mat or image files.

Calling MatClassRSA functions. In order to use MatClassRSA, the repository needs to be in the user's MATLAB path. Once the toolbox is in the path, specific functions are called by referencing both the MATLAB package folder name (without the '+') and the specific function name. For example, to call the `noiseNormalization()` function in the `+Preprocessing` folder, the user would type the following (without the line break):

```
[normData, sigmaInv] =
Preprocessing.noiseNormalization(X, Y);
```

[9] https://github.com/m-guggenmos/megmvpa/blob/master/tutorial_matlab/matlab_distance.ipynb

PREPRINT

For succinctness, further description of MatClassRSA functions in this preprint reference only the function name, without the MATLAB package folder name prepended.

Main user-called functions. We now provide a narrative overview of the main user-called functions. For more information on the functions, users are encouraged to visit the User Manual in the GitHub repository (see footnote 1), which contains detailed descriptions, specifications, and example function calls. The docstrings of these main functions also contain detailed input/output specifications, and the scripts in the ***ExampleFunctionCalls*** and ***IllustrativeAnalyses*** folders of the repository illustrate various input/output specifications.

Module: Preprocessing. This module includes three functions that can be applied to the data prior to all other modules' functions. First, **`shuffleData()`** randomizes the ordering of the trials, while keeping each trial paired with its correct label (and participant identifier, if provided). This function may be useful, for example, when the input data represent observations from multiple recordings and/or participants, with the trials ordered by recording and/or participant; the user may wish to randomize the ordering of the trials prior to calling a cross-validation classification function so that data from a given recording or participant are distributed across the cross-validation folds.

The second function, **`averageTrials()`** trial-averages responses to a common stimulus (and, if specified, from a common participant). As noted by Guggenmos et al. [27], these 'pseudo-trials' offer higher signal-to-noise ratio (SNR) than single trials, which may improve classifier performance. However, trial averaging groups of N trials also reduces the number of observations by roughly a factor of N (subject to remainder handling). Therefore, in practice, users who are interested in operating on pseudo-trials rather than single trials may want to examine tradeoffs between SNR improvements and decreased sample size when group-averaging their specific dataset.

Finally, **`noiseNormalization()`** accounts for variable SNR captured in each sensor and upweights or downweights the data from each sensor on the basis of its SNR. Guggenmos et al. have reported positive impacts of noise normalization on subsequent analyses such as classification [27]. However, our author team has found that at times, noise normalization can negatively impact downstream results when using MatClassRSA classification functions. Users are thus advised to examine effects of this operation on their data; more information is provided in the User Manual. This function is adapted from the code tutorial provided by Guggenmos et al. [27][10] and uses a helper function called `cov1para()`, created by Olivier Ledoit and Michael Wolf and imported directly from the Guggenmos tutorial into MatClassRSA.

[10] https://github.com/m-guggenmos/megmvpa/blob/master/tutorial_matlab/matlab_distance.ipynb

Module: Reliability. Users may wish to assess the *reliability* of their data—that is, the similarity of observations across repeated instances of the data—as a complement to classification. Reliability metrics can be useful in modeling contexts because they provide an upper bound on how well *any* model will be able to explain the data. Previous versions of these functions were used for the reliability analyses reported in Figure 2 of Kong et al. [10].

The MatClassRSA Reliability module includes two functions for such assessments. Broadly, these functions involve random partitioning of the data into two split halves, trial-averaging the data on a per-stimulus basis, and then correlating the results across the stimulus dimension. These operations are repeated across several iterations, each involving a different partitioning of the data. Since splitting the data produces lower reliability estimates than if all data had been considered at once (as in e.g., a decoding analysis), the functions also apply the Spearman-Brown correction, which provides a reliability estimate across all of the data.

The **`computeSpaceTimeReliability()`** function computes reliabilities at each sensor and/or time point. Reliabilities can be reported as temporal trajectory for a single sensor, or a spatial distribution at a single time point; or, time- or space-averaged reliability can be reported for a given sensor or time point, respectively. The second function in this module, **`computeSampleSizeReliability()`**, computes reliabilities for a given sensor or time point over varying numbers of trials, which can be useful when considering the general role of sample size in classification or other RSA results.

Module: Classification. This module contains various functions for training and testing classifier models. Functions are separated according to multiclass and pairwise contexts. Given N stimulus categories, a multiclass classification is choosing among all N categories in model training and prediction. The output is an $N \times N$ confusion matrix; each element i, j expresses how many observations actually from class i were predicted by the model to be from class j (correct classifications are thus listed on the matrix diagonal, where $i = j$). On the other hand, pairwise classifications perform separate 2-class classifications for each pair of stimuli. Outputs from pairwise classifications are summarized in an $N \times N$ matrix, where off-diagonal elements denote the classification accuracy between classes i and j.

Functions are designed to accommodate cross validation as well as standalone training and testing (prediction) functions. Cross validation is an iterative training and testing procedure. In K-fold cross-validation, the entire dataset is partitioned into K non-overlapping subsets called 'folds'. Then, the classifier is trained and tested K times, where one fold is withheld for testing (i.e., not used for training the model) each time. Across all of the folds, each observation is used as a test observation one time. In cases where users do not wish to perform classification iteratively—for example, when

PREPRINT

some data are *only* for training or testing—MatClassRSA also offers functions that only train on the input data; these functions output models, which can be input alongside test data of the user's choice in a predict-only function.

Finally, this module offers multiple classifiers with varying degrees of hyperparameter optimization. First, Linear Discriminant Analysis (LDA) [1] is a popular 'out-of-the-box' classifier that can be run without needing to tune any of its parameters. Next, Support Vector Machine (SVM) [30] contains hyperparameters that can be tuned based on attributes of the input data to further optimize classifier performance. Finally, MatClassRSA offers a Random Forest (RF) classifier [31]; this classifier also has tunable hyperparameters, though RF hyperparameter optimization is not implemented in MatClassRSA at this time.

Across these three dimensions—multiclass/pairwise, cross validation/separate train-test, and with or without hyperparameter optimization—MatClassRSA offers nine classifier functions. Cross-validation functions **`crossValidateMulti()`** and **`crossValidateMulti_opt`** handle multiclass classification, while **`crossValidatePairs()`** and **`crossValidatePairs_opt()`** handle pairwise classifications—with and without hyperparameter optimization, respectively. Separate from cross validation, **`trainMulti()`** and **`trainMulti_opt()`** handle multiclass training only while **`trainPairs()`** and **`trainPairs_opt()`** handle pairwise training only, again with and without hyperparameter optimization, respectively. These `train` functions output models, which can then be applied to new data using the **`predict()`** function.

In addition to the core inputs of the data matrix `X` and labels vector `Y`, users can specify subsets of data features to classify, classifier selection and—when applicable—hyperparameter optimization, feature reduction using Principal Components Analysis (PCA), data centering and scaling, and data partitioning for cross validation and/or hyperparameter optimization. When test observations have labels, statistical significance can be calculated using permutation testing—i.e., comparing the observed result to a null distribution comprising many classification attempts in which stimulus labels were randomized independently of the data trials [32, 33]. Outputs vary by function and can include the specifications of the function call; data-handling parameters related to e.g., PCA, feature subsetting, and train-dev-test partitioning; classification models; predicted labels; and p-values.

Module: RDM Computation. The RDM Computation module contains functions that operate both on proximity matrices (such as those output from the Classification module) and directly on the original data, such as the data matrices input to the Classification functions.

MatClassRSA Classification functions perform both multiclass and pairwise classifications. The confusion matrices output by multiclass classifications can be considered a form of similarity matrix, in that two stimuli that are confused (misclassified) more can be considered more similar than two stimuli that are never confused for one another during classification. However, as confusion matrices are count-based and also include variable values on the diagonal, they may not be optimal RDMs as is. To address this, the **`computeCMRDM()`** function includes options to normalize the incoming confusion matrix—for example, to represent the confusion matrix as estimated conditional probabilities [12, 34] or enforce a self-similarity measure of 1 along the diagonal [34]—as well as symmetrize, convert similarities to distances, and rank the distance values.

