NeuroRA: A Python Toolbox of Representational Analysis from Multi-modal Neural Data

Zitong Lu[1,2,3], Yixuan Ku [1,2,4] *
1. Guangdong Provincial Key Laboratory of Social Cognitive Neuroscience and Mental Health, Department of Psychology, Sun Yat-sen University, Guangzhou, China.
2. Peng Cheng Laboratory, Shenzhen, China.
3. Shanghai Key Laboratory of Brain Functional Genomics, Shanghai Changning-ECNU Mental Health Center, School of Psychology and Cognitive Science, East China Normal University, Shanghai, 200062, China.
4. NYU-ECNU Institute of Brain and Cognitive Science, NYU Shanghai and Collaborative Innovation Center for Brain Science, Shanghai, China.

* correspondence: Yixuan Ku, kuyixuan@mail.sysu.edu.cn
ResearcherID: D-4063-2018
ORCID: 0000-0003-2804-5123

Running title: NeuroRA: RSA toolbox in Python

Abstract

In studies of cognitive neuroscience, multivariate pattern analysis (MVPA) is widely used as it offers richer information than traditional univariate analysis. Representational similarity analysis (RSA), as one method of MVPA, has become an effective decoding method based on neural data by calculating the similarity between different representations in the brain under different conditions. Moreover, RSA is suitable for researchers to compare data from different modalities and even bridge data from different species. However, previous toolboxes have been made to fit specific datasets. Here, we develop NeuroRA, a novel and easy-to-use toolbox for representational analysis. Our toolbox aims at conducting cross-modal data analysis from multi-modal neural data (e.g., EEG, MEG, fNIRS, fMRI, and other sources of neruroelectrophysiological data), behavioral data, and computer-simulated data. Compared with previous software packages, our toolbox is more comprehensive and powerful. Using NeuroRA, users can not only calculate the representational dissimilarity matrix (RDM), which reflects the representational similarity among different task conditions and conduct a representational analysis among different RDMs to achieve a cross-modal comparison. Besides, users can calculate neural pattern similarity (NPS), spatiotemporal pattern similarity (STPS), and inter-subject correlation (ISC) with this toolbox. NeuroRA also provides users with functions performing statistical analysis, storage, and visualization of results. We introduce the structure, modules, features, and algorithms of NeuroRA in this paper, as well as examples applying the toolbox in published datasets.

Keywords:

Representational similarity analysis; multivariate pattern analysis; multi-modal; Python; correlation analysis.

Significance statement:

For the last two decades, neuroscience research envisions the prevalence of multivariate pattern analysis, in which representation similarity analysis (RSA) is one of the core methods. As representation bridges computation and implementation in David Marr's model, RSA bridges data from different modalities, including behavior, EEG, MEG, fMRI, et al., and even different species. Our toolbox NeuroRA is developed based on Python and can be applied for multi-modal neural data, as well as behavioral and simulated data. By calculating the representational dissimilarity matrix, neural pattern similarity, spatiotemporal pattern similarity, and inter-subject correlation with NeuroRA, we can assess representation similarities across datasets, subjects, space, and time. Statistical results can also be assessed by user-defined threshold and output to a data format that could be opened in other toolboxes.

Introduction

In recent years, research on brain science based on neural data has shifted from univariate analysis towards multivariate pattern analysis (MVPA) (Norman et al., 2006). In contrast to the former, the latter accounts for the population coding for neurons. The decoding of neural activity can help scientists better understand the encoding process of neurons. As in David Marr's model, representation bridges the gap between a computation goal and implementation machinery (Marr, 1982). Representational similarity analysis (RSA) (Kriegeskorte et al., 2008a) is an effective MVPA method that can successfully describe the relationship between representations of different data modalities, bridging gaps between humans and animals. Therefore, RSA has been rapidly applied in investigating various cognitive functions, including perception (Evans et al., 2015; Henriksson et al., 2019), memory (Xue et al., 2010), language (Chen et al., 2016), and decision-making (Yan et al., 2016).

With technological development in brain science, various neural recording methods have emerged rapidly. Noninvasive methods that investigate brain activity such as electroencephalography (EEG), magnetoencephalography (MEG), functional magnetic resonance imaging (fMRI), and functional near-infrared spectroscopy (fNIRS) have been widely used for basic research. Meanwhile, invasive techniques such as electrocorticography (ECoG), stereo-electro-encephalography (sEEG), and some other neuroelectrophysiological methods have been applied to humans, non-human primates, and other animal species. The interpretation of results across different recording modalities becomes difficult. The RSA method, however, uses a representation dissimilarity matrix (RDM) to bridge data from different modalities. For example, studies have attempted to combine fMRI results with electrophysiological results (Kriegeskorte et al., 2008b; Muukkonen et al., 2020), MEG results with electrophysiological results (Cichy et al., 2014), or behavioral results and fMRI results (Wang et al., 2018). Furthermore, with the rapid development of artificial intelligence (AI) (Jordan and Mitchell, 2015; Kriegeskorte and Golan, 2019), RSA can be used to compare representations in artificial neural networks (ANN) with brain activities (Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014; Güçlü and van Gerven, 2015; Eickenberg et al., 2017; Bonner and Epstein, 2018; Greene and Hansen, 2018; Kuzovkin et al., 2018; Urgen et al., 2019). In summary, RSA is a useful tool to combine the results of behavior and multi-modal neural data, leading to a better understanding of the brain. Even further, it can help us establish a clearer link between the brain and artificial intelligence.

Other existing tools for RSA include a module in PyMVPA (Hanke et al., 2009), a toolbox for RSA by Kriegeskorte (Nili et al., 2014), and an example in

MNE-Python (Gramfort et al., 2013). However, they all have some shortcomings. MNE can only perform RSA for MEG and EEG data in one example. PyMVPA supports only basic functions, such as calculating the correlation coefficient and data conversion. Kriegeskorte's toolbox attached to their paper is designed mainly based on fMRI data, and users need to be proficient in MATLAB (Nili et al., 2014), which makes it difficult for users to apply to other datasets for EEG, MEG, or other types of data sources. We set to develop a comprehensive and universal toolbox for RSA, and Python was chosen as a suitable programming language. Python is a rapidly rising programming language having significant advantages for scientific computing (Sanner, 1999; Koepke, 2011). Because of its strong expansibility, it is more convenient to use Python for implementing a toolbox for representational analysis. NumPy (Van et al., 2011), Scikit-learn (Pedregosa et al., 2013), and other extensions can execute and simplify various basic computing functions. Thus, some researchers select Python to develop toolkits in psychology and neuroscience, such as PsychoPy (Peirce, 2007) for designing psychological experiment programs, MNE-Python for EEG/MEG data analysis, and PyMVPA for utilizing MVPA in data from different modalities.

We have developed a novel and easy-to-use Python toolbox, NeuroRA (neural representational analysis), for comprehensive representation analysis. NeuroRA aims to use powerful computational resources with Python and conduct cross-modal data analyses for various types of neural data (e.g., EEG, MEG, fNIRS, fMRI, and other sources of neuroelectrophysiological data), as well as behavioral data and computer simulation data. In addition to the traditional functions of RSA, NeuroRA also includes specialized parts of representational analysis described in papers published on different research groups. These include neural pattern similarity (NPS) (Haxby, 2001; Cavanagh et al., 2018), spatiotemporal pattern similarity (STPS) (Xue et al., 2010; Lu et al., 2015), and inter-subject correlation (ISC) (Hasson et al., 2004). In the following sections, we detail the structure and function of NeuroRA and further apply it to several open datasets to guide users to use NeuroRA.

Overview of NeuroRA

NeuroRA is an easy-to-use Python toolbox of representational analysis from multi-modal neural data. Users can apply NeuroRA to track the representation and compare representational similarity among different task conditions and modalities.

The structure and features of NeuroRA are illustrated in **Figure 1**. It can analyze all types of neural (including EEG, MEG, fNIRS, fMRI, and other sources of neuroelectrophysiological data) and behavioral data. By utilizing the

powerful computational toolbox in Python, NeuroRA gives users the ability to mine neural data thoroughly and efficiently.

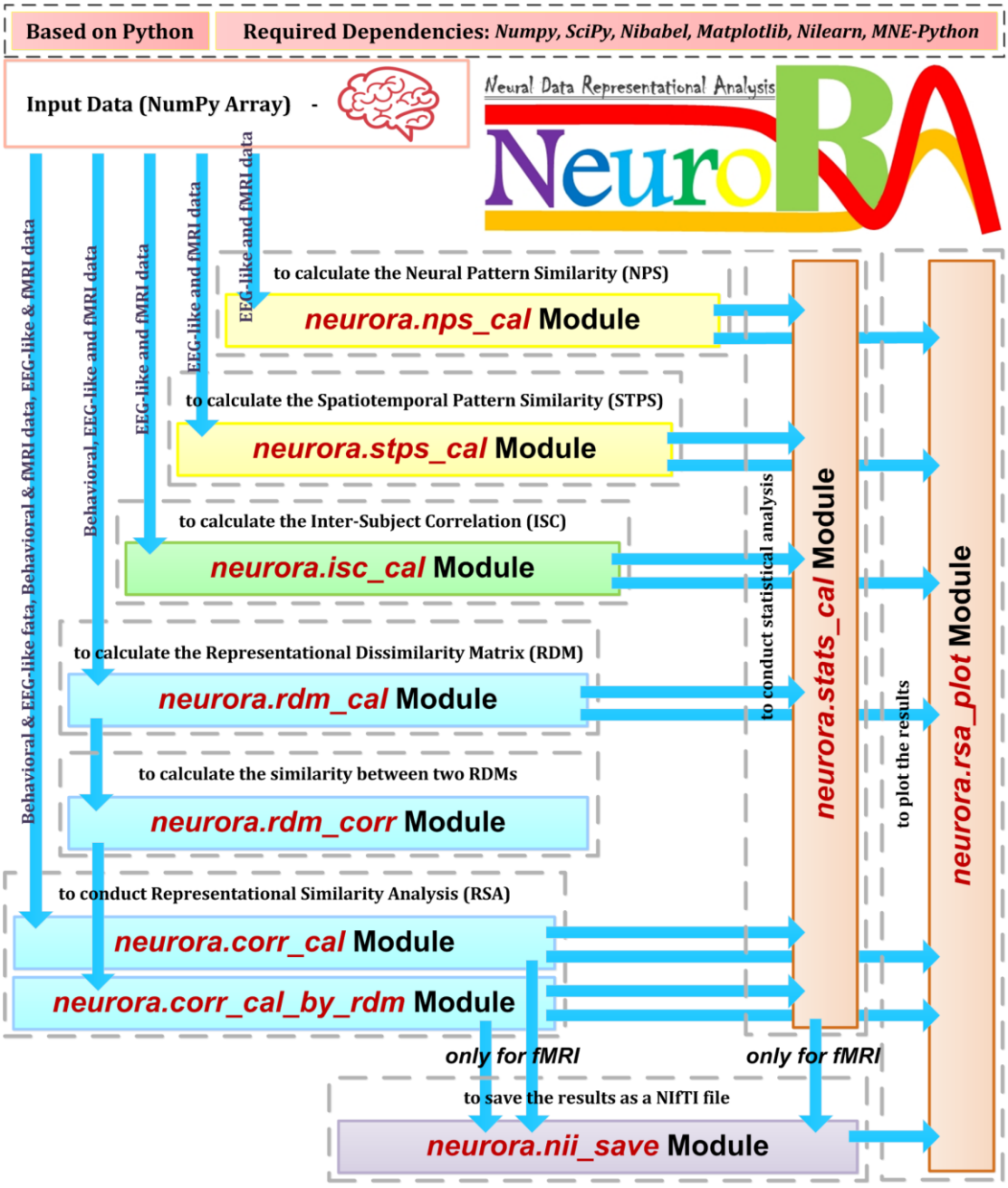

Figure 1 Overview of NeuroRA. NeuroRA is a Python-based toolbox and requires some extension packages, including NumPy, SciPy, Matplotlib, Nilearn, and MNE-Python. It contains several main parts: calculating neural pattern similarity (NPS), spatiotemporal pattern similarity (STPS), inter-subject correlation (ISC), and representation dissimilarity matrix (RDM), comparing representations among different modalities using RDMs, statistical analysis, saving results as a NIfTI file for fMRI data, and plotting the results. Each calculation part corresponds to one to two modules. The blue arrows indicate the feasible data flow.

NeuroRA provides abundant functions. First, NPS module reflects the correlation of brain activities induced under two different conditions. Second, STPS module reflects the representational similarity across different space and time points. Third, ISC module reflects the similarity of brain activities among multiple subjects under the same condition. Fourth, RDM module reflects the representation similarity between different conditions or stimuli with neural data from a given modality. Fifth, NeuroRA performs a correlation analysis between RDMs from different modalities to compare representations across modalities. This procedure can be applied according to different parameters; for example, the calculation can be applied for each subject, for each channel, for each time-point, or a combination of all of them.

In addition to calculating the above values, NeuroRA provides a statistical module to perform statistical analysis based on those values and a visualization module to plot the results, such as RDMs, representational similarities over time, and RSA-results for fMRI. Also, NeuroRA provides a unique approach to save the result of representational analysis back to the widely-used fMRI format NIfTI, generating a file obtained with user-defined output-threshold.

