Tutorial of NeuroRA Version 1.0.4

The Tutorial of NeuroRA provides information on how to use the NeuroRA including its easy-to-use functions.

Before you read it, we assume that you are familiar with Python, especially the matrix operations based on Numpy. If not, don't worry. You only need to spend a little time learning the basic Python syntax and this toolkit is easy to understand.

If there is anything wrong, difficult to understand or having any useful advice during reading it, you can contact me (zitonglu1996@gmail.com) and I will be happy and thankful to know about it.

Written by **Zitong Lu**

August, 2019

Institute of Cognitive Neuroscience

East China Normal University

This tutorial consists of these parts:

- Introduction
- Calculate the RDM (Representational Dissimilarity Matrices)
- Calculate the correlation coefficient between RDMs
- Visualization for results
- Save as a NIfTI file (for fMRI)
- Others

Part 1: Introduction

NueorRA is a Python toolbox for multimode neural data representational analysis.

Neural Data Representational Analysis

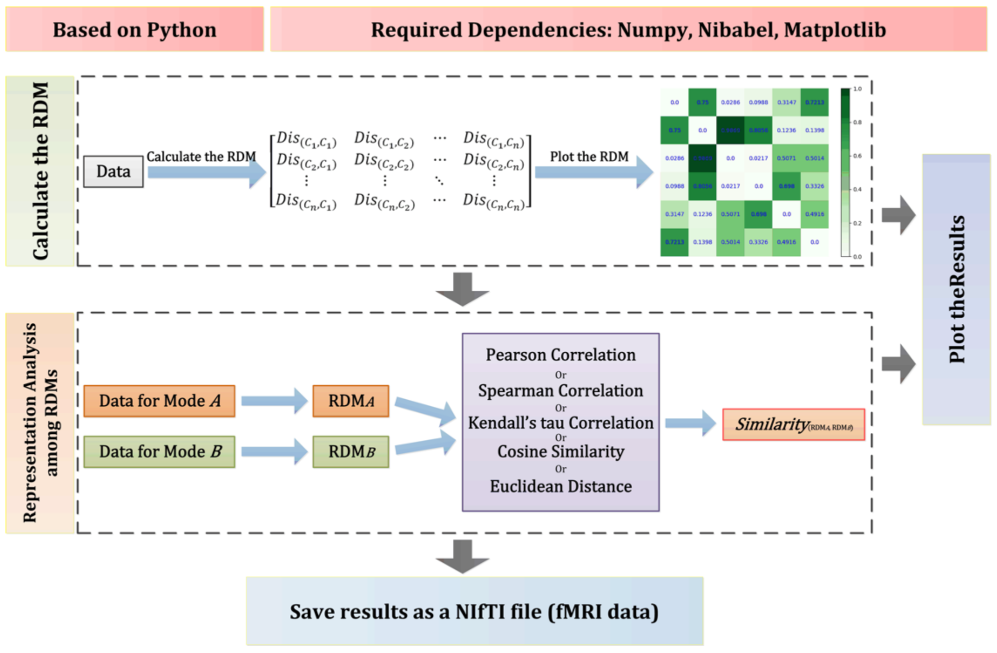
NeuroRA

https://zitonglu1996.github.io/NeuroRA by Zitong Lu

Overview

Representational Similarity Analysis (RSA) has become a popular and effective method to measure the representation of multivariable neural activity in different modes.

NeuroRA is an easy-to-use toolbox based on Python, which can do some works about RSA among nearly all kinds of neural data, including behavioral, EEG, MEG, fNIRS, ECoG, electrophysiological and fMRI data.


Based on Python
Required Dependencies: Numpy, Nibabel, Matplotlib
Calculate the RDM
Data
Calculate the RDM
$Dis_{(C_1,C_1)}$ $Dis_{(C_1,C_2)}$ ⋯ $Dis_{(C_1,C_n)}$
$Dis_{(C_2,C_1)}$ $Dis_{(C_2,C_2)}$ ⋯ $Dis_{(C_2,C_n)}$
$Dis_{(C_n,C_1)}$ $Dis_{(C_n,C_2)}$ ⋯ $Dis_{(C_n,C_n)}$
Plot the RDM
Plot theResults
Representation Analysis among RDMs
Data for Mode A
Data for Mode B
RDMA
RDMB
Pearson Correlation
Or
Spearman Correlation
Or
Kendall's tau Correlation
Or
Cosine Similarity
Or
Euclidean Distance
Similarity(RDMA, RDMB)
Save results as a NIfTI file (fMRI data)

Installation

- *pip install NeuroRA*

Required Dependencies:

Numpy*: a fundamental package for scientific computing*
Matplotlib*: a Python 2D plotting library*
NiBabel*: a package prividing read +/- write access to some common medical and neuroimaging file formats*

Part 2: Calculate the RDM

Module ***rdm_cal.py***:

◆ bhvRDM(bhv_data, sub_opt=0, data_opt=1)

A function for calculating the RDM based on behavioral data

Parameters:

bhv_data:	The behavioral data. If data_opt=0, the shape of bhv_data must be [n_cons, n_subs]. n_cons, n_subs represent the number of conditions & the number of subjects.
sub_opt:	Calculate the RDM for each subject or not. 1 or 0.
data_opt:	If data_opt=1, each subject's each trial has a value of data. If data_opt=0, each subject has a value of data, here ignore the effect of trials.