Similarly, the matrix of accuracies output from pairwise classifications may benefit from further transformation toward being considered an RDM. Pairwise accuracies can already be considered distance measures (higher accuracy representing better decodability and hence greater distance), but pairwise accuracies can range from 0 to 1, while chance-level classification—which would imply a distance of 0—is 0.5 [27]. Hence, the **`shiftPairwiseAccuracyRDM()`** function simply subtracts 0.5 from each element of an input matrix of pairwise accuracies.

The other functions in this module operate directly on the original input M/EEG data. **`computeEuclideanRDM()`** uses cross-validation to compute estimated pairwise distances of the original sensor-space data across the set of stimuli based on Euclidean distance. **`computePearsonRDM()`** similarly provides cross-validated distance estimates, based on Pearson correlation. The cross-validation procedures applied in these functions involve multiple train/test splits of the data followed by distance or correlation measures, respectively; this approach is recommended over simply computing single distance or correlation measures across all data at once as the cross validation helps to remove noise components [27]. Both of these functions are adapted from the code tutorial provided by Guggenmos et al. [27].[11]

Module: Visualization. Finally, the Visualization module includes functions intended to visualize RDMs and other matrices. First, the **`plotMatrix()`** function creates an 'image' visualization of an input matrix, for example a Classification multiclass confusion matrix or pairwise accuracy, or an RDM from the RDM Computation module. The elements of the rendered image are colored according to the corresponding value, and the function call includes options to e.g., print imaged values in the matrix elements and customize attributes of the axes and labels.

The remaining functions in this module are intended for RDM inputs specifically. First, **`plotMDS()`** computes Multidimensional Scaling (MDS) on the input RDM and plots

[11] https://github.com/m-guggenmos/megmvpa/blob/master/tutorial_matlab/matlab_distance.ipynb

PREPRINT

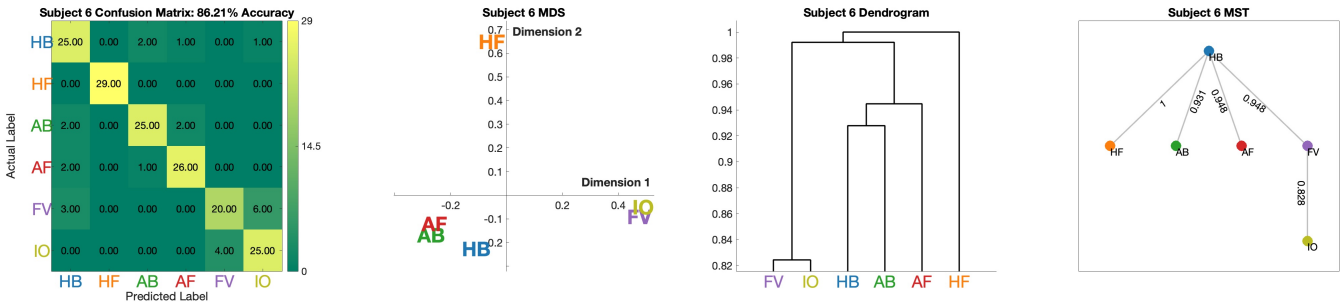

Fig. 2. Example visualizations produced by the functions in the Visualization module. From left to right: The `plotMatrix()` function produces an image of the input matrix, which in the realm of MatClassRSA is most likely to be a multiclass confusion matrix of counts (as shown here), matrix of pairwise accuracies, or RDM; the `plotMDS()` function computes multidimensional scaling of an input RDM and then plots the coordinates of the stimuli along selected MDS dimensions; the `plotDendrogram()` function performs hierarchical clustering of an RDM and visualizes the distances among the stimuli in a tree representation; the `plotMST()` function also visualizes hierarchical clustering, this time as the paths between all stimuli for which overall distance is minimized.

coordinates of the stimulus exemplars for the specified MDS dimensions. MDS is a non-hierarchical clustering technique that converts the pairwise distances of a stimulus set to coordinates along multiple ordered dimensions [35, 36]. The stimuli can then be plotted according to their coordinates from selected dimensions—often the first two dimensions, which are the most important MDS dimensions; this can be useful for depicting clusters or separations that were not visible in the matrix view. Next, **`plotDendrogram()`** computes hierarchical clustering of the RDM and visualizes it as a dendrogram (tree) structure [37]. In this view, each stimulus is represented as a separate 'leaf' of the dendrogram, and the distance between any two stimuli is represented by the distance up the tree that one needs to traverse in order to go from one leaf to the other. Finally, **`plotMST()`** visualizes the distance structure of the RDM exemplars as a Minimum Spanning Tree [38]. This is another hierarchical clustering technique, whose visualization again displays all items of the stimulus set, now linked by the path that represents the minimal total distance across the set. Example visualizations produced by the functions in this module are shown in Fig. 2.

Helper functions. The main functions of the toolbox collectively rely on a number of helper functions. These helper functions are stored together in the `+Utils` folder. The MatClassRSA User Manual includes an Appendix in which each helper function is briefly documented with function-call syntax, a narrative description, dependencies, and usage by other main and helper functions.

Illustrative analyses

Along with the detailed function-level documentation provided in the code docstrings and in the User Manual, MatClassRSA also includes a number of illustrative analyses. These are intended to demonstrate how multiple functions can be called together in more elaborate analyses, and also to illustrate further the functionalities of the toolbox and some of the customizable parameters of the function calls. The ***IllustrativeAnalyses*** folder of the GitHub repository contains one script per analysis documented below, which can be run by users once the example data are also downloaded.

In this section, we highlight selected results from each of the illustrative analyses provided with the toolbox. Users are encouraged to visit the User Manual in the GitHub repository (see footnote 1) to see the complete set of analyses and elaborated explanations of the analysis workflow, code snippets, and further interpretation of results.

Illustrative analysis 0: Download of example data files for illustrative analyses. The first illustrative analysis script does not perform an analysis per se, but rather downloads the example data files which, due to size, are not provided in the GitHub repository and are instead provided in a separate online dataset [25].

Script for this analysis:
`illustrative_0_downloadExampleData.m`

This script locates the ***ExampleData*** folder in the user's local MatClassRSA instance (if more than one instance is found, the first one will be used). It then iterates through (by default) all files in the online data repository [25] and downloads any that are not already found in the ***ExampleData*** folder. Users can specify a subset of files to download, or further edit the script to download files into a different directory.

The example data files are as follows:

- **Object category EEG**:

  `S01.mat`, `S04.mat`, `S05.mat`, `S06.mat`, `S08.mat`

  Single-trial EEG responses to 72 images from six visual object categories (Human Face–HF, Human Body–HB, Animal Face–AF, Animal Body–AB, Fruit/Vegetable–FV, Inanimate Object–IO). Each file contains data from a different adult participant. Each participant experienced each image 72 times in an ERP paradigm. Data files are taken directly from the OCED dataset [28]. For more information on the data and originating study, see Kaneshiro et al. [9].

- **Frequency-following responses (FFRs)**:

  `losorelli_100sweep_epoched.mat`

  Single-channel frequency-following responses (FFRs), which are auditory responses that phase lock to acous-

PREPRINT

tical stimuli. Each of 13 adult participants experienced six speech and music stimuli with a fundamental frequency around 100 Hz. Each stimulus was presented 2500 times per participant. Every observation represents the average of 100 stimulus presentations for a given participant and stimulus. This data file is taken directly from the STAR-FFR-1 dataset [29]. More information on the data and the study for which they were collected can be found in Losorelli et al. [18].

- **Supporting files**:

  OCEDStimuli.zip

  Stimulus images for the OCED data files noted above. If the illustrative script downloads this file, it will also unzip it. These 72 images are a subset of a database of 92 images provided by Kriegeskorte et al. [15].[12] This database, and the larger set of images from which its images are drawn, have been used in several RSA studies on visual object category processing (e.g., [8–10, 14, 15, 39]).

  MatClassRSA_v2_ExampleData_README.pdf

  Informational document describing the dataset.

Illustrative analysis 1: Impact of preprocessing steps on downstream analyses. Additional data preparations, such as those provided in the Preprocessing module, can substantially impact downstream analyses. This illustrative script explores how the trial-averaging, noise-normalization, and data-shuffling functions influence characteristics of the data, multi-channel covariance, and classification outcomes.