The required packages for NeuroRA include NumPy, SciPy (Virtanen et al., 2020), Matplotlib (Hunter, 2007), Nibabel (Brett et al., 2016), Nilearn, and MNE-Python, which are checked and automatically downloaded by installing NeuroRA. NumPy assists with matrix-based computation. SciPy helps with fundamental statistical analysis. Matplotlib and Nilearn are employed for the plotting functions. NiBabel is used to read and generate NIfTI files. MNE-Python is applied to load example MEG data in the demo. Users can download NeuroRA through only one line of command: *pip install neurora*. The website for our toolbox is https://neurora.github.io/NeuroRA/, and the website for online API documentation is https://neurora.github.io/documentation/. Additionally, GitHub URL for its source code is https://github.com/neurora/NeuroRA.

Data Structures in NeuroRA

The calculations in NeuroRA are all based on multidimensional matrices, including deformation, transposition, decomposition, standardization, addition, and subtraction. The data type in NeuroRA is *ndarray*, an N-dimensional array class of NumPy. Therefore, users first convert their neural data into a matrix (*ndarray* type) as the input of NeuroRA, with information on the different dimensions of the matrix, such as the number of subjects, number of conditions, number of channels, and size of the image (see instructions in the software for details). Here, we give users some feasible methods for data conversion for different kinds of neural data in **Supplemental instructions for data conversion**. The outputs of the functions in NeuroRA are square

matrices with the same dimensions as the input matrix. The input and output data structures of key functions for calculation and statistical analysis in NeuroRA are shown in **Table S1** and **Table S2**.

NeuroRA's Modules and Features

NeuroRA provides various functions to perform the representational analysis. Usually, data must be processed in multi-step ways, and this toolkit highly integrates these intermediate processes, making it easy to implement. In NeuroRA, only a simple function is required to complete the analysis. Users can obtain the required results after a necessary conversion of the data format.

Meanwhile, we attempt to add some adjustable parameters to meet the calculation requirements for different experiments and different modalities of data. Users can flexibly change the input parameters in the function to match their data format and computing goals.

NeuroRA includes the following core modules, and more modules could be added in the future or as requested.

nps_cal: A module to calculate the neural pattern similarity based on neural data.

stps_cal: A module to calculate the spatiotemporal pattern similarity based on neural data.

isc_cal: A module to calculate the inter-subject correlation based on neural data.

rdm_cal: A module to calculate RDMs based on multi-modal neural data.

rdm_corr: A module to calculate the correlation coefficient between two RDMs, based on different algorithms, including Pearson correlation, Spearman correlation, Kendall's tau correlation, cosine similarity, and Euclidean distance.

corr_cal_by_rdm: A module to calculate the representational similarities among the RDMs under different modes.

corr_cal: A module to conduct a one-step RSA between two different modes of data.

nii_save: A module to save the representational analysis results in a .nii file for fMRI.

stats_cal: A module to calculate the statistical results.

rsa_plot: A module to plot the results from the representational analysis. It contains the functions of plotting the RDM, plotting the graphs or hotmaps with results from the representational analysis by time sequence based on EEG or EEG-like (such as MEG) data, plotting the results of fMRI representational analysis (montage images and surface images).

Representational Similarity Analysis using NeuroRA

RSA has gradually become a popular method to explore information coding in the brain and computational models. Comparing whole dissimilarities among all task conditions in RDM, RSA becomes an effective approach to track the multidimensional representation among task conditions. On the one hand, researchers can construct hypothesis-based RDM for different conditions, then compare these theoretical models with RDMs from real neural activities to calculate how similar they are (Alfred et al., 2018; Feng et al., 2018; Hall-McMaster et al., 2019; Yokoi and Diedrichsen, 2019; Etzel et al., 2020). As a result, they can infer the information is coded in the brain. On the other hand, researchers can compare differences and similarities among multi-modal data by comparing the distance or correlation among RDMs computed using different data sources (Kriegeskorte et al., 2008b; Cichy et al., 2014; Stolier and Freeman, 2016; Muukkonen et al., 2020). This cross-modal calculation has been increasingly used in comparing brain activities and deep neural networks during object processing (Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014; Güçlü and van Gerven, 2015; Eickenberg et al., 2017; Bonner and Epstein, 2018; Greene and Hansen, 2018; Kuzovkin et al., 2018; Urgen et al., 2019).

- **Calculate one RDM or multiple RDMs**

Constructing an RDM is a typical approach for comparing representations in neural data. By extracting data from two different conditions and calculating the correlations between them, we will obtain the similarity between the two representations under the two conditions. Subtract the obtained similarity index from 1 and get the values of the dissimilarity index in RDM (**Figure 2**). In **Figure 2**, different grating stimuli were observed to produce different neural responses, and the value in RDM presented the dissimilarity of neural activities between two different stimuli. As shown in the figure, the closer the two grating orientations were, the lower the dissimilarity. In a typical object recognition experiment, humans and monkeys were allowed to watch the same sets of visual stimuli (Kriegeskorte, 2008). Researchers calculated the humans' RDM based on fMRI data and the monkeys' RDM based on electrophysiological data.

The results indicated that the neural patterns in RDMs were similar when humans and monkeys observed stimuli that belonged to the same category (animate or inanimate).

Figure 2 Schematic diagram for calculating the RDM. Different data can be obtained under different conditions. The value of the point [*i*, *j*] in RDM is obtained by calculating the dissimilarity (1-correlation coefficient *r*) between the data under condition *i* and that under condition *j*.

Assuming that the measured data from a certain condition under a total of *n* experimental conditions are denoted as $d_1, d_2, \ldots, d_n$, then the following RDM of $n \times n$ can be calculated by the corresponding function under the rdm_cal module from our toolkit:

$$RDM = \begin{pmatrix} D_{11} & D_{12} & \cdots & D_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ D_{n1} & D_{n2} & \cdots & D_{nn} \end{pmatrix}$$

where D_{ij} denotes the dissimilarity between the data under condition *i* and that under condition *j*. The dissimilarity (D_{ij}) is calculated as follows:

$$D_{ij} = 1 - similarity(d_i, d_j)$$

When computing the RDM, multiple measures are provided in NeuroRA, including correlation distance (based on Pearson correlation), Euclidean distance, and Mahalanobis distance. All functions in *neurora.rdm_cal* module has a parameter named *method,* which can be set to change the measure you want (default is for computing based on correlation distance). The application of calculating RDMs is not restricted. NeuroRA can perform computations based on multiple modal neural data from behavioral data to brain imaging data (**Figure 3**).

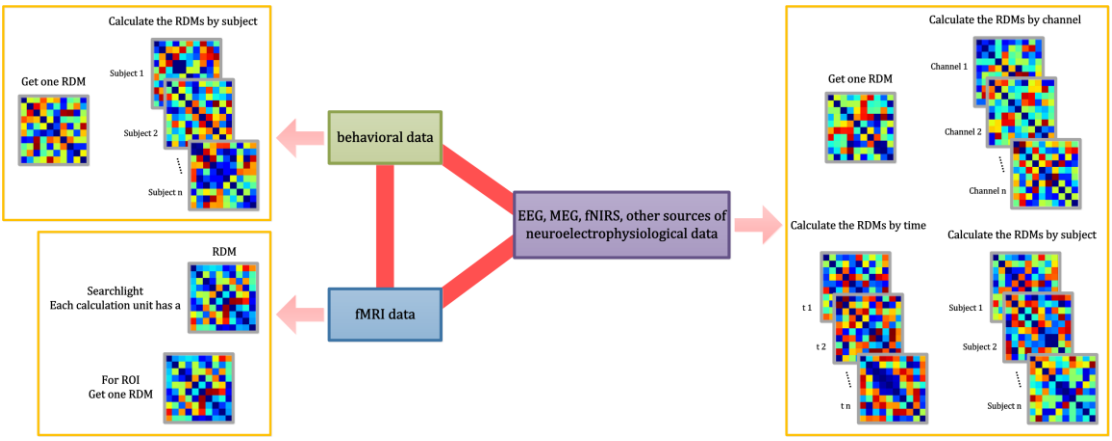

Figure 3 RDM calculations implemented in NeuroRA. NeuroRA is capable of calculating an RDM using data from different modes, including behavior, fMRI, EEG, MEG, fNIRS, and other sources of neuroelectrophysiological data. The red bold lines denote the ability to perform calculations between two modes. The pink arrow denotes the alternative calculation methods of the corresponding mode.

In certain cases, researchers must calculate RDM separately for each participant, or they may calculate RDM independently for each channel or each time point (Henriksson et al., 2019; Hall-McMaster et al., 2019). We resolve these issues in NeuroRA by providing several input parameters in the functions that allow users to make the corresponding changes to get one RDM or multiple RDMs by different subjects or channels or time-windows or searchlight (for fMRI) or specific ROI (for fMRI) (**Figure 3**). Users can change the calculation parameters according to their requirements, and consequently, the corresponding output formats change. Detailed instruction of the shape of the input, the parameter settings with calculation implementation, the corresponding shape of the output, and recommended next steps can be seen in Table **S1**.

➢ Representational analysis among different RDMs

Analysis between two RDMs

NeuroRA provides a convenient way to calculate cross-modal similarity by computing the similarities between two RDMs from different modalities. We offer several solutions to calculate the similarity (or correlation coefficient), including Pearson correlation, Spearman correlation, Kendall's tau correlation, cosine similarity, and Euclidean distance. Users can freely change parameters to select different computing methods.

For the calculations, we first reshape the square matrices into vectors and then calculate similarities (**Figure 4**). Previous studies calculated the

correlation coefficient between two RDMs using the diagonal values, making the result deceptively high (Ritchie et al., 2017). We avoid this by removing the diagonal values and include only half of the matrix to reduce the duplication, as the upper and lower halves of the RDM are identical (**Figure 4**).

Furthermore, NeuroRA provides a permutation test to determine whether the two RDMs are related. The permutation test is based on the random shuffling of data and is suitable for data with a small sample size (Efron and Tibshirani, 1994). We first shuffle the values in the two RDMs and re-calculate the similarity matrix between the two RDMs. By repeating this procedure 5000 times (the number of iterations here can be defined by users), we get the final *p*-values from this permutation distribution.

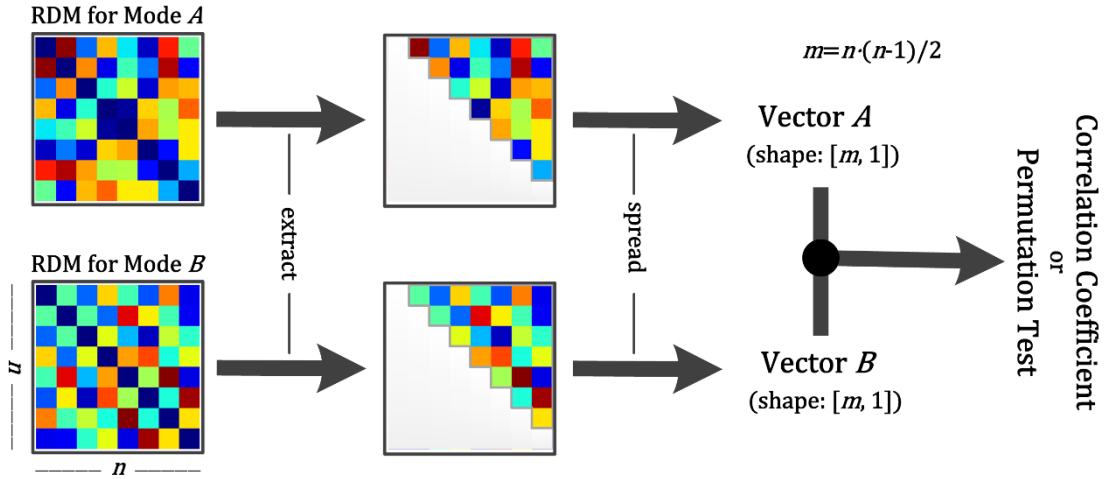

Figure 4 Schematic diagram for calculation between two RDMs. Step 1: Obtain two RDMs from different modal data. Step 2: Extract the points of the upper diagonal (within the grey line). Step 3: Spread them as vectors. Step 4: Calculate the correlation coefficient or conduct a permutation test between two vectors. The former returns the correlation coefficient and the *p*-value, and the latter returns only the *p*-value.

Analysis among multiple RDMs

NeuroRA can also perform calculations based on multiple RDMs, rather than only two RDMs corresponding to two modalities. Consequently, we can expand it to conditions with multiple RDMs from different modalities. For instance, when you obtain a behavioral RDM from behavioral data and wish to compare it with other modal data, a problem may arise as more than one RDM can be obtained based on other modal data, such as EEG or fMRI. Our toolbox provides "searchlight" computation to perform correlation analysis between RDM from one mode (behavior, or any of neural data) and RDM from other modes (each brain region, time interval, or others) one by one (**Figure 3**). For example, calculations based on EEG data can obtain one RDM per channel or time

interval or both (**Figure 5**). **Table 1** is a script example for using NeuroRA to calculate the similarities between behavioral RDM and EEG RDMs per channel by time sequence. Another simple example is when users want to see which brain regions are highly correlated with the behavioral performance or a specific coding model, they can get one behavioral or model RDM based on behavioral response time or accuracy, and they may also get many fMRI RDMs from different regions. Users can put these two kinds of RDMs (behavioral or model RDM and fMRI RDMs) into our function, and they will get results showing the regions that are highly correlated with behavioral or model patterns based on thresholds of significance (*p*-value) or correlation values set by users. (**Table 3**, more details on fMRI calculation are described in the next section). These convenient functions of ergodic computation cover the vast majority of cross-modal research needs.