Returns:

rdm/rdms:	If sub_opt=0, return only one rdm (shape: [n_cons, n_cons]). If sub_opt=1, return rdms (shape: [n_subs, n_cons, n_cons]).

◆ eegRDM(EEG_data, sub_opt=0, chl_opt=0, time_opt=0)

A function for calculating the RDM based on EEG/MEG/fNIRS data

Parameters:

EEG_data:	The EEG/MEG/fNIRS data. The shape of EEG_data must be [n_cons, n_subs, n_trials, n_chls, n_ts]. n_cons, n_subs, n_trials, n_chls, n_ts represent the number of conditions, subjects, trials, channels, frequencies and time-points.
sub_opt:	Calculate the RDM for each subject or not. 1 or 0.
chl_opt:	Calculate the RDM for each channel or not. 1 or 0.
time_opt:	Calculate the RDM for each time-point or not. 1 or 0.

Returns:

rdm/rdms:	If sub_opt=0 and chl_opt=0 and time_opt=0, return only one

rdm (shape: [n_cons, n_cons]). If sub_opt=0 and chl_opt=0 and time_opt=1, return rdms (shape: [n_ts, n_cons, n_cons]). If sub_opt=0 and chl_opt=1 and time_opt=0, return rdms (shape: [n_chls, n_cons, n_cons]). If sub_opt=1 and chl_opt=0 and time_opt=0, return rdms (shape: [n_subs, n_cons, n_cons]). If sub_opt=0 and chl_opt=1 and time_opt=1, return rdms (shape: [n_chls, n_ts, n_cons, n_cons]). If sub_opt=1 and chl_opt=0 and time_opt=1, return rdms (shape: [n_subs, n_ts, n_cons, n_cons]). If sub_opt=1 and chl_opt=1 and time_opt=0, return rdms (shape: [n_subs, n_chls, n_cons, n_cons]). If sub_opt=1 and chl_opt=1 and time_opt=1, return rdms (shape: [n_subs, n_chls, n_ts, n_cons, n_cons]).

◆ ecogRDM(ele_data, chls_num, opt=”all”)

A function for calculating the RDM based on ECoG/electrophysiological data

Parameters:

ele_data:	The ECoG/electrophysiological data. The shape of ele_data must be [n_cons, n_trials, n_chls, n_ts]. n_cons, n_trials, n_chls, n_ts represent the number of conditions, trials, channels, frequencies and time-points.
chls_num:	The number of channels of the data.
opt:	Calculate the RDM for each channel or for each time-point or not. “channels” or “time” or “all”.

Returns:

rdm/rdms:	If opt=”channels”, return rdms (shape: [n_chls, n_cons, n_cons]). If opt=”time”, return rdms (shape: [n_ts, n_cons, n_cons])

◆ fmriRDM(fmri_data, ksize=[3, 3, 3], strides=[1, 1, 1])

A function for calculating the RDM based on fMRI data

Parameters:

fmri_data:	The fmri data. The shape of fmri_data must be [n_cons, n_subs, nx, ny, nz]. n_cons, n_subs, nx, ny, nz represent the

number of conditions, the number of subjects, the size of the fMRI data.

ksize:	ksize=[kx, ky, kz] represents that the calculation unit contains k1*k2*k3 voxels.
strides:	strides=[sx, sy, sz] represents the moving steps along the x, y, z.

Returns:

rdms:	Return rdms for each calculation unit. The shape of rdms is [n_x, n_y, n_z, n_cons, n_cons]. Here, n_x, n_y, n_z represent the number of calculation units along the x, y, z.

Part 3: Calculate the Correlation Coefficient

Module ***rsa_corr.py***:

◆ rsa_correlation_spearman(RDM1, RDM2)

A function for calculating the Spearman correlation coefficient between two RDMs

Parameters:

RDM1:	The shape of RDM1 must be [n_cons, n_cons].
RDM2:	The shape of RDM2 must be [n_cons, n_cons].

Returns:

corr:	The corr contains two values: correlation coefficient and p-value. The shape of corr is [2,].

◆ rsa_correlation_pearson(RDM1, RDM2)

A function for calculating the Pearson correlation coefficient between two RDMs

Parameters:

RDM1:	The shape of RDM1 must be [n_cons, n_cons].
RDM2:	The shape of RDM2 must be [n_cons, n_cons].

Returns:

corr:	The corr contains two values: correlation coefficient and p-value. The shape of corr is [2,].

◆ rsa_correlation_kendall(RDM1, RDM2)

A function for calculating the Kendalls tau correlation coefficient between two RDMs

Parameters:

RDM1:	The shape of RDM1 must be [n_cons, n_cons].
RDM2:	The shape of RDM2 must be [n_cons, n_cons].

Returns:

corr:	The corr contains two values: correlation coefficient and p-value. The shape of corr is [2,].