Script for this analysis:
`illustrative_1_impactOfPreprocessing.m`

For example, trial averaging (using `averageTrials()`) can improve SNR of the data, while also reducing the overall voltage range of the pseudotrial averages. Fig. 3 visualizes single trials and 40-trial averaged pseudotrials of data-shuffled, noise-normalized object category data from `S01.mat` at electrode 96 (located at T4). Trial averaging makes the N170 more visually apparent reduces the overall voltage range of the data. However, trial averaging also reduces the number of available observations.

Noise normalization scales each channel of data according to the inverse square root of the channel-by-channel covariance matrix. Guggenmos et al. found that this step substantially improved downstream decoding performance [27]. In the present example shown in Fig. 4, the plots on the left illustrate how calling the `noiseNormalization()` function changes the covariance structure of the `S01.mat` data. Contrary to Guggenmos et al., in the right-hand plots we find that within-participant classification performance for this specific dataset is slightly lower after noise normalization (66.47% with noise normalization in the bottom plot

[12] https://bit.ly/3Ma1ZSZ, accessed November 19, 2025.

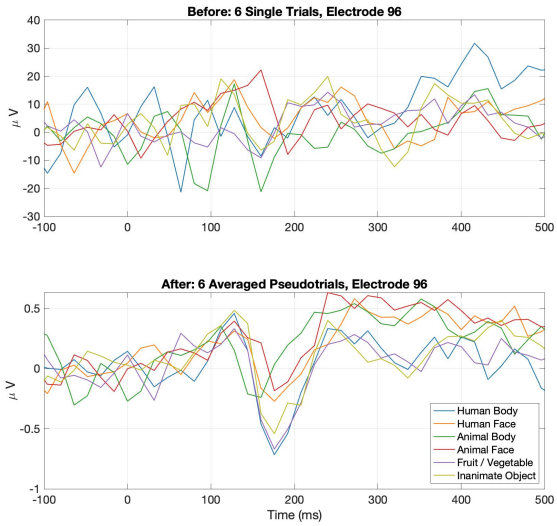

Fig. 3. Effect of trial averaging. Visual responses to images from six object categories from `S01.mat` were averaged at a single electrode. The top plot shows single trials from each category, while the bottom plot shows averages of 40 trials per category.

compared to 78.68% without in the top plot).

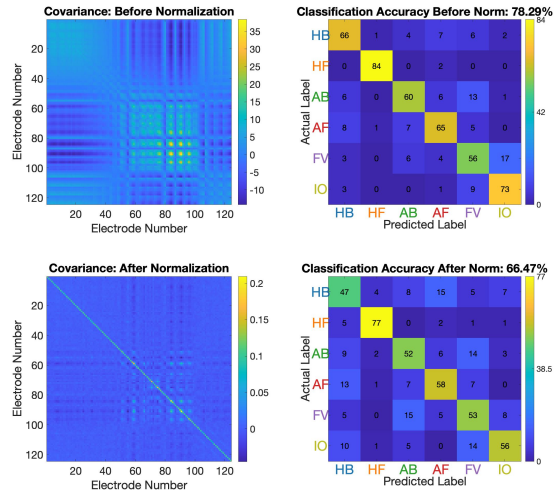

Fig. 4. Noise normalization covariance and within-subject classification. Object category data from `S01.mat` underwent classification with and without noise normalization. The left-hand matrices illustrate the covariance matrices of the data before and after noise normalization. On the right are the 6-class classification confusion matrices. Within-subject accuracy is slightly lower after noise normalization.

Yet, when a classifier is trained on data from one participant and tested on another, we do see benefits of noise normalization. Fig. 5 illustrates an extreme case of this with SVM classifiers trained on object category data from `S01.mat` and then tested on data from `S04.mat`, `S05.mat`, and `S08.mat`. When noise normalization is not applied (top row of plots), confusion matrices from predicting on data from unseen participants show that all predictions have gravitated to the first category, resulting in classification accuracies exactly at chance level (16.67%). We surmise this

is due to the classifier overfitting to training-data features that do not generalize to test data from new participants. When noise normalization is applied (bottom row of plots), the confusion matrices become somewhat more balanced, and classification accuracy improves to 25.48% and higher.

Given the variable impacts of noise normalization between these within-participant and across-participant examples, users are encouraged to conduct exploratory analyses with their own data to determine whether to include noise normalization in their analysis workflow.

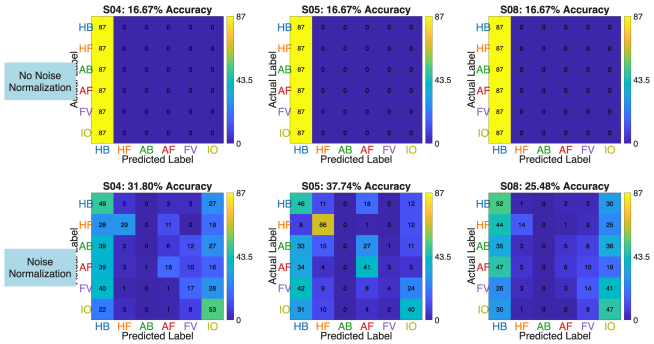

Fig. 5. Noise normalization covariance and across-subject classification. SVM classifiers were trained on object category data from `S01.mat` and then tested on data from `S04.mat`, `S05.mat`, and `S08.mat`, with and without noise normalization. The top plots show poor classifier performance when noise normalization is not used. The bottom plots show that applying noise normalization to the data improves across-subject prediction accuracy.

This script also illustrates how users can visualize their data—for example, to confirm that `shuffleData()` is working as expected. As an example, we show how the FFR dataset—which includes both stimulus labels and optional participant labels for each trial—can be passed into this function with the optional specification to keep participant identifiers coupled with the data observations and labels. Fig. 6 shows the category labels of the ordered trials, before and after data shuffling. As shown in the top plot, the trials as initially loaded are ordered by stimulus: All trials from category 1 followed by all trials from category 2, and so on. If this trial ordering were carried into cross validation, certain classes would be over- or under-represented in each cross-validation partition; as an extreme example, performing 6-fold cross-validation on this ordering of the trials would equate to partitioning the data exactly by class, ensuring that the class of each holdout test partition is not represented *at all* in the training partitions! The bottom plot of this figure shows that after data shuffling, the six categories are well distributed throughout the ordered trials, ensuring a good balance of classes across future data partitions. Fig. 7 depicts the per-participant averaged data across all stimuli before and after data shuffling (each line represents a participant), providing visual confirmation that the mappings between data trials and participant labels are preserved during data shuffling.

Additional sub-analyses, figures, and interpretations for this illustrative analysis can be found in the User Manual.

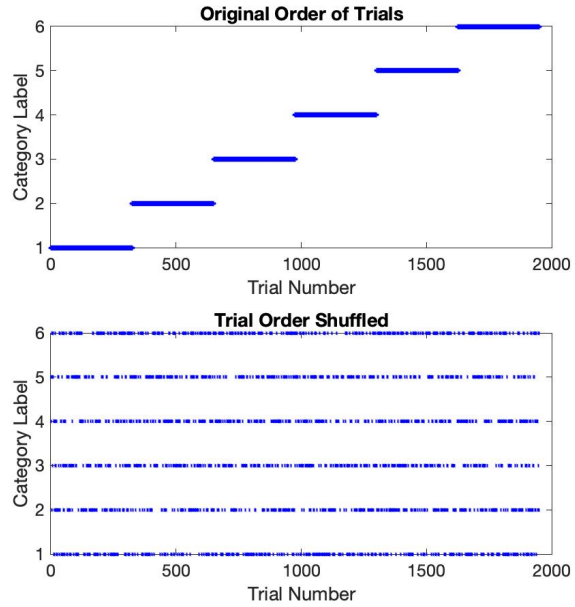

Fig. 6. Distribution of trials before and after data shuffling. The example data file from the Losorelli et al. dataset [29] initially has trials grouped by stimulus. Shuffling the data ensures a better distribution of the stimuli across the dataset.

