Partial correlation-based representational similarity analysis add-on for SPM

Roberto Viviani[1,2]

1: Institute of Psychology, University of Innsbruck, Austria
2: Psychiatry and Psychotherapy Clinic III, University of Ulm, Germany

Roberto Viviani
University of Innsbruck
Institute of Psychology
Innrain 52
6020 Innsbruck, Austria
roberto.viviani@uibk.ac.at

Software manual, March-June 2021

Partial correlation-based representational similarity analysis add-on for SPM

Purpose

The present SPM add-on package implements the computation of representational similarity analyses for functional imaging data in a searchlight sampling scheme, using a partial correlation approach to adjust for sources of bias that may be present due to the model design or due to other sources. It also provides diagnostics for the existence of bias in these analyses, with or without partial correlation. This package makes the methods described in the following report publicly available:

Viviani, R. (2021). Overcoming bias in representational similarity analysis. *arXiv preprint* arXiv:2102.08931.

That report contains the technical rationale for the techniques implemented here and may be cited if needed. Representational similarity analysis was originally described in several publications, including Kriegeskorte et al. (2008), and made available through the toolbox described in Nili et al. (2014). In the following, it will be assumed that the reader is familiar with the concepts of RSA, as formulated in publications such as Kriegeskorte et al. (2008). This writing is a slightly edited version of the description of the package developed for internal use by non-programmer members of the team.

This software package is distributed under the GNU General Public Licence (reproduced at the end of the present document). Its use implies acceptance of the terms specified there, and an understanding that there is no warranty for the software and its fitness and that no formal support or maintenance is provided or implied with this distribution.

Requirements

This software requires:

MATLAB 2013 or higher

the statistics toolbox distributed with MATLAB

SPM12.

SPM is a free software for the analysis of neuroimaging data developed at the The Wellcome Centre for Human Neuroimaging, London, available at https://www.fil.ion.ucl.ac.uk/spm/. SPM is also needed to estimate the coefficients of the general linear model that are subsequently analyzed by the present add-on.

The hardware requirements are those of functional imaging analyses (at least 8MB of RAM, sufficient hard disk space).

Finally, the user is required to be familiar with modelling functional imaging data with the SPM package, with the basic logic of representational analyses (Kriegeskorte et al. 2008), and to be familiar with the process of interacting with MATLAB by entering commands in the console.

Distribution and installation

This software consists of the following MATLAB files:

`rsa_rsm_searchlight.m`	computes the representational similarity analysis
`rsa_create_rsm.m`	creates representational similarity maps for use by `rsa_rsm_searchlight`
`rsa_defaults.m`	a set of defaults values for the computation of the analysis
LICENCE.txt	terms of use

The following files are also part of the package, although as a rule will not be used directly.

`rsa_algorithms.m`	the functions computing the correspondence (correlation) between similarity measures
`rsa_rsmio.m`	a collection of helper functions to handle representational similarity maps
`rsa_utils.m`	a collection of utility functions, mainly handling I/O
`ask_args.m`	a utility to assist in invoking `rsa_rsm_searchlight`
`spm12_searchlight.m`	a modification of the routine distributed with SPM12 for sampling searchlight voxels from volumes, as allowed by the GNU General Public Licence. The modification allows specifying path and names of the files in which the output is saved. The author of this package is responsible for every error that may have been introduced by the modification.

To install the package, copy/unzip these files in the folder of choice. Make sure the folder is in MATLAB's path.

Representational similarity analysis

Representational similarity analysis (RSA, Kriegeskorte et al. 2008) seeks evidence of cortical encoding of properties of stimuli by assessing the concordance between the matrix encoding the pairwise similarity of stimuli and a matrix encoding the similarity of the brain responses to the same stimuli. In the original formulation of Kriegeskorte et al. (2008), the concordance was based on assessing the correlation between measures of dissimilarity. Here, similarity is used instead because it is the most natural representation of the concordance of brain responses through their covariance. Because it is usually possible to convert between measures of similarity and dissimilarity, this alone does not constitute a fundamental difference in approach. This software package computes the concordance between the similarity of stimuli (provided by the user) and a measure of covariance of response in a voxel and its neighborhood using the searchlight method (Kriegeskorte et al. 2006).