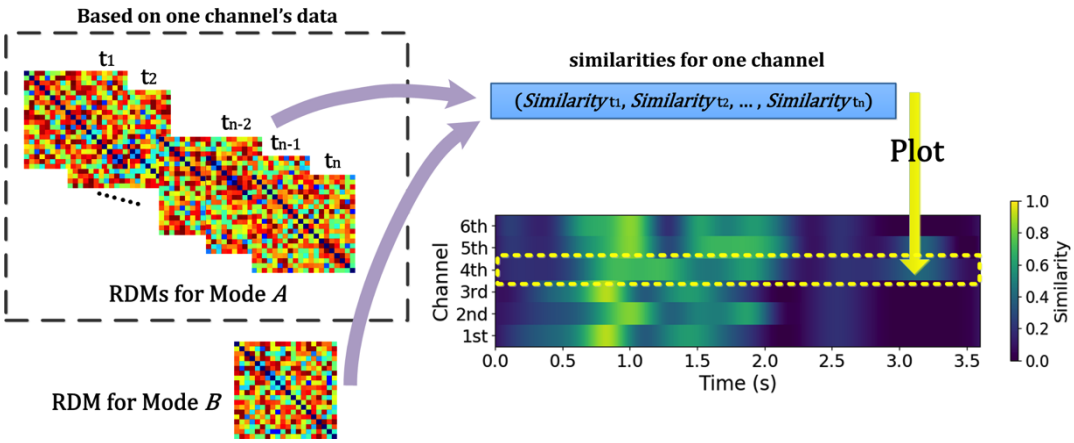

Figure 5 Schematic diagram for calculating similarities between RDM from different modes across time and channel for EEG and EEG-like (such as MEG or sEEG) data. NeuroRA calculates the similarities between RDMs for mode A (EEG and EEG-like data) and one RDM for mode B (such as behavior). Such calculation can be performed across each time-window and each channel. Each value in time-channel result-image (bottom right) corresponds to a similarity index (for example, the Pearson correlation) between RDMs from two Modes.

Table 1 Scripts of representational analysis between behavioral data and EEG data for each channel in NeuroRA. Users can input data from different modes and obtain the correlation between the results of the two modes. If users want to conduct the calculation for each time-windows, they can set the parameters: *time_opt*, *time_win* & *time_step* in function

eegRDM and *bhvANDeeg_corr*.

Scheme 1

```
from neurora.rdm_cal import bhvRDM, eegRDM
from neurora.corr_cal_by_rdm import rdms_corr

# calculate the behavioral RDM for each subject
# the shape of bhv_data should be [n_conditions, n_subjects, n_trials]
# the shape of bhv_rdms will be [n_subjects, n_conditions, n_conditions]
bhv_rdms = bhvRDM(bhv_data, sub_opt=1)

# calculate the eeg RDMs for each channel & each subject
# the shape of eeg_data should be [n_conditions, n_subjects, n_trials, n_channels, n_times]
# the shape of eeg_rdms will be [n_subjects, n_channels, n_conditions, n_conditions]
eeg_rdms = eegRDM(eeg_data, sub_opt=1, chl_opt=1)

# initialize the correlation coefficients
corrs = np.zeros([n_subjects, n_channels, 2], dtype=np.float)

# calculate the correlation coefficients between behavioral RDM and eeg RDMs
# the shape of corrs is [n_subjects, n_channels, 2], 2 represents a r-value & a p-value
for sub in range(n_subjects):
    corrs[sub] = rdm_corr(bhv_rdms[sub], eeg_rdms[sub])
```

Scheme 2

```
from neurora.corr_cal import bhvANDeeg_corr

# calculate the correlation coefficients between behavioral RDM and eeg RDMs
corrs = bhvANDeeg_corr(bhv_data, eeg_data, sub_opt=1, chl_opt=1)
```

To simplify users' experience, our toolbox offers a one-step option between different modes (**Table 1 Scheme 2** is a one-step example for calculating a similarity index between behavior and EEG). Users can input data from two modalities, and the toolbox will directly return the final results of representation analysis. It is very convenient and efficient when users do not need to obtain the RDMs in the intermediate stages. Thus, users can use two modules, *corr_cal* and *corr_cal_by_rdm*, to calculate the representational similarity between two different modalities. The former module provides the calculation based on data from two different modalities. The later module provides the calculation based on RDMs after previous computing from two modalities' data. In both modules for calculating cross-modal similarity, users can set different parameters to meet the requirements under different conditions (calculate for each channel, etc.). More detailed instructions of the shape of the input, the parameter settings with calculation implementation, the corresponding shape of the output, and recommended next steps about these modules are shown in **Table S1**.

- **Representational Analysis for fMRI**

fMRI is a largely used method in cognitive neuroscience. In the RSA of fMRI data (Johnson et al., 2005; Poldrack, 2012; Rosen, 2012; Lawrence et al., 2019), researchers typically wish to calculate RDMs for different brain regions. In NeuroRA, users can conduct representational analysis using ROIs or searchlight across the whole brain (**Figure 6**).

ROI-based computation

For ROI-based computation, users are required to input both fMRI data and a 3-D mask matrix whose size should be consistent with the size of the fMRI image corresponding to fMRI data. The valid voxels which belong to ROI are extracted, and different activities under different conditions of these voxels are spread out as vectors. Then the ROI-based RDM can be calculated by computing the dissimilarities among these vectors. Finally, we can calculate the similarity between this ROI-based RDM and the RDM for another modality. Steps for ROI-based computation with corresponding functions in NeuroRA are shown in **Figure 6a**.

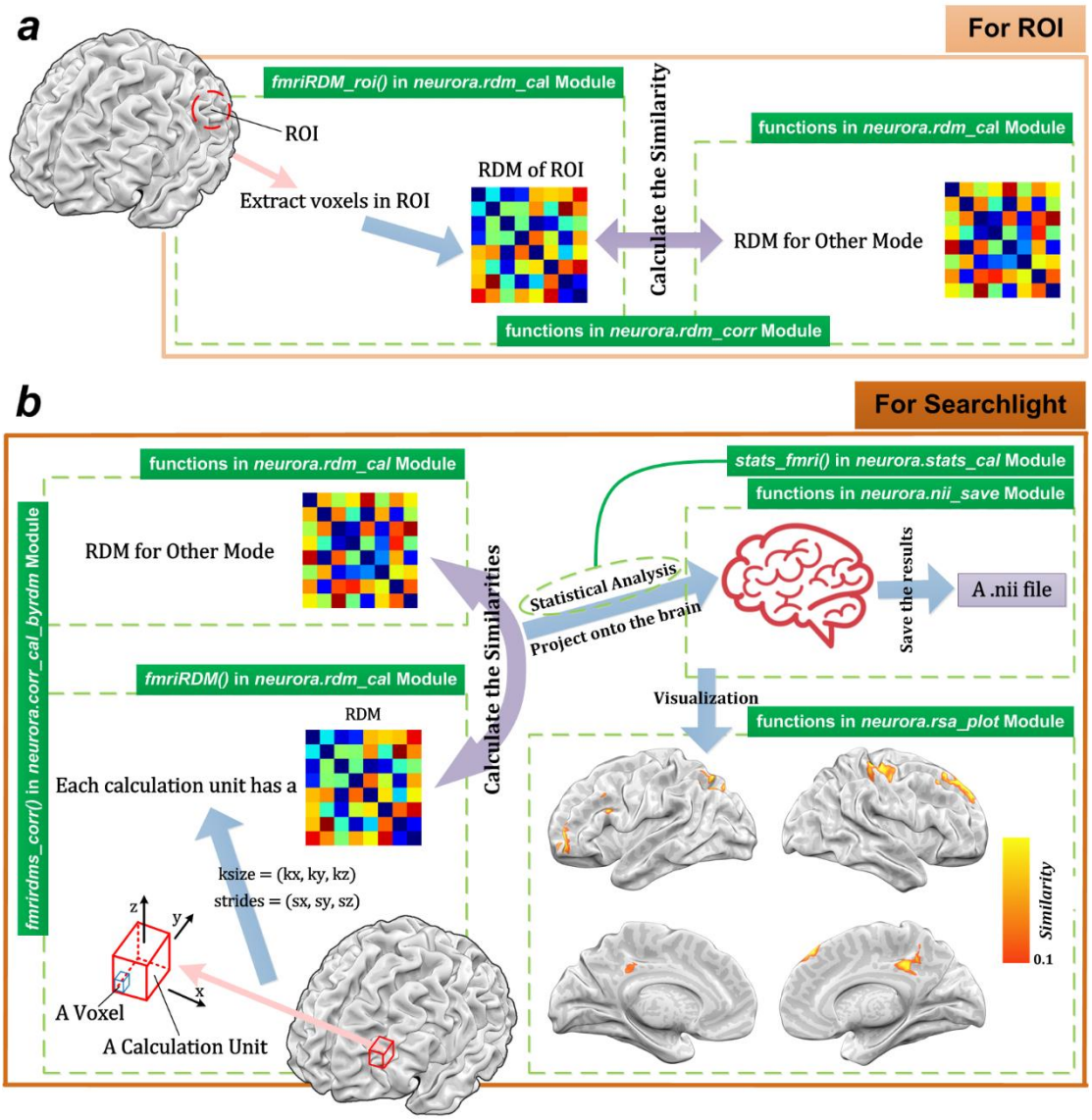

Figure 6 Schematic diagram for representational analysis for fMRI data using NeuroRA. (a) The calculating process for ROI-based analysis. For each ROI, users can calculate the RDM based on the voxels in ROI and get the similarity between ROI RDM and the RDM for other modes. (b) The calculating process for searchlight-based analysis. For each searchlight step, users define the size and strides of the calculation unit. After computations between the RDMs within the searchlight blob for fMRI and the RDM for other modes (e.g., behavioral data, computer-simulated data), a NIfTI file can be obtained. At the bottom right is a demo of the resulting NIfTI file drawn with NeuroELF (http://neuroelf.net), and color-coded regions indicate the strength of representation similarity between two modes. The green text on the green indicates which function to use for the corresponding step.

Searchlight-based computation

Searchlight related functions in NeuroRA provide rich parameters for user customization. For each searchlight step, users can customize the size of the

basic calculation unit $[k_x, k_y, k_z]$. Each *k* indicates the number of voxels along the corresponding axis. The strides between different calculation unit must be decided as $[s_x, s_y, s_z]$. The strides refer to how far the calculation unit is moved before another computation is made. Each *s* indicates how many voxels exist between two adjacent calculation units along the corresponding axis. For the fMRI data of size [*X*, *Y*, *Z*], the kernel size is usually set to be more than one voxel so that each voxel can exist in multiple kernels (calculation units). Therefore, *N* computations are required here:

$$N = (\lfloor (X - k_x)/s_x \rfloor + 1) \times (\lfloor (Y - k_y)/s_y \rfloor + 1) \times (\lfloor (Z - k_z)/s_z \rfloor + 1)$$

This implies that *N* RDMs must be calculated, which are each related to the corresponding calculation unit. After obtaining searchlight RDMs, users can calculate the similarities between fMRI and other modes. In NeuroRA, the final correlation coefficient of one voxel is the mean value of the correlation coefficients calculated by all kernels that contain this voxel.

Table 2 Script of searchlight representational analysis between fMRI data and a coding model in NeuroRA. The calculation parameters of fMRI data are *ksize=[3, 3, 3]* and *strides=[1, 1, 1].* Users can just input different data and obtain the correlation results between two modes.

```
from neurora.rdm_cal import fmriRDM
from neurora.corr_cal_by_rdm import fmrirdms_corr
import numpy as np

# calculate the searchlight fMRI RDMs for each subject
# the shape of fmri_data should be [n_conditions, n_subjects, nx, ny, nz]
# nx, ny, nz represent the size of fMRI-img
# here, the size of calculation unit is [3, 3, 3] and the strides for calculating is [1, 1, 1]
# the shape of fmri_rdms will be [n_subjects, n_x, n_y, n_z]
# n_x, n_y, n_z represent the number of calculation units for searchlight along the x, y, z axis.
fmri_rdms = fmriRDM(fmri_data, ksize=[3, 3, 3], strides=[1, 1, 1], sub_opt=1)

# initialize the correlation coefficients
corrs = np.zeros([n_subjects, n_x, n_y, n_z, 2], dtype=np.float)

# calculate the correlation coefficients between searchlight fMRI RDMs and a model RDM
# the shape of model_rdm should be [n_conditions, n_conditions]
# the shape of corrs will be [n_subjetcs, n_x, n_y, n_z, 2], 2 represents a r-value & a p-value
for sub in range(n_subjects):
    corrs[sub] = fmrirdms_corr(model_rdm, fmri_rdms[sub])
```

Figure 6b shows the steps for searchlight-based computation with corresponding functions in NeuroRA. **Table 2** is a script demo to understand

how to conduct a searchlight-based analysis for fMRI data. We could first calculate the fMRI RDMs within each searchlight blob and then obtain similarities between fMRI RDMs and a behavioral RDM or a coding model RDM, which is constructed based on the hypothesis all over the whole brain. In a hypothesis-based RDM, values corresponding to the same condition have the highest similarity, and values corresponding to different conditions have a low similarity.