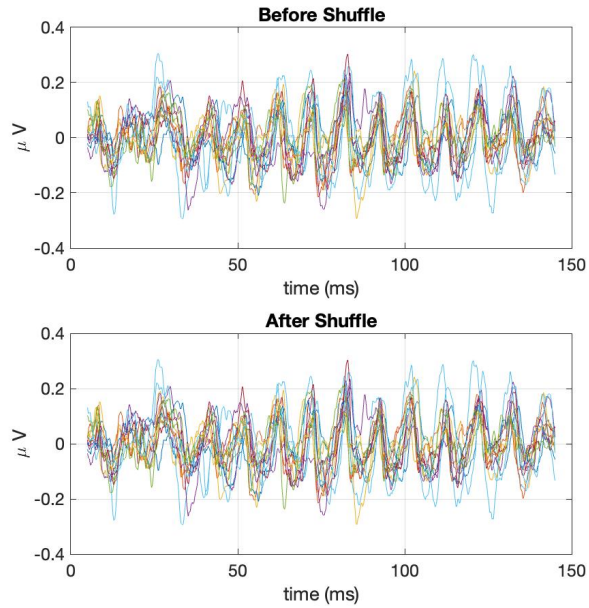

Fig. 7. Per-participant averaged data are consistent before and after data shuffling. FFR responses phase locked to acoustical stimuli with a fundamental frequency of 100 Hz were input to the `shuffleData()` function with the optional participant identifier included. Each line represents the average of all data (across all stimuli) for one participant before and after averaging. This figure provides visual confirmation that the function preserved the data-participant mappings.

Illustrative analysis 2: Single-channel analyses. Multi-channel M/EEG data offer rich spatiotemporal information. As an alternative to visualizing spatial weight vectors output by classifiers—which can lack visual smoothness (e.g., [13])

and require special care in interpretation [40]—single-channel analyses can offer broad insight into relevant spatial features.

Script for this analysis:
`illustrative_2_singleChannelAnalyses.m`

For example, users may wish to identify electrodes that record reliable data. In Fig. 8, single-channel reliabilities for object category participant `S01.mat`, computed using `computeSpaceTimeReliability()`, are displayed on a scalp map, highlighting especially reliable data recorded from electrodes over right occipital cortex.

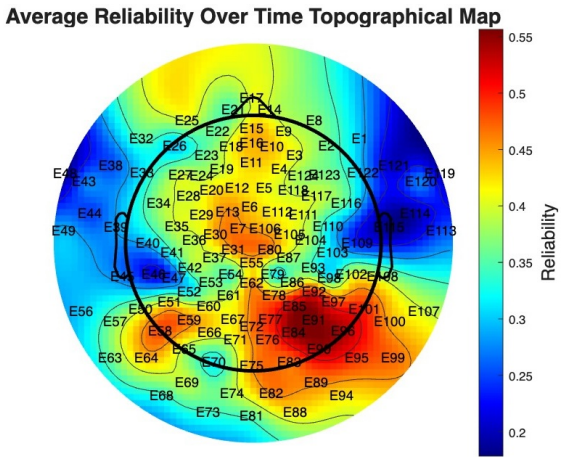

Fig. 8. Topomap of single-channel reliabilities. Average reliability across all time points was computed for each sensor of the `S01.mat` object category data and visualized on a scalp map. Sensors over right occipital cortex recorded the most-reliable data.

While reliability analyses can help to identify channels that record consistent data over repeated trials, single-channel classification analyses can be a useful complementary analysis to derive scalp maps highlighting sensor locations whose data classify well and may therefore be task-relevant. An example from the same participant, with data classified using LDA with the `crossValidateMulti()` function, is shown in Fig. 9. We see broad alignment between this scalp map and the reliability map, as well as with the group-averaged results originally reported by Kaneshiro et al. [9].

Further elaboration, figures, and interpretations for this analysis are described in the User Manual.

Illustrative analysis 3: Time-resolved analyses. One of the greatest assets of M/EEG data is high temporal resolution. While multivariate analyses such as classification can optimize high-dimensional data, restricting input data to pre-identified relevant temporal features can aid data interpretation, reduce processing time, and even boost classifier performance. MatClassRSA functions can be used to analyze data subsets along the time dimension.

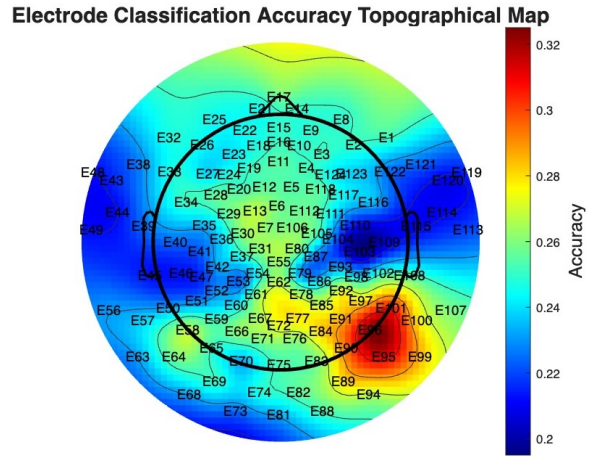

Fig. 9. Topomap of single-channel classification rates. Six-class classifications of the object category data from `S01.mat` were performed independently on data from each electrode. Similar to the reliabilties map, the topomap of classifier accuracies highlights right occipital cortex.

Script for this analysis:
`illustrative_3_timeResolvedAnalyses.m`

First, as a complement to single-channel reliabilities, the `computeSpaceTimeReliability()` function can also be used to compute reliabilities across all electrodes at each time point; the results can then be plotted as time course. Fig. 10 shows the reliability over time of object category participant `S01.mat`, computed at the exemplar (72-class) level across all electrodes. As can be seen from the plot, reliability remains near zero in the pre-stimulus and early post-stimulus intervals until around 80 msec post-stimulus onset—around the time of the P1 component. It then peaks around the timing of the N170 component and decreases toward the end of the response epoch.

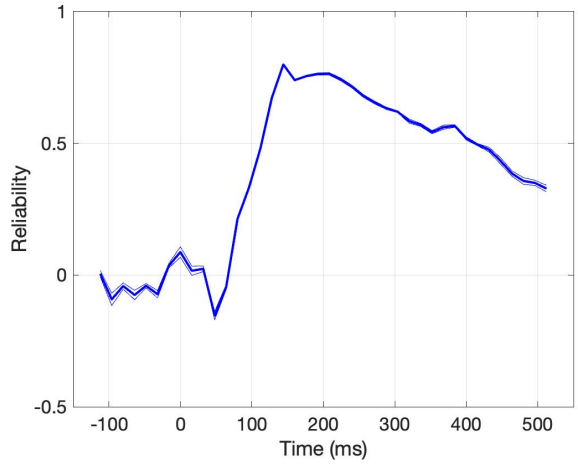

Fig. 10. Time-resolved reliability, averaged across electrodes. Reliability, averaged across all electrodes, was computed for each time point of `S01.mat` data and visualized. Non-negative reliabilities correspond in time to the P1 ERP component, and peak around the time of the N170 component.

This type of temporal information can be useful in specifying classification analyses as well. Fig. 11 involves the classification of shuffled, group-averaged data (15 trials per

PREPRINT

group) from object category data from `S01.mat` using `crossValidateMulti()` with LDA. Here, when data from all time points (from -112 to 512 msec relative to stimulus onset) and electrodes are included for 6-class classification, the classifier accuracy of 42.69% is well above chance level of 16.67%. However, when data from only 128 to 208 msec post-stimulus onset are input to the classifier, the classifier accuracy improves to 69.88%, with a stronger confusion matrix diagonal.

Additional reliability, ERP, and classification results as well as visualizations are described in detail in the User Manual.