A downside of RSA is the bias introduced in the assessment of concordance by the lack of orthogonality of regressors and/or the lack of independence of the scans. Both are usually present in functional imaging designs. The present add-on implements basic diagnostics for bias and a partial correlation approach to alleviate this problem, as described in Viviani (2021). Various terms can be specified for partialling out in the correlation, including the estimated covariance of the model coefficients involved in the analysis ('BCov') or an estimate of these covariance from the masked volume ('SCov') or simply the

cross-products of these coefficients ('BB'), or any user-specified term (see below for other implementational differences from the original package by Nili et al. 2014). A problem in correcting for the bias due to the design is obtaining a sufficiently good estimate of the covariance of the model coefficients, a problem of difficult solution in the presence of lack of independence of successive scans (as is the case in fMRI). Because of this limitation, there is no guarantee that RSA bias can be overcome.

Data preparation

To apply the present package the user is required to preliminarily fit a linear model in each subject. The linear model will contain a predictor (column of the design matrix) for every stimulus or stimulus class. In addition, a representational similarity map for the same predictors should be saved to disk as discussed in the next section. Thus, there is a one-to-one relationship between the predictors of the model and the stimuli or stimulus classes whose properties are characterized by representational similarity. In typical RSA analyses, this translates into a design matrix for the fMRI data in which each trial or presentation of the stimulus is modelled by its own regressor. In this case, ordering the regressors in the design matrix in the temporal order in which the trials occurred has the advantage that the covariance arising from the nonorthogonality due to the adjacency of the trials becomes evident in maps of estimates of this confound as higher values near the diagonal. One of the techniques to deal with this confound is only available if the designed matrix is ordered temporally (off-diagonal offset, see below).

At the end of the data preparation, there will be a folder for each of the subjects included in the study, and each folder should contain the following files:

beta_00xx.nii	the set of beta images computed by SPM containing the parametric maps of the coefficients of the model. However, the present software can also be configured to accept files other than these as input if necessary (for example, contrast images).
mask.nii	the mask computed by SPM specifying the voxels containing the model fit. In alternative, the software can be configured to accept a file with another name located in each directory as an alternative mask, or one single file providing a single mask for the whole dataset.
RSM.mat	the file containing the representational maps of the stimuli. This file may have any name. In alternative, a single file for the whole dataset may be provided if the stimuli and their presentation were the same for all subjects.
spm.mat	the file computed by SPM containing details of the model and the fit. This file is only needed if using 'BCov' as a confound or similarity map in the analysis (see below).

With the exception of the representational maps of the stimuli, all these files are automatically computed by SPM when a model is estimated. Creation of the representational maps by the user is discussed in the next section.

Creation of the representational similarity maps

To compute the similarity analysis, `rsa_rsm_searchlight` will load a file containing the **similarity maps** of the stimuli, the **similarity maps file**. There can be one such file per subject (located in the folder of the beta images for that subject) or one single file for all subjects

(when the similarity maps are identical in all subjects). Each similarity map file can contain one or more similarity maps. Each map is stored in an associative array (MATLAB's `struct`): the **similarity map structure**. Its most important fields are:

`name`	the name of the map
`rsm`	a matrix containing the similarity map. Only the upper diagonal part of this matrix is used in the code

This `struct` can be created by calling the function `rsa_create_rsm`:

```
> rsm = rsa_create_rsm(features, 'map_name');
```

In this call, `features` is a cell matrix, where each column is a property defining the similarity of the items, and each row is an item, whereas `map_name` is the name of the map. Let us assume, for example, that the experiment consists in the presentation of facial expressions, and that we are classifying the stimuli in terms of two features, the emotion represented in the expression and its valence. Then the variable `features` may assume the following form:

```
> features = {'sad',     'negative';
              'disgust',  'negative';
              'happy',    'positive';
              'sad',      'negative';
              'fearful',  'negative';
              'happy',    'positive';
              ...
              };
```

with one row per stimulus. `rsa_create_rsm` will create the representational map in the `rsm` field of the similarity map structure based on the properties in common between stimuli. The representational map will be a matrix of order equal to the number of rows of `features`.