Save results as a NIfTI file

NeuroRA provides two functions in *nii_save* module, *corr_save_nii*() and *stats_save_nii*(), to save the similarity result or the statistical result as a NIfTI file with thresholding parameters as well. These two functions are used similarly. The former function is used for saving the results of *r*-values after calculating the similarities between fMRI mode and another mode. The latter function is used for saving the results of *t*-values after statistical analysis. **Table 3** is a script to help users understand how to use *corr_save_nii*() to save the similarity results as a NIfTI file. Users can set certain thresholds for *p*-values, *r*-values (only in *corr_save_nii*() function) or *t*-values (only in *stats_save_nii*() function). Also, users can select Family-Wise-Error (FWE) or False-Discovery-Rate (FDR) correction methods to control for multiple comparisons across the whole brain. Furthermore, users can choose whether to smooth the results, whether to plot automatically, etc. For example, if the threshold for *p*-value is set as 0.05, the final NIfTI file returned will be filtered with $p < 0.05$, and all voxels with $p >= 0.05$ will be set as 0.

Table 3 Script of saving the calculation results as a NIfTI file for fMRI data. Users can get the correlation results based on the script in **Table 2**. The NIfTI file can be obtained by entering some necessary parameters.

```
from neurora.nii_save import corr_save_nii

# corrs represents the similarities (correlation coefficients) between fMRI and other mode
# the shape of corrs should be [n_x, n_y, n_z, 2]
# filename represents the filename of the result .nii file
# affine represents the information of the fMRI-image array data in a reference space
# here, the size of fMRI-image is [60, 60, 60], the size of calculation unit is [3, 3, 3] and the
# strides for calculating is [1, 1, 1]
filename = "demo_result.nii"
corr_save_nii(corrs, filename, affine, size=[60, 60, 60], size=[60, 60, 60], ksize=[3, 3, 3],
    strides=[1, 1, 1], p=0.05, correct_method='FDR')

# The output is an [60, 60, 60] NumPy-array
# And a .nii file named 'demo-results.nii' has been generated
```

Other Representational Analysis

In addition to RSA, users can conduct the analysis of NPS, STPS, and ISC with NeuroRA. Detailed implementation of these analysis methods can be seen in **Supplementary information for the implementation of analysis methods**. Our toolkits have separate modules to conduct these calculations (**Table 4**). Just like RSA from multiple modalities, the calculations for these other representational analysis methods support EEG-like data as well as fMRI data. Users can calculate the results for each channel or region, each time-window from a time series, each ROI or searchlight blobs (for fMRI) as they wish by selecting different functions and setting specific parameters. These calculations are used in a similar way to calculate RDM or RSA, as described in the above sections. In detail, **Table S1** shows the shape of the input, the parameter settings with calculation implementation, the corresponding shape of the output, and recommended next steps in the analysis. Additionally, users can use *help*() (a built-in function in Python) to see and understand the detailed description of the purpose of the specific function or module.

Table 4 Modules and functions for NPS, STPS, ISC in NeuroRA.

Analysis Method	Module for Computing	Functions for Statistical Analysis
NPS	*neurora.nps_cal* module	For EEG-like: *neurora.stats_cal.stats*() For fMRI: *neurora.stats_cal.stats_fmri*()
STPS	*neurora.stps_cal* module	For EEG-like: *neurora.stats_cal.stats_stps*() For fMRI: *neurora.stats_cal.stats_stpsfmri*()
ISC	*neurora.isc_cal* module	For EEG-like: *bneurora.stats_cal.stats*() For fMRI like: *neurora.stats_iscfmri*()

Statistical Analysis

NeuroRA provides functions for statistical analysis based on the representational analysis results. The inputs are the similarity maps for each subject, which can be obtained by functions in calculation modules (*corr_cal*, *corr_cal_by_rdm*, *nps_cal*, *stps_cal,* and *isc_cal* modules), and the output will be the statistical results (a *t*-value & *p*-value map) (**Table 5**). The output from the functions of calculation modules always includes an *r*-value map and a *p*-value map. Although only the *r*-value map is used for subject-level statistical analysis, users can directly input the output of functions in calculation modules

as the input of functions in *stats_cal* module for convenience.

In this part, the correlation coefficients calculated by calculation modules are tested against zero for significance. Besides, we add a permutation test to all processes of statistical analysis. This means the statistical significance could be assessed through a permutation test by randomly shuffling the data and calculated the results for many iterations (for example, 5000) to draw a distribution. Real data exceeding 95% of the distribution are regarded as significant. **Table 5** is a script to show how to use *stats_cal* module to conduct statistical analysis for RSA results from different modes. Users can independently choose to use the permutation test or not and change the iteration number by set parameters in related functions.

Table 5 Example of statistical analysis for channel-time based EEG RSA calculation and searchlight fMRI RSA calculation.

Type of Calculation	Example Script

channel-time based EEG-like calculation	from neurora.corr_cal import bhvANDeeg_corr
	from neurora.stats_cal import stats
	
	# calculate the correlation coefficients between behavioral data and EEG data
	corrs=bhvANDeeg_corr(bhv_data, eeg_data, sub_opt=1, chl_opt=1, time_opt=1)
	
	# the shape of corrs should be [n_subs, n_chls, n_ts, 2]
	stats(corrs, permutation=True, iter=1000)
	
	# The output is an [n_chls, n_ts, 2] NumPy-array
	# 2 represents a *t*-value and a *p*-value

searchlight fMRI calculation	from neurora.corr_cal import bhvANDfmri_corr
	from neurora.stats_cal import stats_fmri
	
	# calculate the correlation coefficients between behavioral data and fMRI data
	corrs=bhvANDfmri_corr(bhv_data, eeg_data, sub_opt=1, chl_opt=1, time_opt=1)
	
	# the shape of corrs should be [n_subs, n_x, n_y, n_z, 2]
	stats_fmri(corrs, permutation=True, iter=10000)
	
	# The output is an [n_x, n_y, n_z, 2] NumPy-array

Visualization of Results

NeuroRA provides several functions to visualize the results in *rsa_plot* module. Some typical features are shown in **Figure 7**.

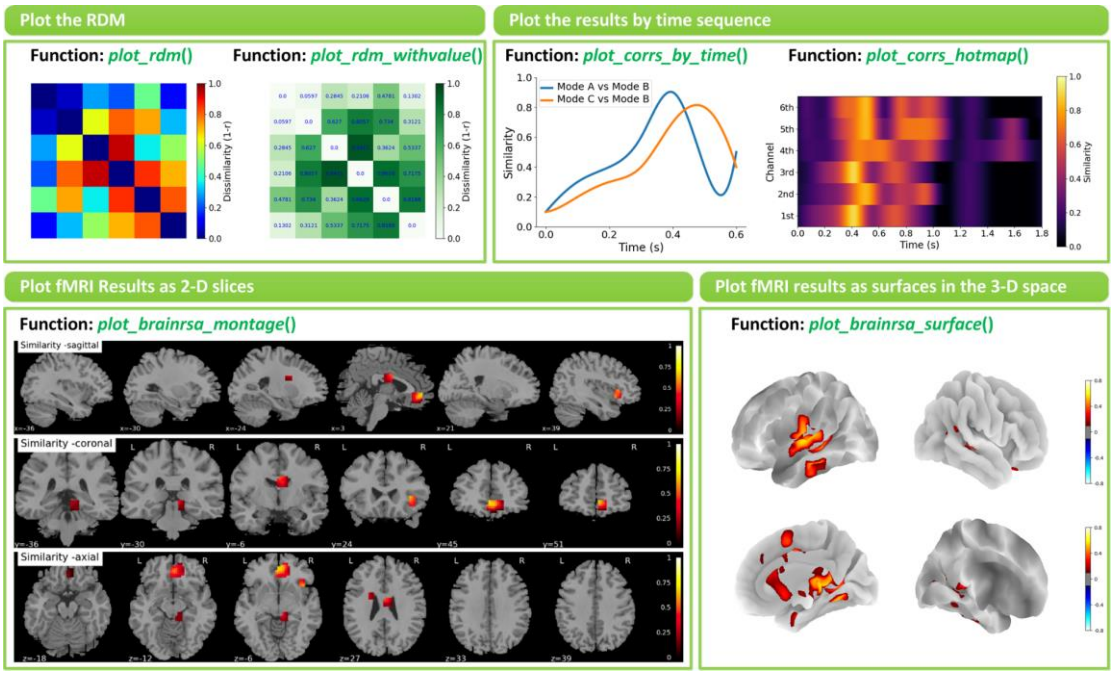

Figure 7 Examples of visualizations implemented in NeuroRA. Left-top: Plot the RDM by function *plot_rdm()* and *plot_rdm_withvalue*(). Right-top: Plot the results by time sequence by function *plot_corrs_by_time*() and *plot_corrs_by_hotmap*(). Left-down: Plot fMRI results as 2-D slices by function *plot_brainrsa_montage*(). Right-down: Plot fMRI results as surface in the 3-D space by function *plot_brainrsa_surface*().

The basic option is to visualize RDMs by function *plot_rdm*() or *plot_rdm_withvalue*(). The more advanced option for EEG-like data is to visualize the results across different time points. On the one hand, users can use specific functions, *plot_corrs_by_time*() and *plot_tbytsim_withstats*(), to plot the curve. On the other hand, users can use specific functions, *plot_corrs_hotmap*(), *plot_corrs_hotmap_stats*() (for *r*-values), *plot_stats_hotmap*() (for *t*-values) and *plot_nps_hotmap*() (for NPS), to plot the hotmap. Also, NeuroRA has options for plotting fMRI results on a brain. Users can use functions such as *plot_brainrsa_glass*(), *plot_brainrsa_montage*() and *plot_brainrsa_regions*() to plot fMRI results as 2-D slices, and use *plot_brainrsa_surface*() to plot results as surfaces in the 3-D space. Feature and applicability of functions in *rsa_plot* module are shown in **Table 6**. The implementation of visualization requires the Pyplot module in the Matplotlib and nilearn package.

Table 6 Feature and applicability of functions for plotting results in NeuroRA.

Function	Feature and Applicability

for RDM	*plot_rdm*()	Plot the RDM -The input should be an RDM (N_conditions × N_conditions).
	plot_rdm_withvallue()	Plot the RDM with values -The input should be an RDM (N_conditions × N_conditions).
for EEG-like	*plot_corrs_by_time*()	Plot the correlation coefficients for different conditions by time sequence -The input should be a matrix (N_conditions×N_time-points) of correlation coefficients.
	plot_tbytsim_withstats()	Plot the similarity averaging all subjects by time sequence with statistical results -The input should be a matrix (N_subs × N_time-points) of similarities.
	plot_corrs_hotmap()	Plot the hotmap of correlation coefficients for channels/regions by time sequence -The input should be a matrix (N_channels×N_time-points) of correlation coefficients.
	plot_corrs_hotmap_stats()	Plot the hotmap of correlation coefficients for channels/regions by time sequence with the significant outline -The input should be a matrix (N_channels×N_time-points) of correlation coefficients and a matrix (N_channels × N_time-points × 2) of *t*-values and *p*-values.
	plot_stats_hotmap()	Plot the hotmap of statistical results for channels/regions by time sequence -The input should be a matrix (N_channels×N_time-points×2) of *t*-values and *p*-values.
	plot_nps_hotmap()	Plot the hotmap of NPS for channels/regions by time sequence -The input should be a matrix (N_channels×N_time-points) of similarities.
for fMRI	*plot_brainrsa_glass*()	Plot the 2-D projection of the RSA-results -The input should be the .nii file generated by functions in *neurora.nii_save* module
	plot_brainrsa_montage()	Plot the RSA-results by different cuts -The input should be the .nii file generated by functions in *neurora.nii_save* module
	plot_brainrsa_regions()	Plot the high-correlation regions of RSA-results by three cuts (frontal, axial, and lateral) -The input should be the .nii file generated by functions in *neurora.nii_save* module
	plot_brainrsa_surface()	Plot the RSA-results into a brain surface -The input should be the .nii file generated by functions in *neurora.nii_save* module

We also provide several code demos in NeuroRA on the publicly available datasets. The first demo is based on visual-92-categories-task MEG datasets (Cichy et al., 2014). We extracted the first three subjects' data. **Figure 8a** shows

the correlation-based RDMs of three different time-points using NeuroRA (SVM-based RDMs in Cichy et al., 2014 for the first three subjects can be seen in **Figure S1a**) and the temporal similarity results by comparing with the neural representations of 200ms and 800ms. There were more similar neural patterns when participants were viewing human faces (the small blue squares in RDMs), and representations of nearby times were more similar. The second demo is based on the subject2's data in Haxby fMRI datasets (Haxby, 2001). **Figure 8b** shows the searchlight-based RSA results between an "animate-inanimate" coding model RDM and searchlight RDMs from fMRI data. The results indicated that the temporal cortex was primarily involved in coding information of animate or inanimate. The third demo is based on EEG datasets from a working memory task using NeuroRA (Bae and Luck, 2018). We extracted the first five subjects' event-related potentials (ERP) data. **Figure 8c** shows the RSA-based decoding results by comparing a coding model RDM and temporal RDMs from EEG data (The temporal SVM-based decoding results of these five subjects can be seen in **Figure S1b**). Both orientation and position could be successfully decoding based on ERP data in a visual working memory task. In these demos, user can learn how to use NeuroRA to perform representational analysis and plot the main results, including calculating RDMs from different time points (**Figure 8a**), correlations over the time series (**Figure 8a**), searchlight calculation between the brain activities and an "animate-inanimate" coding model (**Figure 8b**), using the hypothesis-based RDM to fit RDMs based on neural activities by time sequence (**Figure 8c**) and so on (see more: https://github.com/neurora/NeuroRA/tree/master/demo). These demos contain several critical sections: loading data & preprocessing, calculating RDMs, calculating the neural similarities or similarity matrix, and plotting. Users can download the tutorial on NeuroRA website and find further details.