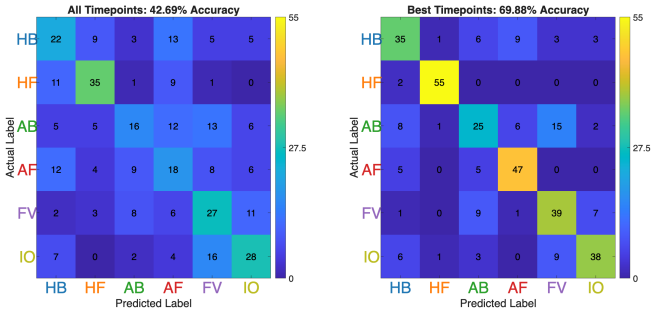

Fig. 11. Confusion matrices when classifying over all time and best time window. Six-class classifications of the object category data from `S01.mat` were performed across the entire response epoch (-112 to 512 msec—left) as well as across the best time points (128 to 208 msec—right) as identified by a sliding-time-window classification (described in the User Manual). Compared to classifying all available data, restricting the time dimension of the input data to the best-performing segment improves classification performance from 42.69% to 69.88%.

Illustrative analysis 4: Train-test SVM optimization. MatClassRSA classification functions with optimization offer fairly broad default grids over which to tune the SVM hyperparameter(s). In this illustrative analysis, we demonstrate how SVM hyperparameters can be tuned over successive grid searches, where optimal parameters from one attempt can specify a finer grid in the next, and further improve classifier performance over hyperparameters optimized in the default grid specification only.

Script for this analysis:
`illustrative_4_trainTestOptimizeSVM.m`

To begin, we performed a 6-class SVM classification on the object category data from `S01.mat`. We called the `crossValidateMulti_opt()` function from the Classification module, using the rbf kernel and optimizing γ and `C` using default grid parameters (5 logarithmically spaced points between 10^{-5} and 10^{5}). As shown in Fig. 12, classifier performance was 66.48%, well above a chance level of 16.67%.

The optimal hyperparameters identified in the first classification's default grid search were γ of 0.0032 and `C` of 316. In a follow-up classification, we specified finer-grained γ and `C` grid parameters around these initial optimal values. This second round of optimization identified optimal gamma

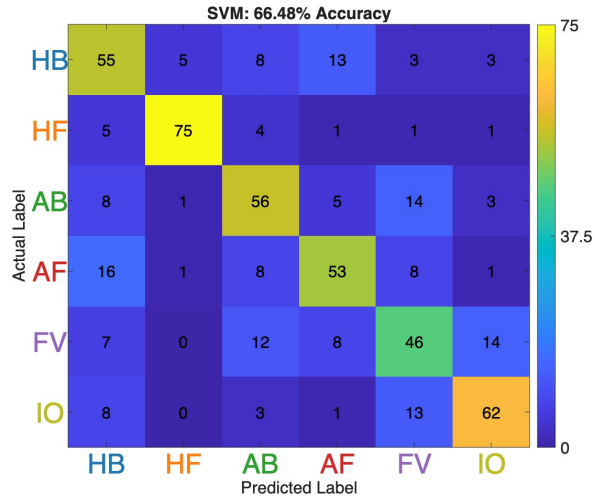

Fig. 12. Confusion matrix for SVM classification with default grid-search hyperparameters. Six-class classifications of the object category data from `S01.mat` were performed using SVM with an rbf kernel. Hyperparameters γ and `C` were optimized over the default grid parameters.

of 0.0146 and optimal C of 1000, and improved classifier performance to an accuracy of 81.23% as shown in Fig. 13.

Further elaboration and interpretations for this analysis are described in the User Manual.

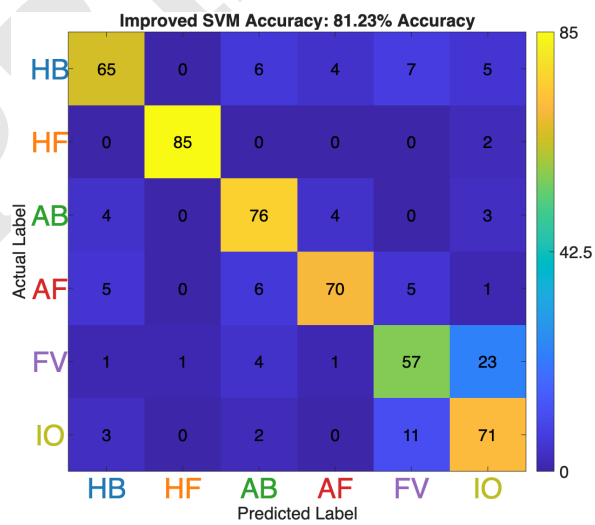

Fig. 13. Confusion matrix for SVM classification with refined grid-search hyperparameters. Following from the previous figure, when a second SVM rbf classification is performed on `S01.mat` data with hyperparameters optimized on a finer grid based on the optimal γ and C from the initial optimization, the overall accuracy improves.

Illustrative analysis 5: Comparison of different RDM constructions. MatClassRSA offers various means for creating RDMs, both from classifier outputs and directly from the original sensor-space data. This analysis provides examples of different RDM constructions.

Script for this analysis:
`illustrative_5_compareRDMs.m`

PREPRINT

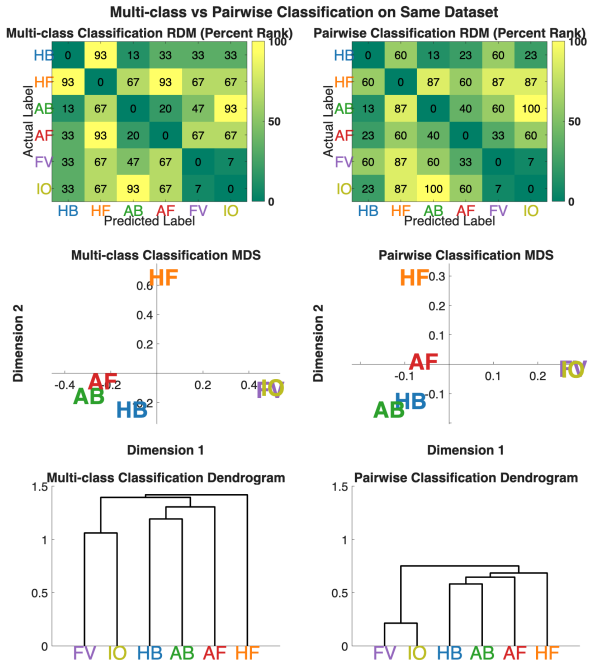

Fig. 14. Six-class multiclass and pairwise accuracy visualizations. Object category data from `S06.mat` underwent 6-class classifications. Top: Percentile-ranked RDMs have a highly similar distance structure. Middle and bottom: Visualizations of RDMs (not percentile ranked) as MDS and dendrograms show similar category structure across the two approaches, with larger values for multiclass.

We first performed both multiclass and pairwise 6-class classification of `S06.mat` object category data using LDA with the `crossValidateMulti()` and `crossValidatePairs()` functions, respectively. Classifier outputs were converted to RDMs. RDMs were percentile ranked for visualization only and plotted using the `plotMatrix()` function, while MDS and dendrogram plots were created from non-percentile-ranked RDMs and visualized using the `plotMDS()` and `plotDendrogram()` functions. As shown in Fig. 14, the percentile-ranked RDMs look similar across the two approaches. The MDS and dendrogram plots also look similar in overall shape, although the distance values are larger for the multiclass approach compared to the pairwise.

We also computed 6-class Pearson and Euclidean RDMs directly from channel 96 of the `S06.mat` data. Data underwent shuffling, noise normalization, and trial averaging prior to being input to the `computePearsonRDM()` and `computeEudlideanRDM()` functions. The RDMs were then visualized using both the `plotMatrix()` and `plotMST()` functions. Fig. 15 shows the resulting RDMs (top), which have been percentile-ranked for visualization. The bottom row shows the minimum spanning trees, with distances reported in their original units. Even with differing ranges of distance values for each RDM technique as shown in the MST plots, these approaches show similar stimulus distance structure (note that MST path lengths do not convey relative path distance values).