Using strings to define qualitative properties is convenient, but it is also possible to use a matrix of integral numbers to represent index variables (as in R, where a number can represent a factor level). An equivalent representation of `features` would be given by the following:

```
> features = uint8([1, 1;
                    2, 1;
                    3, 2;
                    1, 1;
                    4, 1;
                    3, 2;
                    ...
                    ]);
```

Any number of maps may be defined and collected into an array of similarity map structures. In the following example, an array of two similarity maps is created, based on the properties 'emotion' and 'sex' of the stimuli.

```
> sex =      {'female',  'young';
              'male',     'elderly';
              'male',     'young';
              'female',   'other';
              'female',   'elderly';
              ...
              };
```

```
> rsm(1) = rsa_create_rsm(features, 'emotion');
> rsm(2) = rsa_create_rsm(sex,      'sex');
```

This array should then be saved to disk in the similarity maps file, which will be read by `rsa_rsm_searchlight`:

```
> save('RSM.mat', 'rsm');
```

Here, `RSM.mat` is a similarity maps file saved in the current directory, containing an array of two similarity maps, 'emotion', and 'sex'.

At the time of writing, the present package also accepts dissimilarity maps created with the software of the original rsa toolbox (Nili et al. 2014). Internally, these maps are converted into similarity maps by computing `1 - rsm`.

Similarity maps based on quantitative features or on mixed features

It is also possible to define similarity on the basis of a quantitative metric. For example, suppose that we had normative data on the arousal elicited by the faces used in the experiment. We could then define a map based on the arousal properties of the stimuli:

```
> arousal = [1.5;
             2.0;
             0.5;
             3.2;
             1.8;
             ...
             ];
```

When `rsa_create_rsm` encounters a matrix of doubles, it attempts to form a similarity matrix based on the Euclidean distance of the values (if there is only one feature type in a single column) or the correlation (if there are more than one feature types). Care must be taken for features based on the correlation of few values, as they may easily produce NaN's (when the values are identical). Correlations based on few values might be poorly defined. You can control the type of metric used to compute the similarity in this case by providing a third argument to `rsa_create_rsm`:

```
> rsm = rsa_create_rsm(arousal, 'arousal', 'euclidean');
```

Valid metric values are: euclidean, squaredeuclidean, seclidean (standardized Euclidean), mahalanobis, correlation, spearman, cosine, jaccard. If there is only one feature column, only Euclidean-based metrics are defined. For distance metrics (i.e., dissimilarity), the similarity map is defined by `1 - rsm`, i.e. one minus the values given by the metric.

Note that the first argument to `rsa_create_rsm`, which provides the data for the similarity, is a matrix of data all having the same type. Similarity may not be defined based on a mixture of qualitative and quantitative data (in the example at hand, valence and arousal). It is not clear in general how qualitative and quantitative features may be combined to form a similarity matrix. If a specific mixed list of features is present, however, one may define a custom function to compute the similarity, and pass it as an argument to `rsa_create_rsm`. Suppose to have saved the metric function in `mixed_metric.m`. The call then may be made thus:

```
> rsm = rsa_create_rsm(mixed_features, 'emotion', @mixed_metric);
```

The custom metric function is internally passed to MATLAB's `pdist` function; hence, test the custom function with `pdist` before applying it here. To see how the custom metric

function may be coded, see `rsa_create_rsm.m`, which contains two functions of this kind, used for the qualitative features of the previous section.