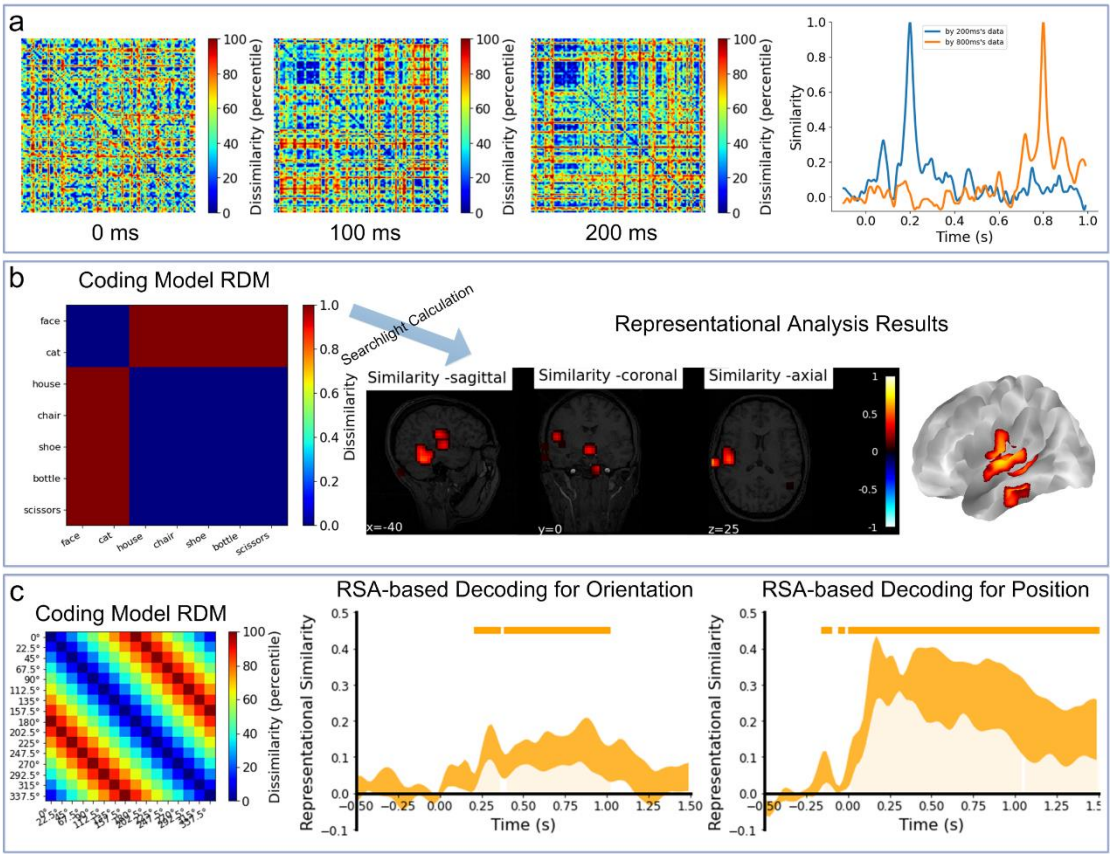

Figure 8 Demo results. (a) Left: The RDMs of 0ms, 100ms, 200ms, 300ms based on all 302 channels' MEG data for the first three subjects (data from Cichy et al., 2014). Right: Use the neural representations of 200ms and 800ms to calculate the similarities with all time-points' neural representations. (b) The searchlight results between an 'animate-inanimate' coding model RDM and searchlight RDMs based on subject2's data (data from Haxby, 2001). In this coding model RDM, we assume that there are consistent representations among animate objects and inanimate objects. (c) The RSA-based decoding results for orientation and position by calculating the correlation coefficients between a coding model RDM and EEG RDMs by time sequence based on the first five subjects' data in experiment 2 (data from Bae and Luck, 2018). In this coding model RDM, we assume that a large difference between the corresponding two angles corresponds to high dissimilarity, and vice versa. In the two rightmost plots, the small orange rectangles inside the plotting area and orange shadow indicate $p < 0.05$; line width reflects ± SEM.

User Support

To report any bugs in the code or submit any queries or suggestions about our toolbox, users can use the issue tracker on GitHub: https://github.com/neurora/NeuroRA/issues. We will reply and act accordingly as soon as possible.

Discussion

RSA provides a novel way of observing big data, which is powerful in the field of cognitive neuroscience. An increasing number of studies have used such multivariate analysis to obtain novel information that could not be observed through univariate analysis (Mahmoudi et al., 2012; Sui et al., 2012; Haxby et al., 2014). More importantly, experimental data obtained from different modalities must be assessed simultaneously, and RSA is a suitable method way to quantitatively compare results from different modalities with distinctive dimensions, resolutions and even obtained from different species (Cichy and Pantazis, 2017; Salmela et al., 2016).

In the present study, we have developed a Python-based toolbox that can perform representation analysis for neural data from many different modalities. Compared with other toolkits or modules that can also implement RSA, our toolbox has a much wider application and more convenient and rich functionalities that users can use tiny codes to conduct not only RSA but also NPS, STPS, ISC, statistical analysis, and visualization, especially for the analysis of multi-modal data and cross-modal comparisons. Moreover, it is open-source, free to use, and cross-platform.

For detailed information on each module and function in our toolbox, including the format of input data, the choice of parameters, and the format of output data, users can refer to our toolbox tutorial. To further understand the specific implementation of each function in the toolbox, people can read the source code directly. If users encounter any problems or difficulties during use, they can consult NeuroRA's tutorials and email our developers.

Although we already implemented the essential functions for representational analysis, there are still several limitations to be addressed in the future. First, NeuroRA is not yet designed to process the raw data. However, users can utilize other toolboxes such as EEGLAB (Delorme and Makeig, 2004), MNE (Gramfort et al., 2013), and Nibabel (Brett et al., 2016) to import data and transfer them into a format fit for NeuroRA. We plan to develop an integrated format conversion function in the next version. Second, there is still significant room for improving the computational performance of NeuroRA, especially in the iterative calculation of fMRI data, which is often slow. This is due to nested loops in the code structure when we need to traverse the data from the entire brain and iterate the random shuffle many times. In the future, we will reduce the time by optimizing functions with GPUs and using some multithreaded methods to accelerate some computing processes. Third, there is no graphical user interface (GUI) right now, which we plan to design and implement based on PyQt in a future version. Users could then obtain the final results by dragging

the data file to a specific location in the GUI with the mouse and filling in the relevant parameters. Fourth, object-oriented programming methods can also be applied to our toolkit development. We can build some classes with some public methods requiring the visualization or statistical analysis parameters and some private methods for data management of the multidimensional matrices hidden from the user. Fifth, we need to add some features for the plotting part, such as returning the matplotlib object, assembling subplots and saving them instead of displaying plots on screen only. Sixth, we hope to provide a more straightforward version by streamlining the full analysis workflow building on existing functions. After simplifying the intermediate process, users don't need to call other functions to do extra operations for data transformation. Finally, although we added unit tests in our toolbox, the tests available assess only the shapes of the output corresponding to different inputs rather than check the correctness of the computations. The work to add them will be an important part of NeuroRA's future development.

Through NeuroRA, for the first time, we provide a method for researchers to perform representation analysis with neural data from many different modalities. However, this is only a starting point. With the development of the algorithm and applications for representational analysis, we will include new functionalities, such as using the classification-based decoding accuracy between neural activities under two conditions as the value in an RDM (Cichy et al., 2014; Cichy and Pantazis, 2017; Xie et al., 2020) and automatically generating RDMs for each layer in a deep convolutional neural network, as well as new visualization tools which can plot the space representation of neural activities with t-SNE and show the dynamic representational analysis results. We hope that many exciting findings can be observed through our toolbox, and we would like to collaborate with researchers interested in this tool to improve the toolbox further.

Information Sharing Statement

NeuroRA is available at https://pypi.org/project/neurora/. The website for NeuroRA is https://neurora.github.io/NeuroRA/, and the tutorial of the toolbox can be download here. Also, users can read online API documentation on https://neurora.github.io/documentation/. The code for our toolbox NeuroRA can be accessed on GitHub: https://github.com/neurora/NeuroRA.

References

Alfred, K.L., Connolly, A.C., & Kraemer, D.J.M. (2018). Putting the pieces together: Generating a novel representational space through deductive

reasoning. *NeuroImage*, 183, 99–111. https://doi.org/10.1016/j.neuroimage.2018.07.062

Bae, G.Y., & Luck, S.J. (2019). Dissociable decoding of spatial attention and working memory from EEG oscillations and sustained potentials. *The Journal of Neuroscience*, 38(2), 409-422. https://doi.org/10.1523/JNEUROSCI.2860-17.2017

Brett, M., Hanke, M., Cipollini, B., Côté, M.-A., Markiewicz, C., Gerhard, S., et al. (2016). Nibabel: Access a cacophony of neuro-imaging file formats, version 2.1. 0. *Zenodo*. https://doi.org/10.5281/zenodo.591597

Bonner, M.F., & Epstein, R.A. (2018). Computational mechanisms underlying cortical responses to the affordance properties of visual scenes. PLOS *Computational Biology*, 14(4), e1006111. https://doi.org/10.1371/journal.pcbi.1006111

Cavanagh, S.E., Towers, J.P., Wallis, J.D., Hunt, L.T., & Kennerley, S.W. (2018). Reconciling persistent and dynamic hypotheses of working memory coding in prefrontal cortex. *Nature Communications*, 9, 3498. https://doi.org/10.1038/s41467-018-05873-3

Chen, Y., Shimotake, A., Matsumoto, R., Kunieda, T., Kikuchi, T., Miyamoto, S., et al. (2016). The 'when' and 'where' of semantic coding in the anterior temporal lobe: Temporal representational similarity analysis of electrocorticogram data. *Cortex*, 79, 1–13. https://doi.org/10.1016/j.cortex.2016.02.015

Cichy, R.M., & Pantazis, D. (2017). Multivariate pattern analysis of MEG and EEG: A comparison of representational structure in time and space. *NeuroImage*, 158, 441–454. https://doi.org/10.1016/j.neuroimage.2017.07.023

Cichy, R.M., Pantazis, D., & Oliva, A. (2014). Resolving human object recognition in space and time. *Nature Neuroscience*, 17(3), 455–462. https://doi.org/10.1038/nn.3635

Delorme, A., & Makeig, S. (2004). EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134(1), 9–21. https://doi.org/10.1016/j.jneumeth.2003.10.009

Eickenberg, M., Gramfort, A., Varoquaux, G., & Thirion, B. (2017). Seeing it all: Convolutional network layers map the function of the human visual system.

NeuroImage, 152, 184–194. https://doi.org/10.1016/j.neuroimage.2016.10.001

Efron, B., & Tibshirani, R.J. (1994). *An introduction to the bootstrap*. Boca Raton: CRC press.

Evans, S., & Davis, M.H. (2015). Hierarchical Organization of Auditory and Motor Representations in Speech Perception: Evidence from Searchlight Similarity Analysis. *Cerebral Cortex*, 25(12), 4772–4788. https://doi.org/10.1093/cercor/bhv136

Etzel, J.A., Courtney, Y., Carey, C.E., Gehred, M.Z., Agrawal, A., & Braver, T.S. (2020). Pattern Similarity Analyses of FrontoParietal Task Coding: Individual Variation and Genetic Influences. *Cerebral Cortex*, 30(5), 3167-3183. https://doi.org/10.1093/cercor/bhz301

Feng, C., Yan, X., Huang, W., Han, S., & Ma, Y. (2018). Neural representations of the multidimensional self in the cortical midline structures. *NeuroImage*, 183, 291-299. https://doi.org/10.1016/j.neuroimage.2018.08.018

Garcia, S., Guarino, D., Jaillet, F., et al. (2014). Neo: an object model for handling electrophysiology data in multiple formats. *Frontiers in Neuroinformatics*, 8. https://doi.org/10.3389/fninf.2014.00010.