Finally, we compared 72-class Pearson and multiclass

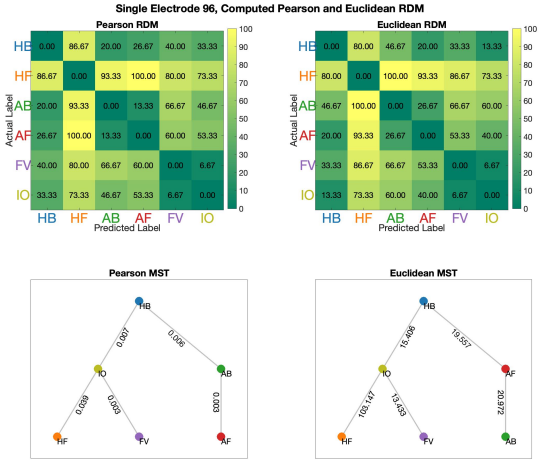

Fig. 15. Six-class Pearson and Euclidean RDMs and MST plots. Six-class RDMs were computed from `S06.mat` data, channel 96. Top: Percentile-ranked RDMs are broadly similar across the two techniques. Bottom: MST plots are also highly similar in structure, even while the specific (non-percentile-ranked) distances occupy different data scales across the two techniques.

classification RDMs and MDS computed from channel 96 of the same `S06.mat` data. Results are summarized in Fig. 16. Contrary to previous 6-class figures, we see more variability between the two approaches: The structure of certain categories is more visually evident in the Pearson than the classification RDM, reflected in blocking along the RDM diagonal particularly for Human Face exemplars (rows 13–24) and Animal Body and Face exemplars (rows 25–48). Human Face and Animal Face exemplars also appear to form tighter clusters in the Pearson MDS plot, as shown by the orange and red dots, respectively. For both MDS plots, Human Face exemplars (orange dots) appear to be relatively well clustered and separated from other categories.

For more information, figures, and interpretations for these analyses, please visit the MatClassRSA User Manual.

Illustrative analysis 6: Customizing figures with stimulus images. The final illustrative analysis demonstrates how the basic MatClassRSA visualization functionalities can be customized and extended—for example to annotate plots with images rather than text labels.

Script for this analysis:
`illustrative_6_figureCustomizations.m`

Trial-averaged data from object category dataset `S01.mat` underwent 6-class classification. In the confusion matrix shown in Fig. 17, rows and columns are labeled with example images from each stimulus category rather than the usual numeric or text labels. Users can leverage this example to customize stimulus labels in their own matrix, MDS, dendrogram, and MST plots.

PREPRINT

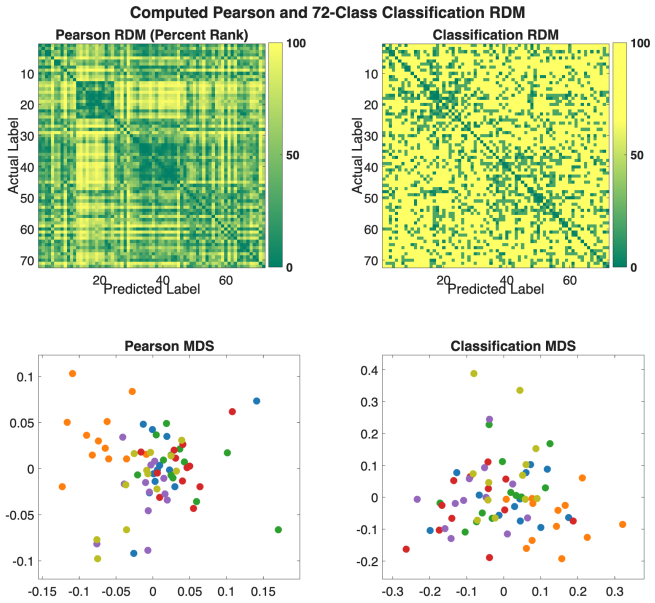

Fig. 16. 72-class Pearson and classification-based RDMs. Channel 96 of the `S06.mat` object category data were used to construct 72-class Pearson and multiclass classification RDMs. These approaches produce markedly different results: By inspection, the Pearson RDM (top) shows a clearer category structure in e.g., rows 13–24 (Human Face) and rows 25–48 (Animal Body/Face) as well as stronger clustering of some categories (colors) in the MDS plot (bottom). Orange dots, which reflect Human Face exemplars, appear to separate from other categories for both RDM construction approaches.

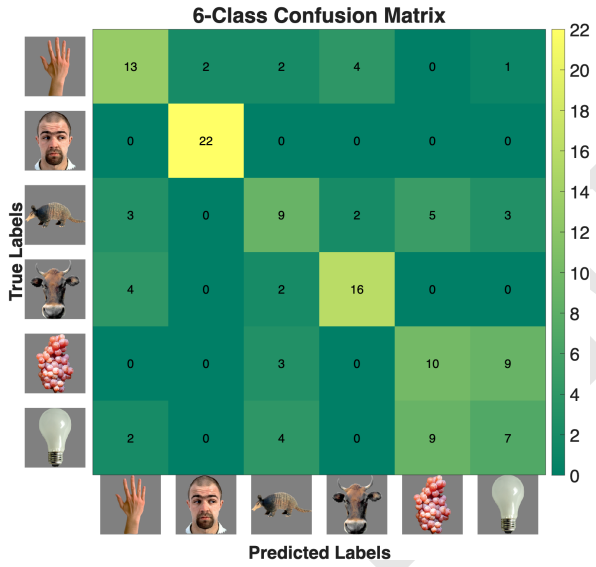

Fig. 17. 6-class confusion matrix customized with stimulus images. The object category data from `S01.mat` underwent 6-class classification. The confusion matrix visualization has been customized with an example image from each of the stimulus categories. The images shown in this figure are from the database of object category images provided by Kriegeskorte et al. [15]; see the description of Illustrative Analysis 0 and footnote 12, above, for more information.

Further elaboration and interpretations for this analysis are provided in the User Manual.

Discussion

We have presented the v2 release of MatClassRSA, a MATLAB toolbox that classifies M/EEG data, performs additional preprocessing and reliability analyses, and also converts M/EEG data and classifier outputs to RDMs and visualizes them. The current release offers a restructured and expanded codebase as well as an updated User Manual and illustrative analyses. Numerous studies have demonstrated the utility of M/EEG classification as a data-driven tool for analyzing high-dimensional data and data with large stimulus sets, and as a means of deriving RDMs that can then be compared with other representations of the stimulus set using RSA. Classifying spatial and/or temporal subsets of the input data in 'searchlight' approaches also provides a data-driven means of identifying task-relevant features of the neural data. The predictive nature of classification in general also highlights potential diagnostic and clinical applications.

MatClassRSA is by no means the only resource for M/EEG classification; a number of other MATLAB toolboxes—such as BCILAB [41], CoSMoMVPA [42], ADAM [43], EEGsig [44], and MVPAlab [45]—as well as Python toolboxes with M/EEG classification functionalities [46–50], decoding toolboxes for other neuroimaging data modalities [51, 52], and more general analytic frameworks for M/EEG [22, 53–55] are freely available, along with tutorials (e.g., [27, 56, 57]). As part of the growing ecosystem of M/EEG decoding toolboxes, MatClassRSA focuses on constructing and visualizing RDMs using classification and other approaches. The expanded offerings of the v2 release—including preprocessing steps [27], reliability calculations, and non-classification RDM constructions—complement the functionalities of other toolboxes. Users are encouraged to explore their options to find the best tool to suit their analysis and workflow needs.

We acknowledge limitations of the current work. For example, the selection of classifiers remains fairly limited; and the toolbox is currently not designed to handle data with missing values, nor is it validated for data with highly imbalanced classes. Interpretation and visualization of classifier weights is not currently implemented, and statistical analyses are limited to permutation testing for single classification runs. By design, MatClassRSA does not include upstream data cleaning or downstream RSA analysis functionalities, and users are encouraged to engage other toolboxes already specializing in those steps [22, 23]. Finally, the lack of a GUI may hinder adoption of MatClassRSA by researchers who are accustomed to working in graphical environments. However, we hope that the extensive documentation and example code provided in the toolbox repository lowers the barrier to entry for *all* researchers, while arguably promoting better reproducibility and shareability of analyses through script- and function-based implementations [42].

Future work on MatClassRSA can involve implementing additional functionalities within the toolbox, or providing bridging functionalities to better connect MatClassRSA function outputs to complementary functions of other toolboxes, such as the creation of 'publication-ready' figures [43] and advanced statistics [42, 43, 45]. While the toolbox has been validated with multiple datasets, we expect

PREPRINT

that real-world usage with new data will bring to light new bugs, edge cases, and feature requests that can be addressed in future versions. Finally, we will be assessing usability of the current release with researchers from various disciplines including Education, Psychology, and Computer Science, to identify how the toolbox can be improved with regard to its implementation and documentation.