Computation of the similarity analysis

Before the similarity analysis may be computed, the data should have been prepared as described above in the section 'Preparation of data'. The similarity analysis is conducted by the function `rsa_rsm_searchlight`, which may be invoked as follows:

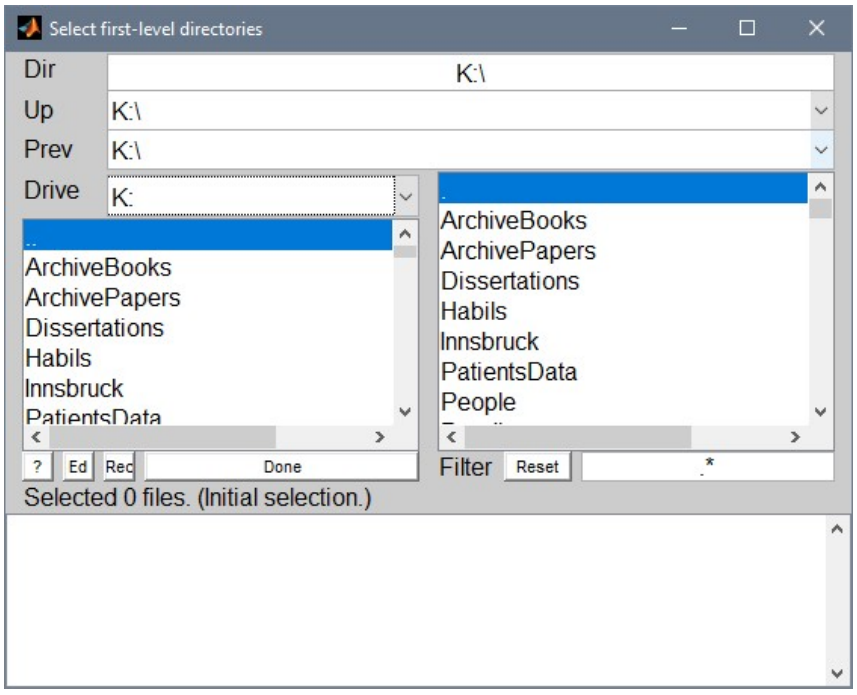
```
> str = rsa_rsm_searchlight(args);
```

where `args` is a `struct` variable containing the arguments of the call. The best way to fill in this associative array is to call `rsa_rsm_searchlight` with the special parameter `ask_args`:

```
> args = rsa_rsm_searchlight(ask_args);
```

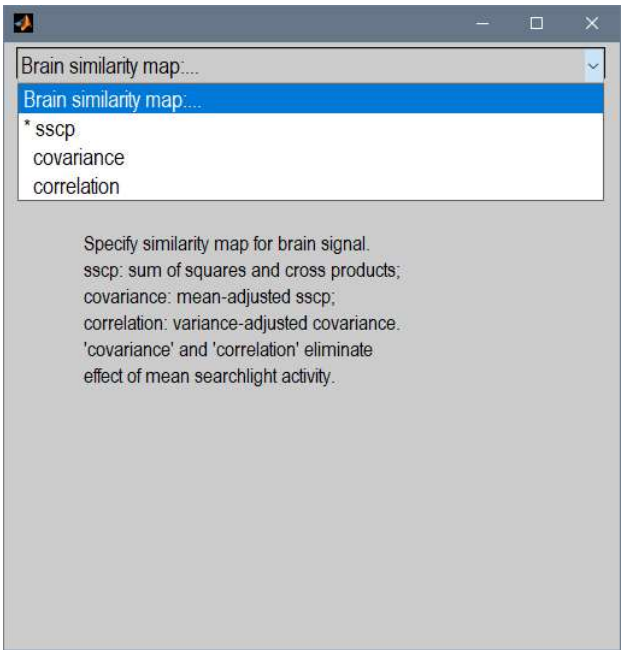
This call does not execute the analysis; it only collects the parameters of the call in `args` (this is similar to the 'batch' window in SPM12; the role of the information in the window is played by the content in the fields of the `args` variable). One could also fill these fields manually. When called in this way, `rsa_rsm_searchlight` prompts the user to supply the information required to execute the function, such as the folders where the beta images, the similarity map files and (if required) the mask files and spm.mat are stored, and the parameters of the RSA.