Gramfort, A., Luessi, M., Larson, E., Engemann, D.A., Strohmeier, D., Brodbeck, C., et al. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7. https://doi.org/10.3389/fnins.2013.00267

Greene, M.R., & Hansen, B.C. (2018). Shared spatiotemporal category representations in biological and artificial deep neural networks. *PLoS Computational Biology*, 14(7), e1006327. https://doi.org/10.1371/journal.pcbi.1006327

Güçlü, U., & van Gerven, M.A. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27), 10005–10014. https://doi.org/10.1523/JNEUROSCI.5023-14.2015

Hall-McMaster, S., Muhle-Karbe, P.S., Myers, N.E., Stokes, M.G. (2019) Reward Boosts Neural Coding of Task Rules to Optimize Cognitive Flexibility. *The Journal of Neuroscience*, 39(43), 8549-8561. https://doi.org/10.1523/JNEUROSCI.0631-19.2019

Hanke, M., Halchenko, Y.O., Sederberg, P.B., Hanson, S.J., Haxby, J.V., &

Pollmann, S. (2009). PyMVPA: A Python Toolbox for Multivariate Pattern Analysis of fMRI Data. *Neuroinformatics*, 7(1), 37–53. https://doi.org/10.1007/s12021-008-9041-y

Hasson, U., Nir, Y., Levy, I., Fuhrmann, G., Malach, R. (2004) Intersubject synchronization of cortical activity during natural vision. Science, 303, 1634-1640. https://doi.org/10.1126/sciencce.1089506

Haxby, J.V. (2001). Distributed and Overlapping Representations of Faces and Objects in Ventral Temporal Cortex. *Science*, 293(5539), 2425–2430. https://doi.org/10.1126/science.1063736

Haxby, J.V., Connolly, A.C., & Guntupalli, J.S. (2014). Decoding neural representational spaces using multivariate pattern analysis. *Annual review of neuroscience*, 37, 435-456. https://doi.org/10.1146/annurev-neuro-062012-170325

Henriksson, L., Mur, M., & Kriegeskorte, N. (2019). Rapid Invariant Encoding of Scene Layout in Human OPA. *Neuron*, 103(1), 161-171.e3. https://doi.org/10.1016/j.neuron.2019.04.014

Hunter, J.D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3), 90–95. https://doi.org/10.1109/mcse.2007.55

Johnson, M. K., Raye, C. L., Mitchell, K. J., Greene, E. J., Cunningham, W. A., & Sanislow, C. A. (2005). Using fMRI to investigate. *Cognitive, Affective, & Behavioral Neuroscience*, 5(3), 339–361. http://doi.org/10.3758/cabn.5.3.339

Jordan, M.I., & Mitchell, T.M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260. https://doi.org/10.1126/science.aaa8415

Khaligh-Razavi, S.-M., & Kriegeskorte, N. (2014). Deep Supervised, but Not Unsupervised, Models May Explain IT Cortical Representation. *PLoS Computational Biology*, 10(11), e1003915. https://doi.org/10.1371/journal.pcbi.1003915

Koepke, H. (2011). Why Python rocks for research. *Hacker Monthly*, 8.

Kriegeskorte, N. (2008). Representational similarity analysis – connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*. https://doi.org/10.3389/neuro.06.004.2008

Kriegeskorte, N., & Golan, T. (2019). Neural network models and deep learning. *Current Biology*, 29(7), R231–R236. https://doi.org/10.1016/j.cub.2019.02.034

Kriegeskorte, N., Mur, M., Ruff, D. A., Kiani, R., Bodurka, J., Esteky, H., et al. (2008). Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey. *Neuron*, 60(6), 1126–1141. https://doi.org/10.1016/j.neuron.2008.10.043

Kuzovkin, I., Vicente, R., Petton, M., Lachaux, J.-P., Baciu, M., Kahane, P., et al. (2018). Activations of deep convolutional neural networks are aligned with gamma band activity of human visual cortex. *Communications Biology*, 1(1). https://doi.org/10.1038/s42003-018-0110-y

Lawrence, S. J. D., Formisano, E., Muckli, L., & de Lange, F. P. (2019). Laminar fMRI: Applications for cognitive neuroscience. *NeuroImage*, 197, 785-791. doi:10.1016/j.neuroimage.2017.07.004

Lu, Y., Wang, C., Chen, C., & Xue, G. (2015). Spatiotemporal Neural Pattern Similarity Supports Episodic Memory. Current Biology, 25(6), 780–785. https://doi.org/10.1016/j.cub.2015.01.055

Lu, Z., & Ku, Y. (2020). NeuroRA: A Python Toolbox of Representational Analysis from Multi-modal Neural Data. bioRxiv. https://doi.org/10.1101/2020.03.25.008086

Mahmoudi, A., Takerkart, S., Regragui, F., Boussaoud, D., & Brovelli, A. (2012). Multivoxel Pattern Analysis for fMRI Data: A Review. *Computational and Mathematical Methods in Medicine*, 2012, 1–14. https://doi.org/10.1155/2012/961257

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco: W. H. Freeman.

Muukkonen, I., Ölander, K., Numminen, J., & Salmela, V.R. (2020). Spatio-temporal dynamics of face perception. *NeuroImage*, 209, 116531. https://doi.org/10.1016/j.neuroimage.2020.116531

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., & Kriegeskorte, N. (2014). A Toolbox for Representational Similarity Analysis. *PLOS Computational Biology*, 10(4), e1003553. https://doi.org/10.1371/journal.pcbi.1003553

Norman, K. A., Polyn, S. M., Detre, G. J., & Haxby, J. V. (2006). Beyond mind-reading: Multi-voxel pattern analysis of fMRI data. *Trends in Cognitive Sciences*, 10(9), 424–430. https://doi.org/10.1016/j.tics.2006.07.005

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.

Peirce, J. W. (2007). PsychoPy—Psychophysics software in Python. *Journal of Neuroscience Methods*, 162(1–2), 8–13. https://doi.org/10.1016/j.jneumeth.2006.11.017

Poldrack, R. A. (2012). The future of fMRI in cognitive neuroscience. *NeuroImage*, 62(2), 1216–1220. https://doi.org/10.1016/j.neuroimage.2011.08.007

Ritchie, J. B., Bracci, S., & Op de Beeck, H. (2017). Avoiding illusory effects in representational similarity analysis: What (not) to do with the diagonal. *NeuroImage*, 148, 197–200. https://doi.org/10.1016/j.neuroimage.2016.12.079

Rosen, B.R., & Savoy, R.L. (2012). fMRI at 20: Has it changed the world? *NeuroImage*, 62(2), 1316–1324. https://doi.org/10.1016/j.neuroimage.2012.03.004

Salmela, V., Salo, E., Salmi, J., & Alho, K. (2016). Spatiotemporal Dynamics of Attention Networks Revealed by Representational Similarity Analysis of EEG and fMRI. *Cerebral Cortex*. https://doi.org/10.1093/cercor/bhw389

Sanner, M. F. (1999). Python: a programming language for software integration and development. *Journal of Molecular Graphics and Modelling*, 17(1), 57–61.

Stolier, R.M., & Freeman, J.B. (2016). Neural pattern similarity reveals the inherent intersection of social categories. *Nature Neuroscience*, 19(6), 795–797. https://doi.org/10.1038/nn.4296

Sui, J., Adali, T., Yu, Q., Chen, J., & Calhoun, V. D. (2012). A review of multivariate methods for multi-modal fusion of brain imaging data. *Journal of Neuroscience Methods*, 204(1), 68–81. https://doi.org/10.1016/j.jneumeth.2011.10.031

Urgen, B.A., Pehlivan, S., & Saygin, A.P. (2019). Distinct representations in occipito-temporal, parietal, and premotor cortex during action perception

revealed by fMRI and computational modeling. *Neuropsychologia*, 127, 35-47. https://doi.org/10.1016/j.neuropsychologia.2019.02.006

van der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science & Engineering*, 13(2), 22–30. https://doi.org/10.1109/MCSE.2011.37

Virtanen, P., Gommers, R., Oliphant, T.E., et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17, 261-272. https://doi.org/10.1038/s41592-019-0686-2

Wang, X., Xu, Y., Wang, Y., Zeng, Y., Zhang, J., Ling, Z., & Bi, Y. (2018). Representational similarity analysis reveals task-dependent semantic influence of the visual word form area. *Scientific Reports*, 8(1), 3047. https://doi.org/10.1038/s41598-018-21062-0

Xie, S., Kaiser, D., Cichy, R.M. (2020). Visual imagery and perception share representations in the alpha frequency band. *Current Biology*, 30(15), 2621-2627.e5. https://doi.org/10.1016/j.cub.2020.04.074

Xue, G., Dong, Q., Chen, C., Lu, Z., Mumford, J. A., & Poldrack, R. A. (2010). Greater Neural Pattern Similarity Across Repetitions Is Associated with Better Memory. *Science*, 330(6000), 97–101. https://doi.org/10.1126/science.1193125

Yamins, D.L.K., Hong, H., Cadieu, C.F., Solomon, E.A., Seibert, D., & DiCarlo, J.J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23), 8619–8624. https://doi.org/10.1073/pnas.1403112111

Yan, C., Su, L., Wang, Y., Xu, T., Yin, D., Fan, M., et al. (2016). Multivariate Neural Representations of Value during Reward Anticipation and Consummation in the Human Orbitofrontal Cortex. *Scientific Reports*, 6(1). https://doi.org/10.1038/srep29079

Yokoi, A., & Diedrichsen, J. (2019). Neural Organization of Hierarchical Motor Sequence Representations in the Human Neocortex. *Neuron*, 103(6), 1178-1190. https://doi.org/10.1016/j.neuron.2019.06.017

Acknowledgements

We gratefully acknowledge the support of the National Social Science Foundation of China (17ZDA323), the Shanghai Committee of Science and Technology (19ZR1416700) and the Hundred Top Talents Program from Sun Yat-sen University to YK. This manuscript has been released as a pre-print at *bioRxiv*, (Lu Z & Ku Y)

Supplementary Materials

Supplementary instructions for data conversion

For EEG/MEG data

Users can use MATLAB toolbox such as EEGLab (http://sccn.ucsd.edu/eeglab/) to do preprocessing and obtain *.mat* files, then use SciPy (https://www.scipy.org) to load EEG data (*.mat*) as ndarray-type data. Sample codes:

```
>>> import scipy.io as sio
>>> data = sio.loadmat(filename)[“data”])
```

Or users can use MNE (https://mne-tools.github.io) to do preprocessing and return ndarray-type data. Sample codes:

```
>>> # here epoch should be an Epoch object in MNE-Python
>>> data = epoch.get_data()
```

Also, for EEG data, users can use Neo (Garcia et al., 2014) (https://neuralensemble.org/neo/) to preprocess and return ndarray-type data. See more detail in Neo io module, and it provides many methods for reading different formats from different EEG acquisition systems.

For fMRI data

We strongly recommend users to use Nibabel (https://nipy.org/nibabel/) to load fMRI data as ndarray-type data. Sample codes:

```
>>> import nibabel as nib
>>> data = nib.load(fmrifilename).get_fdata()
```

For fNIRS data

For raw data from device, users can use Numpy (http://www.numpy.org) to load fNIRS data (*.txt* or *.csv*) as ndarray-type data. Sample codes:

```
>>> import numpy as np
>>> # load fNIRS data of .txt file as ndarray
>>> data = np.loadtxt(txtfilename)
>>> # load fNIRS data of .csv file as ndarray
>>> data = np.loadtxt(csvfilename, delimiter, usecols, unpack)
```

For other sources of neuroelectrophysiological data

Users can use Brainstorm (https://neuroimage.usc.edu/brainstorm/) to preprocess and obtain .mat files and then use SciPy to load ECoG data (*.mat*) as ndarray-type data.

Or users can use Neo (https://neuralensemble.org/neo/) to do preprocessing and return ndarray-type data. See more detail in Neo io module, and it provides many methods for reading different formats from different neuroelectrophysiology acquisition systems.

Also, users can use pyABF (https://github.com/swharden/pyABF) for Axon system, to load electrophysiology data (*.abf*) as ndarray-type data. Sample codes:

```
>>> import pyabf
>>> # the electrophysiology data file name with full address
>>> abf = pyabf.ABF("demo.abf")
>>> # access sweep data
>>> abf.setSweep(sweepNumber, channel)
>>> # get sweep data with sweepY
>>> data = abf.sweepY
```

Notes

Two functions, *NumPy.reshape*() & *NumPy.transpose*(), are recommended for further data transformation.

Table S1 Basic structure of inputs and outputs of functions in some key calculation modules in NeuroRA. This table shows the shape of input data, the shape of output data corresponding to different parameter settings and recommended next steps for some key calculation modules, includes *neurora.nps_cal* module, *neurora.isc_cal* module, *neurora.stps_cal* module, *neurora.rdm_cal* module, *neurora.corr_cal* module and *neurora.corr_cal_by_rdm* module. The variable definitions are shown in Table S3.

neurora.nps_cal **module**

a module for calculating the neural pattern similarity based on neural data

neurora.nps_cal,nps**()** – to calculate the neural pattern similarity (NPS) for EEG-like data			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[2^a, *n_subs*, *n_trials*, *n_chls*, *n_ts*]	*sub_opt*=0 -return the average results for all subjects	[*n_chls*, *int*((*n_ts*-*time_win*)/*time_step*)+*1*, 2^b]	——
	sub_opt=1 -return the results for all subject	[*n_subs*, *n_chls*, *int*((*n_ts*-*time_win*)/*time_step*)+*1*, 2^b]	*neurora.stats_cal.stats*() -to the conduct statistical analysis *neurora.rsa_plot.plot_corrs_hotmap*() -to plot (average the subjects first)
neurora.nps_cal,nps_fmri**()** – to calculate the neural pattern similarity (NPS) for fMRI data (searchlight)			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[2^a, *n_subs*, *nx*, *ny*, *nz*]	——	[*n_subs*, *n_x*, *n_y*, *n_z*, 2^b]	*neurora.stats_cal.stats_fmri*() -to the conduct statistical analysis *neurora.nii_save.corr_save_nii*() -to save one subject's results as a .nii file *neurora.nii_save.stats_save_nii*() -to save the statistical results as a .nii file