ACKNOWLEDGEMENTS

This research was supported by the Patrick Suppes Gift Fund and the Roberta Bowman Denning Fund for Humanities and Technology (BK). The authors gratefully acknowledge Yiran Duan, Steven Losorelli, Vaidehi Natu, Duc T. Nguyen, Ethan Roy, Greta Vilidaite, and Xiaoqian Yan for their valuable feedback and insights contributed while using development versions of the toolbox. We also thank Peter J. Kohler for technical suggestions, Daniel O'Leary and Andero Uusberg for advice on input data specifications, Amilcar Malave for guidance on the code release, and an anonymous reviewer for constructive feedback on a previous version of the toolbox. Finally, we thank Marcos Perreau Guimaraes and Hyung-Suk Kim for their past contributions that are reconceptualized in this toolbox; and Patrick Suppes for championing M/EEG classification in cognitive neuroscience research.

This preprint was prepared using the *HenriquesLab bioRxiv* Overleaf template by Ricardo Henriques.[13]

AUTHOR CONTRIBUTIONS

Designed the toolbox architecture: BCW, AMN, BK. Implemented the code: BCW, RG, NCLK, BK. Validated the code: BCW, RG, NCLK, BK. Documented the toolbox: BCW, RG, NCLK, BK. Created illustrative analyses: RG, BK. Provided theoretical and domain expertise: FR, AMN. Wrote the preprint: BK. Provided feedback and editing: BCW, RG, NCLK, FR, AMN, BK.

DECLARATION ON THE USAGE OF AI

The authors used Azure OpenAI gpt-4o-mini through Stanford University's AI Playground platform[14] for assistance in performing Git and LaTeXoperations as well as for clarifying, debugging, and documenting code files written by co-authors who had become less active on the project. We did not otherwise use any AI tools for developing the code or for writing the user manual or preprint.

References

1. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2 edition, 2009.
2. Christopher R Holdgraf, Jochem W Rieger, Cristiano Micheli, Stephanie Martin, Robert T Knight, and Frederic E Theunissen. Encoding and decoding models in cognitive electrophysiology. *Frontiers in systems neuroscience*, 11:61, 2017.
3. Radoslaw Martin Cichy and Dimitrios Pantazis. Multivariate pattern analysis of MEG and EEG: A comparison of representational structure in time and space. *NeuroImage*, 158: 441–454, 2017.
4. Benjamin Blankertz, Gabriel Curio, and Klaus-Robert Müller. Classifying single trial EEG: Towards brain computer interfacing. In *Advances in Neural Information Processing Systems*, pages 157–164, 2002.
5. Marios G Philiastides and Paul Sajda. Temporal characterization of the neural correlates of perceptual decision making in the human brain. *Cerebral Cortex*, 16(4):509–518, 2006. doi: 10.1093/cercor/bhi130.
6. Changming Wang, Shi Xiong, Xiaoping Hu, Li Yao, and Jiacai Zhang. Combining features from ERP components in single-trial EEG for discriminating four-category visual objects. *Journal of Neural Engineering*, 9(5):056013, 2012.
7. Thomas Carlson, David A. Tovar, Arjen Alink, and Nikolaus Kriegeskorte. Representational dynamics of object vision: The first 1000 ms. *Journal of Vision*, 13(10), 2013. doi: 10.1167/13.10.1.
8. Radoslaw Martin Cichy, Dimitrios Pantazis, and Aude Oliva. Resolving human object recognition in space and time. *Nature neuroscience*, 17(3):455–462, 2014.
9. Blair Kaneshiro, Marcos Perreau Guimaraes, Hyung-Suk Kim, Anthony M. Norcia, and Patrick Suppes. A representational similarity analysis of the dynamics of object processing using single-trial EEG classification. *PLOS ONE*, 10(8):1–27, 08 2015. doi: 10.1371/journal.pone.0135697.
10. Nathan C L Kong, Blair Kaneshiro, Daniel L K Yamins, and Anthony M Norcia. Time-resolved correspondences between deep neural network layers and EEG measurements in object processing. *Vision Research*, 172:27–45, 2020.
11. Siying Xie, Stefanie Hoehl, Merle Moeskops, Ezgi Kayhan, Christian Kliesch, Bert Turtleton, Moritz Köster, and Radoslaw M Cichy. Visual category representations in the infant brain. *Current Biology*, 32(24):5422–5432, 2022.
12. Patrick Suppes, Marcos Perreau Guimaraes, and Dik Kin Wong. Partial orders of similarity differences invariant between EEG-recorded brain and perceptual representations of language. *Neural Computation*, 21(11):3228–3269, 2009. ISSN 08997667.
13. Irina Simanova, Marcel van Gerven, Robert Oostenveld, and Peter Hagoort. Identifying object categories from event-related EEG: Toward decoding of conceptual representations. *PLoS ONE*, 5(12):e14465, 12 2010. doi: 10.1371/journal.pone.0014465.
14. Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, 2(4), 2008. ISSN 1662-5137. doi: 10.3389/neuro.06.004.2008.
15. Nikolaus Kriegeskorte, Marieke Mur, Douglas A Ruff, Roozbeh Kiani, Jerzy Bodurka, Hossein Esteky, Keiji Tanaka, and Peter A Bandettini. Matching categorical object representations in inferior temporal cortex of man and monkey. *Neuron*, 60(6):1126–1141, 2008. ISSN 0896-6273. doi: http://dx.doi.org/10.1016/j.neuron.2008.10.043.
16. Susan G Wardle, Nikolaus Kriegeskorte, Tijl Grootswagers, Seyed-Mahdi Khaligh-Razavi, and Thomas A Carlson. Perceptual similarity of visual patterns predicts dynamic neural activation patterns measured with MEG. *Neuroimage*, 132:59–70, 2016.
17. Yiwen Li, Mingming Zhang, Shuaicheng Liu, and Wenbo Luo. EEG decoding of multidimensional information from emotional faces. *NeuroImage*, 258:119374, 2022.
18. Steven Losorelli, Blair Kaneshiro, Gabriella A Musacchia, Nikolas H Blevins, and Matthew B Fitzgerald. Factors influencing classification of frequency following responses to speech and music stimuli. *Hearing Research*, 398:108101, 2020.
19. Li Su, Elisabeth Fonteneau, William Marslen-Wilson, and Nikolaus Kriegeskorte. Spatiotemporal searchlight representational similarity analysis in EMEG source space. In *2012 International Workshop on Pattern Recognition in NeuroImaging (PRNI)*, pages 97–100, July 2012. doi: 10.1109/PRNI.2012.26.
20. Li Su, Isma Zulfiqar, Fawad Jamshed, Elisabeth Fonteneau, and William Marslen-Wilson. Mapping tonotopic organization in human temporal cortex: representational similarity analysis in EMEG source space. *Frontiers in Neuroscience*, Volume 8 - 2014, 2014. doi: 10.3389/fnins.2014.00368.
21. Ilkka Muukkonen, Kaisu Ölander, Jussi Numminen, and Viljami R Salmela. Spatio-temporal dynamics of face perception. *NeuroImage*, 209:116531, 2020.
22. Arnaud Delorme and Scott Makeig. EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134(1):9–21, 2004. ISSN 0165-0270. doi: http://dx.doi.org/10.1016/j.jneumeth.2003.10.009.
23. Hamed Nili, Cai Wingfield, Alexander Walther, Li Su, William Marslen-Wilson, and Nikolaus Kriegeskorte. A toolbox for representational similarity analysis. *PLOS Computational Biology*, 10(4):1–11, 04 2014. doi: 10.1371/journal.pcbi.1003553.
24. Bernard C Wang, Anthony M Norcia, and Blair Kaneshiro. MatClassRSA: A Matlab toolbox for M/EEG classification and visualization of proximity matrices. *bioRxiv*, 2017. doi: 10.1101/194563.
25. Bernard C Wang, Raymond Gifford, Nathan C L Kong, Feng Ruan, Anthony M Norcia, and Blair Kaneshiro. Example data for MatClassRSA v2 release. In *Stanford digital repository*, 2025. doi: 10.25740/kv831rr3606. https://purl.stanford.edu/kv831rr3606.
26. Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011. ISSN 2157-6904. doi: 10.1145/1961189.1961199.
27. Matthias Guggenmos, Philipp Sterzer, and Radoslaw Martin Cichy. Multivariate pattern analysis for MEG: A comparison of dissimilarity measures. *Neuroimage*, 173:434–447, 2018.
28. Blair Kaneshiro, Steinunn Arnardóttir, Anthony M Norcia, and Patrick Suppes. Object category EEG dataset (OCED). In *Stanford Digital Repository*, 2015. https://purl.stanford.edu/tc919dd5388.
29. Steven Losorelli, Blair Kaneshiro, Gabriella A Musacchia, Karanvir Singh, Nikolas H Blevins, and Matthew B Fitzgerald. Stanford translational auditory research laboratory-frequency following response dataset 1 (STAR-FFR-01). In *Stanford digital repository*, 2019. https://purl.stanford.edu/cp051gh0103.
30. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3): 273–297, 1995. ISSN 0885-6125. doi: 10.1007/BF00994018.
31. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324.
32. Polina Golland and Bruce Fischl. Permutation tests for classification: towards statistical significance in image-based studies. In *Proceedings of IPMI'03: The 18th International Conference on Information Processing in Medical Imaging*, pages 330–341. Springer, 2003.
33. Quentin Noirhomme, Damien Lesenfants, Francisco Gomez, Andrea Soddu, Jessica Schrouff, Gaëtan Garraux, André Luxen, Christophe Phillips, and Steven Laureys. Biased binomial assessment of cross-validated estimation of classification accuracies illustrated in diagnosis predictions. *NeuroImage: Clinical*, 4:687–694, 2014.
34. Roger N Shepard. Stimulus and response generalization: Tests of a model relating generalization to distance in psychological space. *Journal of Experimental Psychology*, 55(6): 509–523, 1958.
35. Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17 (4):401–419, 1952. ISSN 0033-3123. doi: 10.1007/BF02288916.
36. Roger N Shepard. The analysis of proximities: Multidimensional scaling with an unknown distance function. II. *Psychometrika*, 27(3):219–246, 1962. ISSN 0033-3123. doi: 10.1007/BF02289621.
37. Richard Dubes and A K Jain. Clustering methodologies in exploratory data analysis. In Marshall C Yovits, editor, *Advances in Computers*, volume 19, pages 113–228. Elsevier, 1980. doi: http://dx.doi.org/10.1016/S0065-2458(08)60034-0.
38. J C Gower. Some distance properties of latent root and vector methods used in multivariate analysis. *Biometrika*, 53(3–4):325–338, 1966. doi: 10.1093/biomet/53.3-4.325.
39. Roozbeh Kiani, Hossein Esteky, Koorosh Mirpour, and Keiji Tanaka. Object category structure in response patterns of neuronal population in monkey inferior temporal cortex. *Journal of neurophysiology*, 97(6):4296–4309, 2007.
40. Stefan Haufe, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *Neuroimage*, 87:96–110, 2014.
41. Christian Andreas Kothe and Scott Makeig. BCILAB: A platform for brain–computer interface development. *Journal of Neural Engineering*, 10(5):056014, 2013.