◆ rsa_similarity(RDM1, RDM2)

A function for calculating the Cosine Similarity between two RDMs

Parameters:

RDM1:	The shape of RDM1 must be [n_cons, n_cons].
RDM2:	The shape of RDM2 must be [n_cons, n_cons].

Returns:

similarity:	The Cosine Similarity.

◆ rsa_distance(RDM1, RDM2)

A function for calculating the Euclidean Distances between two RDMs

Parameters:

RDM1:	The shape of RDM1 must be [n_cons, n_cons].
RDM2:	The shape of RDM2 must be [n_cons, n_cons].

Returns:

dist:	The Euclidean Distance.

◆ rsa_distance(RDM1, RDM2, iter=1000)

A function for permutation test between two RDMs

Parameters:

RDM1:	The shape of RDM1 must be [n_cons, n_cons].

RDM2:	The shape of RDM2 must be [n_cons, n_cons].
iter:	The number of iterations.

Returns:

p:	The p-value.

Module ***corr_cal.py***:

◆ bhvANDeeg_corr(bhv_data, EEG_data, sub_opt=0, bhv_data_opt=1, chl_opt=0, time_opt=0, method="spearman")

A function for calculating the Similarity/Correlation Coefficient between behavioral data and EEG/MEG/fNIRS data

Parameters:

method:	"spearman" or "pearson" or "kendall" or "similarity" or "distance". They correspond to different methods of calculation.

Returns:

corr/corrs:	The correlation coefficients corresponding to the RDMs.

◆ bhvANDecog_corr(bhv_data, ele_data, chls_num, ecog_opt="allin", method="spearman")

A function for calculating the Similarity/Correlation Coefficient between behavioral data and ECoG/electricophysiological data

Returns:

corr/corrs:	The correlation coefficients corresponding to the RDMs.

◆ bhvANDfmri_corr(bhv_data, fmri_data, bhv_data_opt=1, ksize=[3, 3, 3], strides=[1, 1, 1], method="spearman")

A function for calculating the Similarity/Correlation Coefficient between behavioral data and fMRI data

Returns:

corrs:	The correlation coefficients corresponding to the RDMs. The shape of corrs is [n_x, n_y, n_z, 2].

◆ eegANDfmri_corr(eeg_data, fmri_data, chl_opt=0, ksize=[3, 3, 3], strides=[1, 1, 1], method="spearman")

A function for calculating the Similarity/Correlation Coefficient between EEG/MEG/fNIRS data and fMRI data

Returns:

corrs:	The correlation coefficients corresponding to the RDMs.

Module ***corr_cal_byrdm.py***:

◆ eegrdms_corr(demo_rdm, EEG_rdms, method="spearman")

A function for calculating the Similarity/Correlation Coefficient between RDMs based on EEG/MEG/fNIRS data and a demo RDM

Parameters:

demo_rdm:	The shape must be [n_cons, n_cons].
EEG_rdm:	The shape must be [n_ts, n_cons, n_cons] or [n_chls, n_cons, n_cons].

Returns:

corrs:	The correlation coefficients corresponding to the RDMs.

◆ fmrirdms_corr(demo_rdm, fMRI_rdms, method="spearman")

A function for calculating the Similarity/Correlation Coefficient between RDMs based on fMRI data and a demo RDM

Parameters:

demo_rdm:	The shape must be [n_cons, n_cons].

fmri_rdm:	The shape must be [n_x, n_y, n_z, n_cons, n_cons].

Returns:

corrs:	The correlation coefficients corresponding to the RDMs. The shape of corrs is [n_x, n_y, n_z, 2].

Part 4: Visualization for Results

Module ***rdm_plot.py***:

◆ plot_rdm_1(rdm) / plot_rdm_2(rdm)

Two functions for plotting the RDM

Module ***corr_plot_time.py***:

◆ plot_corrs_bytime(corrs, time_unit=[0, 1])

A function for plotting the correlation coefficients by time sequence

Parameters:

corrs:	corrs represent the correlation coefficients point-by-point. Its shape must be [ts, 2].
Time_unit:	time_unit=[start_t, t_step]. Here, start_t represent the start time and t_step represent the time between two adjacent time-points.

Part 5: Save as a NIfTI file (for fMRI)

Module ***corr_to_nii.py***:

◆ corr_save_nii(corrs, filename, affine, size=[60, 60, 60], ksize=[3, 3, 3], strides=[1, 1, 1], p=1, r=0, similarity=0, distance=0)

A functions for saving the correlation coefficients as a .nii file

Parameters:

corrs:	corrs represent the correlation coefficients. Its shape must be [n_x, n_y, n_z, 2].
filename:	The filename of the NIfTI file. Don't need a suffix.
affine:	An affine array that tells you the position of the image array data in a reference space.
p, r, similarity, distance:	They represent the threshold value for calculation.

Returns:

img_nii:	The matrix form of the NIfTI file.

Part 6: Others

Module ***stuff.py***:

◆ limtozero(x)

A functions for zeroing the value close to zero.

Parameters:

x:	A value.

Returns:

0

◆ get_affine(file_name)

A functions for getting the affine.

Parameters:

file_name:	The file_name of a fMRI file.

Returns:

affine