			(after statistical analysis)
***neurora.nps_cal,nps_fmri_roi*()** – to calculate the neural pattern similarity (NPS) for fMRI data for ROI			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
fmri_data: [2^a, *n_subs*, *nx*, *ny*, *nz*] mask_data: [*nx*, *ny*, *nz*]	——	[*n_subs*, 2^b]	——

neurora.isc_cal module

a module for calculating the inter-subject correlation based on neural data

***neurora.isc_cal,isc*()** – to calculate the inter subject correlation (ISC) for EEG-like data			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_subs*, *n_chls*, *n_ts*]	——	[*n_subs!*/(*2!*×(*n_subs*-*2*)*!*), *n_chls*, i*nt*((*n_ts*-*time_win*)/*time_step*)+*1*, 2^b]	*neurora.stats_cal.stats*() -to the conduct statistical analysis *neurora.rsa_plot.plot_corrs_hotmap*() -to plot (average the subjects first)
***neurora.isc_cal,isc_fmri*()** – to calculate the inter subject correlation (ISC) for fMRI data (searchlight)			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_ts*, *n_subs*, *nx*, *ny*, *nz*]	——	[*n_ts*, *n_subs!*/(*2!*×(*n_subs*-*2*)*!*), *n_x*, *n_y*, *n_z*, 2^b]	*neurora.stats_cal.stats_iscfmri*() -to the conduct statistical analysis *neurora.nii_save.stats_save_nii*() -to save the statistical results as a .nii file (after statistical analysis)

***neurora.isc_cal,nps_fmri_roi*()** – to calculate the inter subject correlation (ISC) for fMRI data for ROI			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
fmri_data: [*n_ts*, *n_subs*, *nx*, *ny*, *nz*] mask_data: [*nx*, *ny*, *nz*]	——	[*n_ts*, *n_subs!*/(*2!*×(*n_subs*-*2*)*!*), 2^b]	——

neurora.stps_cal module

a module for calculating the spatiotemporal pattern similarity based on neural data

***neurora.stps_cal,stps*()** – to calculate the spatiotemporal pattern similarity (STPS) for EEG-like data			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_subs*, *n_trials*, *n_chls*, *n_ts*]	——	[*n_subs*, *8**, *n_chls*, *int*((*n_ts*-*time_win*)/*time_step*) *+1*]	*neurora.stats_cal.stats_stps*() -to the conduct statistical analysis *neurora.rsa_plot.plot_corrs_hotmap*() -to plot (average the subjects and eight conditions first)
***neurora.stps_cal,stps_fmri*()** – to calculate the spatiotemporal pattern similarity (STPS) for fMRI data (searchlight)			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_subs*, *n_trials*, *n_x*, *n_y*, *n_z*]	——	[*n_subs*, *8**, *n_x*, *n_y*, *n_z*]	*neurora.stats_cal.stats_stpsfmri*() -to the conduct statistical analysis *neurora.nii_save.stats_save_nii*() -to save the statistical results as a .nii file (after statistical analysis)

***neurora.stps_cal,stps_fmri_roi*()** – to calculate the spatiotemporal pattern similarity (STPS) for fMRI data for ROI			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
fmri_data: [*n_subs*, *n_trials*, *nx*, *ny*, *nz*] mask_data: [*nx*, *ny*, *nz*]	——	[*n_subs*, *8**]	——

neurora.rdm_cal module

a module for calculating the RDM based on multimode neural data

***neurora.rdm_cal,bhvRDM*()** – to calculate the RDM(s) for behavioral data			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_cons*, *n_subs*, *n_trials*]	*sub_opt*=0 -return the average RDM for all subjects	[*n_cons*, *n_cons*]	fuctions in *neurora.rdm_corr* module -to calculate the similarity between two RDMs functions in *neurora.corr_cal_by_rdm* module -to calculate the similarities between other RDMs and the behavioral RDM *neurora.rsa_plot.plot_rdm*() -to plot one RDM
	sub_opt=1 -return the RDMs for all subject	[*n_subs*, *n_cons*, *n_cons*]	
***neurora.rdm_cal,eegRDM*()** – to calculate the RDM(s) for EEG-like data			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_cons*, *n_subs*, *n_trials*, *n_chls*,	*sub_opt*=0 & *chl_opt*=0 & *time_opt*=0 -average the subjects and trials, and	[*n_cons*, *n_cons*]	fuctions in *neurora.rdm_corr* module -to calculate the similarity between two

n_ts]	return only one RDM		RDMs *neurora.corr_cal_by_rdm.rdms_corr*() -to calculate the similarities between RDMs of EEG-like data and a demo RDM *neurora.rsa_plot.plot_rdm*() -to plot one RDM
	sub_opt=0 & *chl_opt*=0 & *time_opt*=1 -average the subjects and trials, calculate for each time-window and return RDMs for each time-window	[*int*((*n_ts-time_win*)/*time_step*)+*1*, *n_cons*, *n_cons*]	
	sub_opt=0 & *chl_opt*=1 & *time_opt*=0 -average the subjects and trials, calculate for each channel and return RDMs for each channel	[*n_chls*, *n_cons*, *n_cons*]	
	sub_opt=0 & *chl_opt*=1 & *time_opt*=1 -average the subjects and trials, calculate for each channel and each time-window, and return RDMs for each channel and each time-window	[*n_chls*, *int*((*n_ts-time_win*)/*time_step*)+*1*, *n_cons*, *n_cons*]	
	sub_opt=1 & *chl_opt*=0 & *time_opt*=0 -average the trials, calculate for each subject, return RDMs for each subject	[*n_subs*, *n_cons*, *n_cons*]	
	sub_opt=1 & *chl_opt*=0 & *time_opt*=1 -average the trials, calculate for each subject and each time-window, and return RDMs for each subject and each time-window	[*n_subs*, *int*((*n_ts-time_win*)/*time_step*)+*1*, *n_cons*, *n_cons*]	
	sub_opt=1 & *chl_opt*=1 & *time_opt*=0 -average the trials, calculate for each subject and each channel and return RDMs for each subject and each channel	[*n_subs*, *n_chls*, *n_cons*, *n_cons*]	
	sub_opt=1 & *chl_opt*=1 & *time_opt*=1	[*n_subs*, *n_chls*, *int*((*n_ts-*	

	-average the trials, calculate for each subject, each channel and each time-window, and return RDMs for each subject, each channel and each time-window	*time_win*)/*time_step*)+*1*, *n_cons*, *n_cons*]	

***neurora.rdm_cal,fmriRDM*()** – to calculate the RDM(s) for fMRI data (searchlight)

shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
[*n_cons*, *n_subs*, *nx*, *ny*, *nz*]	*sub_opt*=0 -calculate for each subject, and return the average RDMs for all subjects	[*n_x*, *n_y*, *n_z*, *n_cons*, *n_cons*]	fuctions in *neurora.rdm_corr* module -to calculate the similarity between two RDMs *neurora.corr_cal_by_rdm.fmrirdms_corr*() -to calculate the similarities between RDMs of fMRI data and a demo RDM *neurora.rsa_plot.plot_rdm*() -to plot one RDM
	sub_opt=1 -calculate for each subject, and return the RDMs for all subject	[*n_subs*, *n_x*, *n_y*, *n_z*, *n_cons*, *n_cons*]	

***neurora.rdm_cal,fmriRDM_roi*()** – to calculate the RDM(s) for fMRI data (for ROI)

shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
fmri_data: [*n_subs*, *n_trials*, *nx*, *ny*, *nz*] mask_data: [*nx*, *ny*, *nz*]	*sub_opt*=0 -calculate for each subject, and return the average RDM for all subjects	[*n_cons*, *n_cons*]	———
	sub_opt=1 -calculate for each subject, and return the RDMs for all subject	[*n_subs*, *n_cons*, *n_cons*]	

***neurora.corr_cal* module**

a module for calculating the similarity between two different modes' data

***neurora.corr_cal,bhvANDeeg_corr*()** – to calculate the similarity between behavioral data and EEG-like data

shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
bhv_data: [*n_cons*, *n_subs*, *n_trials*] eeg_data: [*n_cons*, *n_subs*, *n_trials*, *n_chls*, *n_ts*]	*sub_opt*=0 & *chl_opt*=0 & *time_opt*=0 -calculate one RDM for behavioral data and one RDM for EEG-like data, return the similarity between these two RDMs	[2^b]	When *sub_opt*=1: *neurora.stats_cal.stats*() -to the conduct statistical analysis
	sub_opt=0 & *chl_opt*=0 & *time_opt*=1 -calculate one RDM for behavioral data and multiple RDMs for each time-window for EEG-like data, and return the similarity between behavioral RDM and EEG-like data's RDMs	[*int*((*n_ts*-*time_win*)/*time_step*)+*1*, 2^b]	
	sub_opt=0 & *chl_opt*=1 & *time_opt*=0 -calculate one RDM for behavioral data and multiple RDMs for each channel for EEG-like data, and return the similarity between behavioral RDM and EEG-like data's RDMs	[*n_chls*, 2^b]	
	sub_opt=0 & *chl_opt*=1 & *time_opt*=1 -calculate one RDM for behavioral data and multiple RDMs for each channel and each time-window for EEG-like data, and return the similarity between behavioral RDM and EEG-like data's RDMs	[*n_chls*, *int*((*n_ts*-*time_win*)/*time_step*)+*1*, 2^b]	

sub_opt=1 & *chl_opt*=0 & *time_opt*=0 -calculate multiple RDMs for each subject for behavioral data and multiple RDMs for each subject for EEG-like data, and return the similarity between behavioral RDMs and EEG-like data's RDMs	[*n_subs*, 2^b]
sub_opt=1 & *chl_opt*=0 & *time_opt*=1 -calculate multiple RDMs for each subject for behavioral data and multiple RDMs for each subject and each time-window for EEG-like data, and return the similarity between behavioral RDMs and EEG-like data's RDMs	[*n_subs*, *int*((*n_ts-time_win*)/*time_step*)+*1*, 2^b]
sub_opt=1 & *chl_opt*=1 & *time_opt*=0 -calculate multiple RDMs for each subject for behavioral data and multiple RDMs for each subject and each channel for EEG-like data, and return the similarity between behavioral RDMs and EEG-like data's RDMs	[*n_subs*, *n_chls*, 2^b]
sub_opt=1 & *chl_opt*=1 & *time_opt*=1 -calculate multiple RDMs for each subject for behavioral data and multiple RDMs for each subject, each channel and each time-window for EEG-like data, and return the similarity between behavioral RDMs and EEG-like data's RDMs	[*n_subs*, *n_chls*, *int*((*n_ts-time_win*)/*time_step*)+*1*, 2^b]

***neurora.corr_cal,bhvANDfmri_corr*()** – to calculate the similarity between behavioral data and fMRI data (searchlight)			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
bhv_data: [*n_cons*, *n_subs*, *n_trials*] fmri_data: [*n_cons*, *n_subs*, *nx*, *ny*, *nz*]	*sub_result*=0 -calculate for each subject, and return the average similarities for all subjects	[*n_x*, *n_y*, *n_z*, 2^b]	When *sub_result*=1: *neurora.stats_cal.stats_fmri*() -to the conduct statistical analysis
	sub_result=1 -calculate for each subject, and return the similarities for all subject	[*n_subs*, *n_x*, *n_y*, *n_z*, 2^b]	
***neurora.rdm_cal,eegANDfmri_corr*()** – to calculate the similarity between EEG-like data for fMRI data (for ROI)			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
eeg_data: [*n_cons*, *n_subs*, *n_trials*, *n_chls*, *n_ts*] fmri_data: [*n_cons*, *n_subs*, *nx*, *ny*, *nz*]	*chl_opt*=1 & *sub_result*=1 -calculate multiple RDMs for EEG-like data for each channel and each subject and RDMs for fMRI data for each subject, return the similarities between EEG-like RDMs and fMRI RDMs for each subject	[*n_subs*, *n_chls*, *n_x*, *n_y*, *n_z*, 2^b]	When *sub_result*=1: *neurora.stats_cal.stats*() -to the conduct statistical analysis
	chl_opt=1 & *sub_result*=0 -calculate multiple RDMs for EEG-like data for each channel and each subject and RDMs for fMRI data for each subject, return the averaged similarities between EEG-like RDMs and fMRI RDMs for all subjects	[*n_chls*, *n_x*, *n_y*, *n_z*, 2^b]	
	chl_opt=0 & *sub_result*=1 -calculate multiple RDMs for EEG-like	[*n_subs*, *n_x*, *n_y*, *n_z*, 2^b]	

	data for each subject and RDMs for fMRI data for each subject, return the averaged similarities between EEG-like RDMs and fMRI RDMs for all subjects		
	chl_opt=0 & *sub_result*=0 -calculate one RDM for EEG-like data and multiple RDMs after averaging subjects for EEG-like data, and return the similarities between EEG-like RDM and fMRI data's RDMs	[*n_x*, *n_y*, *n_z*, 2^b]	

neurora.corr_cal_by_rdm **module**

a module for calculating the similarity between two different modes' data

neurora.corr_cal_by_rdm,rdms_corr**()** – to calculate the similarity between RDMs of EEG-like data and a demo RDM			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**
demo_rdm: [*n_cons*, *n_cons*] eeg_rdms: [*n_cons*, *n_cons*] or [*n1*, *n_cons*, *n_cons*] or [*n1*, *n2*, *n_cons*, *n_cons*] or [*n1*, *n2*, *n3*, *n_cons*, *n_cons*]	____	[2^b] or [*n1*, 2^b] or [*n1*, *n2*, 2^b] or [*n1*, *n2*, *n3*, 2^b]	When *sub_opt*=1: *neurora.stats_cal.stats*() -to the conduct statistical analysis
neurora.corr_cal_by_rdm,fmrirdms_corr**()** – to calculate the similarity between fMRI searchlight RDMs and a demo RDM			
shape of input data	**parameter settings**	**corresponding shape of output data**	**recommended next steps**

demo_rdm: [*n_cons*, *n_cons*] fmri_rdms: [*nx*, *ny*, *nz*, *n_cons*, *n_cons*]	——	[*nx*, *ny*, *nz, 2^b*]	When *sub_result*=1: *neurora.stats_cal.stats*() -to the conduct statistical analysis

Table S2 Basic structure of inputs and outputs of functions in *stats_cal* module in NeuroRA. The variable definitions are shown in Table S3.

stats() – to conduct the statistical analysis for results of EEG-like data	
shape of input data: [*n_subs*, *n_chls*, *n_ts*, 2^b]	shape of output data: [*n_cons*, *n_cons*]
stats_fmri() – to conduct the statistical analysis for results of fMRI data (searchlight)	
shape of input data: [*n_subs*, *n_chls*, *n_ts*, 2^b]	shape of output data: [*n_cons*, *n_cons*]
stats_iscfmri() – to conduct the statistical analysis for results of fMRI data (ISC searchlight)	
shape of input data: [*n_ts*, *n_subs!/(2!×(n_subs-2)!)*, *n_x*, *n_y*, *n_z*, 2^b]	shape of output data: [*n_ts*, *n_x*, *n_y*, *n_z*, 2^b]
stats_stps() – to conduct the statistical analysis for results of EEG-like data (for STPS)	
shape of input data: *corrs1*: [*n_subs*, *n_chls*, *n_ts*] *corrs2*: [*n_subs*, *n_chls*, *n_ts*]	shape of output data: [*n_chls*, *n_ts*, 2^b]
stats_stpsfmri() – to conduct the statistical analysis for results of fMRI data (searchlight)	
shape of input data: *corrs1*: [*n_subs*, *n_x*, *n_y*, *n_z*] *corrs2*: [*n_subs*, *n_x*, *n_y*, *n_z*]	shape of output data: [*n_x*, *n_y*, *n_z*, 2^b]

Table S3 Definitions of the variables in Table S1 and Table S2.