[13] https://www.overleaf.com/latex/templates/henriqueslab-biorxiv-template/nyprsybwffws

[14] https://uit.stanford.edu/service/aiplayground

PREPRINT

42. Nikolaas N Oosterhof, Andrew C Connolly, and James V Haxby. CoSMoMVPA: multi-modal multivariate pattern analysis of neuroimaging data in Matlab/GNU Octave. *Frontiers in neuroinformatics*, 10:27, 2016.
43. Johannes J Fahrenfort, Joram Van Driel, Simon Van Gaal, and Christian NL Olivers. From ERPs to MVPA using the amsterdam decoding and modeling toolbox (ADAM). *Frontiers in neuroscience*, 12:368, 2018.
44. Fardin Ghorbani, Soheil Hashemi, Ali Abdolali, and Mohammad Soleimani. EEGsig machine learning-based toolbox for end-to-end EEG signal processing. *arXiv preprint arXiv:2010.12877*, 2020.
45. David López-García, Jose MG Peñalver, Juan M Górriz, and María Ruz. MVPAlab: A machine learning decoding toolbox for multidimensional electroencephalography data. *Computer Methods and Programs in Biomedicine*, 214:106549, 2022.
46. Michael Hanke, Yaroslav O Halchenko, Per B Sederberg, Emanuele Olivetti, Ingo Fründ, Jochem W Rieger, Christoph S Herrmann, James V Haxby, Stephen J Hanson, and Stefan Pollmann. PyMVPA: a unifying approach to the analysis of neuroscientific data. *Frontiers in neuroinformatics*, 3:337, 2009.
47. Mario M Krell, Sirko Straube, Anett Seeland, Hendrik Wöhrle, Johannes Teiwes, Jan H Metzen, Elsa A Kirchner, and Frank Kirchner. pySPACE—a signal processing and classification environment in Python. *Frontiers in Neuroinformatics*, 7:40, 2013.
48. Aurélien Appriou, Léa Pillette, David Trocellier, Dan Dutartre, Andrzej Cichocki, and Fabien Lotte. BioPyC, an open-source python toolbox for offline electroencephalographic and physiological signals classification. *Sensors*, 21(17):5740, 2021.
49. Karl M Kuntzelman, Jacob M Williams, Phui Cheng Lim, Ashok Samal, Prahalada K Rao, and Matthew R Johnson. Deep-learning-based multivariate pattern analysis (dMVPA): A tutorial and a toolbox. *Frontiers in Human Neuroscience*, 15:638052, 2021.
50. Gopal Chandra Jana, Abhishek Karmakar, and Anupam Agrawal. EEG VMAC toolbox: A user-friendly open-source toolbox for EEG signals visualization manipulation analysis and classification. *Procedia Computer Science*, 258:1062–1080, 2025.
51. Jessica Schrouff, Maria J Rosa, Jane M Rondina, Andre F Marquand, Carlton Chu, John Ashburner, Christophe Phillips, Jonas Richiardi, and Janaina Mourao-Miranda. PRoNTo: pattern recognition for neuroimaging toolbox. *Neuroinformatics*, 11(3):319—337, July 2013. ISSN 1539-2791. doi: 10.1007/s12021-013-9178-1.
52. Dominik Grotegerd, Ronny Redlich, Jorge R C Almeida, Mona Riemenschneider, Harald Kugel, Volker Arolt, and Udo Dannlowski. MANIA—a pattern classification toolbox for neuroimaging data. *Neuroinformatics*, 12(3):471–486, 2014. ISSN 1559-0089. doi: 10.1007/s12021-014-9223-8.
53. Robert Oostenveld, Pascal Fries, Eric Maris, and Jan-Mathijs Schoffelen. Fieldtrip: open source software for advanced analysis of meg, eeg, and invasive electrophysiological data. *Computational intelligence and neuroscience*, 2011(1):156869, 2011.
54. Alexandre Gramfort, Martin Luessi, Eric Larson, Denis Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7: 267, 2013. ISSN 1662-453X. doi: 10.3389/fnins.2013.00267.
55. Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen, and Matti S Hämäläinen. MNE software for processing MEG and EEG data. *NeuroImage*, 86:446 – 460, 2014. ISSN 1053-8119. doi: http://dx.doi.org/10.1016/j.neuroimage.2013.10.027.
56. Benjamin Blankertz, Steven Lemm, Matthias Treder, Stefan Haufe, and Klaus-Robert Müller. Single-trial analysis and classification of ERP components—a tutorial. *NeuroImage*, 56(2): 814–825, 2011.
57. Tijl Grootswagers, Susan G Wardle, and Thomas A Carlson. Decoding dynamic brain patterns from evoked responses: A tutorial on multivariate pattern analysis applied to time series neuroimaging data. *Journal of cognitive neuroscience*, 29(4):677–697, 2017.

PREPRINT