Variable	Definition
2[a]	2 conditions (to calculate the NPS)
n_subs	the number of subjects
n_trials	the number of trials
n_chls	the number of channels
n_ts	the number of time-points
time_window	the number of time-points in each time interval for the calculation
time_step	the number of time-points in each time step for calculation
2[b]	2 values, including an *r*-value and a *p*-value
nx, *ny*, *nz*	the size of fMRI-img
n_x, *n_y*, *n_z*	the number of calculation unit for searchlight along the x, y, z axis
*8**	8 conditions for comparing (after calculating the STPS)
n_cons	the number of conditions

Supplementary information for implementation of analysis methods

Neural Pattern Similarity (NPS)

NPS is used to compare the neural activities under two different conditions. Users can input the data under two different conditions to functions in *neurora.nps_cal* module to obtain the similarity result. Two conditions can be two different task conditions in an experiment, such as viewing two kinds of stimuli (e.g., response to faces vs. response to houses in Haxby et al., 2001). In NeuroRA, we convert multidimensional neural activities into vectors and then calculate the similarities (Pearson Correlation) between two vectors corresponding to two different conditions. To be more specific, we provide multiple functions to calculate NPS for EEG-like data and fMRI data. For EEG-like data (using *nps*() function), users can set the time-window (the parameter: *time_win*) and time-step (the parameter: *time_step*) for customized computing. We calculate NPS for each subject. If *sub_opt*=0, our functions will return the average RDM(s) for all subjects. If *sub_opt*=1, our functions will return RDM(s) for each subject. For instance, if user sets *time_opt*=1, *time_win*=10, and *time_step*=5, the function will calculate the NPS results based on each time-window consisting of ten time-points, and the time step is a time interval consisting of 5 time-points between two time-windows for calculation. For fMRI data, users can set the calculation unit for searchlight calculation (using *nps_fmri*() function) or calculate the result from an ROI by a given mask matrix (using *nps_fmri_roi*() function). NPS is calculated for each subject, and users can receive all subjects' NPS results.

Spatiotemporal Pattern Similarity (STPS)

STPS is an improvement in NPS for a memory and learning task, and it reflects the representational similarity across different space and time points. In the experiment that researchers further want to calculate STPS, participants are asked to view stimuli under different conditions, and each stimulus is repeated several times (Xue et al., 2010; Lu et al., 2015). STPS can be used to track the representation of this learning process. In NeuroRA, *stps_cal* module is provided to calculate STPS. Users need to input data of neural activities with label information for each trial. Two kinds of labels, labels for the category of the item in each trial and labels for the subject remembering or forgetting the item in each trial, need to input. For STPS computing, we extract the data vector for each trial and calculate the correlation matrix among all the trials. Then we extract the correlation coefficients (Pearson Correlation) from the correlation matrix under different conditions based on the labels users input. Also, we provide multiple functions to calculate STPS for both EEG-like data and fMRI data. For EEG-like data (using *stps*() function), users can set the time-window (the parameter: *time_win*) and time-step (the parameter: *time_step*) for

customized computing. For instance, if user sets *time_win*=10 and *time_step*=5, the function will calculate the STPS results based on each time-window consisting of ten time-points, and the time step is a time interval consisting of 5 time-points between two time-windows for calculation. For fMRI data, users can set the calculation unit for searchlight calculation (using *stps_fmri*() function) or calculate the result from an ROI by a given mask matrix (using *stps_fmri_roi*() function). For both EEG-like and fMRI data, STPS is calculated for each subject, and users can receive all subjects' STPS results.

Inter-Subject Correlation (ISC)

ISC is used to calculate the similarity of brain activities among multiple subjects (Hasson et al., 2004). In NeuroRA, *isc_cal* module is provided to calculate ISC. We convert multidimensional neural activities into vectors and calculate the correlation coefficients (Pearson Correlation) of all pairs of subjects. Consistent with NPS and STPS, users can calculate ISC for both EEG-like data and fMRI data using NeuroRA. For EEG-like data (using *isc*() function), users can set the time-window (the parameter: *time_win*) and time-step (the parameter: *time_step*) for customized computing. For instance, if user sets *time_win*=10 and *time_step*=5, the function will calculate the ISC results based on each time-window consisting of ten time-points, and the time step is a time interval consisting of 5 time-points between two time-windows for calculation. For fMRI data, users can set the calculation unit for searchlight calculation (using *isc_fmri*() function) or calculate the result from an ROI by a given mask matrix (using *isc_fmri_roi*() function). For both EEG-like and fMRI data, ISC is calculated for each subject, and users can receive all subjects' ISC results.

Representational Similarity Analysis (RSA)

Calculating RDMs: NeuroRA provided *rdm_cal* module to calculate RDMs for data from different modalities (using *bhvRDM*() function for behavioral data, *eegRDM*() for EEG-like data, *fmriRDM*() and *fmriRDM_roi*() functions for fMRI data). In all functions in *rdm_cal* module, we calculate RDM(s) for each subject. If *sub_opt*=0, our functions will return the average RDM(s) for all subjects. If *sub_opt*=1, our functions will return RDM(s) for each subject.

Also, there are several alternative methods such as correlation distance, Euclidean distance, and Mahalanobis distance to construct an RDM. Users can set the parameter *method* to choose which method they like. We reshaped activations under different conditions into feature vectors. For correlation distance (*method*= "*correlation*"), values in each RDM are calculated the 1 – Pearson rho between vectors under two corresponding conditions. For Euclidean distance (*method*= "*euclidean*"), values in each RDM are calculated the Euclidean distance between two vectors, which means the distance

between two points in high-dimensional space. Then we rescale all values to the range [0, 1]. For Mahalanobis distance (*method*= "*mahalanobis*"), values in each RDM are calculated the Mahalanobis distance between vectors under two corresponding conditions and are rescaled to the range [0, 1].

Specifically, for EEG-like data (using *eegRDM*() function), we average the trials under each condition first. For channel-based computing (*chl_opt*=1), we calculate RDMs for each channel and return each channel's results. For time-based computing (*time_opt*=1), we calculate RDMs for each time-window based on parameters (*time_win* and *time_step*) user sets. For instance, if user sets *time_opt*=1, *time_win*=10, and *time_step*=5, the function will calculate the RDMs based on each time-window consisting of ten time-points, and the time step is a time interval consisting of 5 time-points between two time-windows for calculation.

Calculating representational similarity: NeuroRA provides three modules to calculate representational similarity. In *rdm_corr* module, functions are applied to calculate the similarity between two RDMs. Users can choose different functions to get the appropriate similarity index. We first extract the values of the upper half of two RDMs and reformed them into two vectors. *rdm_correlation_spearman*(), *rdm_correlation_pearson*() and *rdm_correlation_kendall*() are used to calculate Spearman correlation, Pearson correlation and Kendall's tau correlation, respectively. These three functions return an *r*-value and a *p*-value. The other two functions, *rdm_similarity*() and *rdm_distance*(), are applied to calculate the cosine similarity and Euclidean distance between two RDMs, respectively. These two functions return the distance.

In *corr_cal_by_rdm* module, we provide functions to conduct complex calculations based on multiple RDMs. Users input RDMs from two different modalities, behavioral RDM and EEG-like RDMs (using *rdms_corr*() function) or behavioral RDM and fMRI RDMs (using *fmrirdms_corr*() function). In these processes, we essentially do the calculation by calling functions in *rdm_corr* module. Setting different values for the parameter method, such as "spearman", "pearson", "kendall", "similarity" and "distance", corresponds to different calculation methods for computing the similarity among RDMs.

In *corr_cal* module, we provide functions to obtain similarities from data from different modalities (using *bhvANDeeg_corr*() function for behavioral and EEG-like data, using *bhvANDfmri_corr*() function for behavioral and fMRI data, using *eegANDfmri_corr*() function for EEG-like and fMRI data). The implementation of this module is combining *rdm_cal* module and *corr_cal_by_rdm* module. We use *rdm_cal* module to calculate RDMs based on neural data. Then we use *corr_cal_by_rdm* module to get the

representational similarity between the two modalities.

Statistical Analysis

Statistical analysis is a necessary step after conducting various representational analyses. Due to different shapes of outputs corresponding to different computing, NeuroRA provides necessary functions to realize statistical analysis for different representational analysis results in *stats_cal* module (see Table S2). In all functions, all samples in input maps of correlation coefficients from subjects are tested against zero for significance based on *stats* module in Scipy. Accordingly, these functions will return a *t*-value map and a *p*-value map reflecting the statistical results.

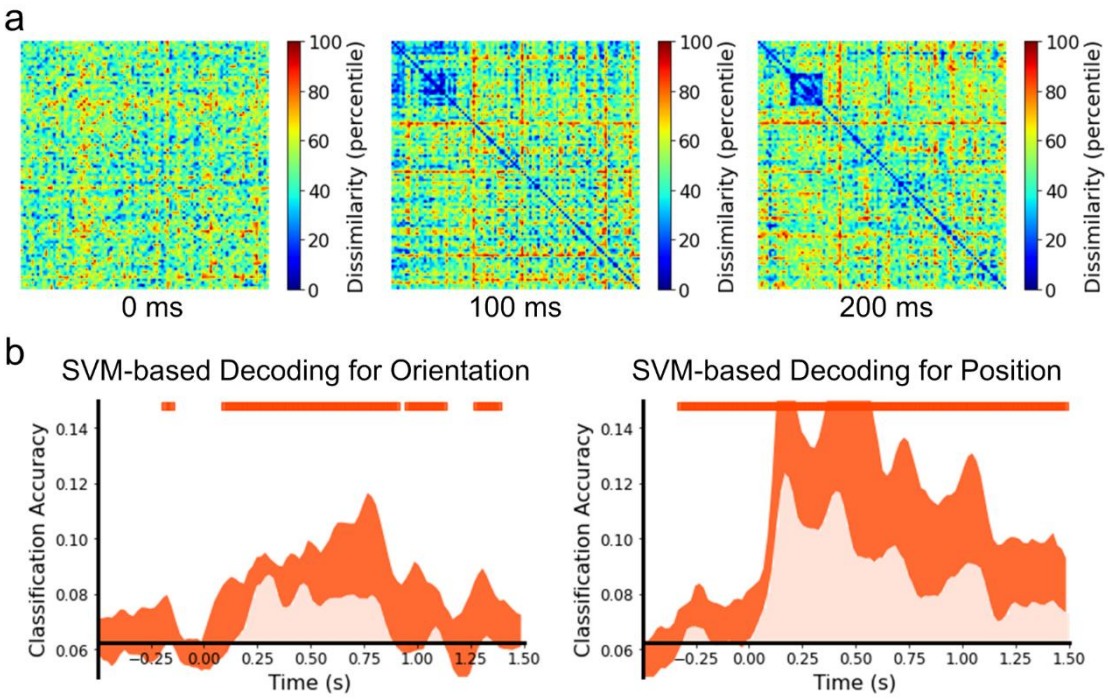

Figure S1 SVM-based results compared with results using NeuroRA in Figure 8a and Figure 8c. (a) RDMs of the first three subjects, calculated by SVM-based classification (Cichy et al., 2014) and plotted by *neurora.rsa_plot.plot_rdm*(). Cichy et al. used the classification accuracy of two categories as the value corresponding to two different conditions in RDM. (b) Orientation and position decoding results of the first five subjects in experiment 2 by SVM-based classification (Bae and Luck, 2018). The results were reproduced by codes rewritten in Python. In the two rightmost plots, the small red rectangles inside the plotting area and red shadow indicate $p < 0.05$; line width reflects ± SEM.