The user is first asked to select the folders containing the estimated model.



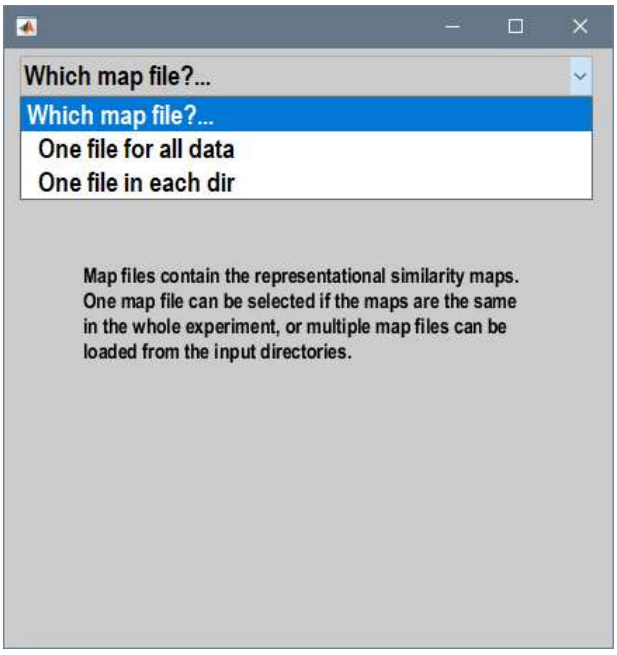
This is a standard SPM dialog, and will look different on your system. Select all folders that will be analysed in the model.

In the next step, the user is prompted to select the type of similarity map to capture similarity in the brain signal.



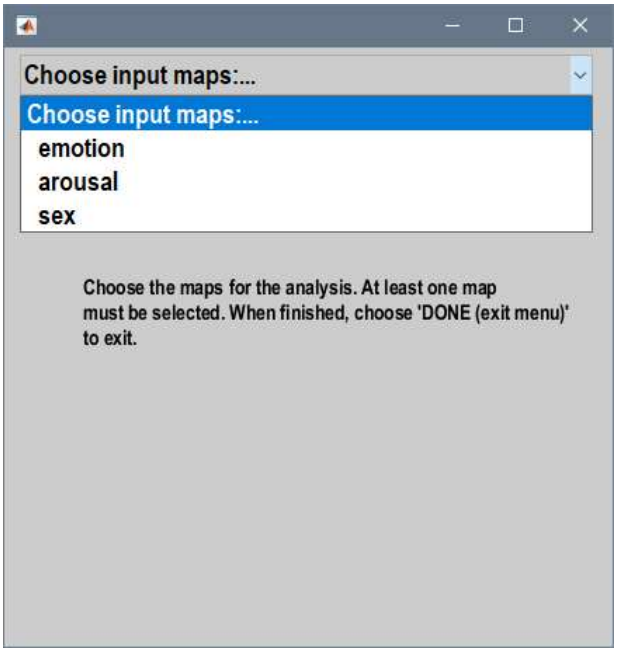
The sum of squares and cross products ('sscp') also includes mean activity in the similarity representation, which is not necessarily what the original representational analysis was supposed to capture (Kriegeskorte et al. 2008). To exclude mean activity, choose 'covariance' or 'correlation'. When 'correlation' is chosen, the adjustment for the design confound may not work as well as in the other options.

In the next step, the user is asked to indicate whether the similarity maps are contained in the folders of the estimated models, or whether a single similarity maps file will be used for all subjects:

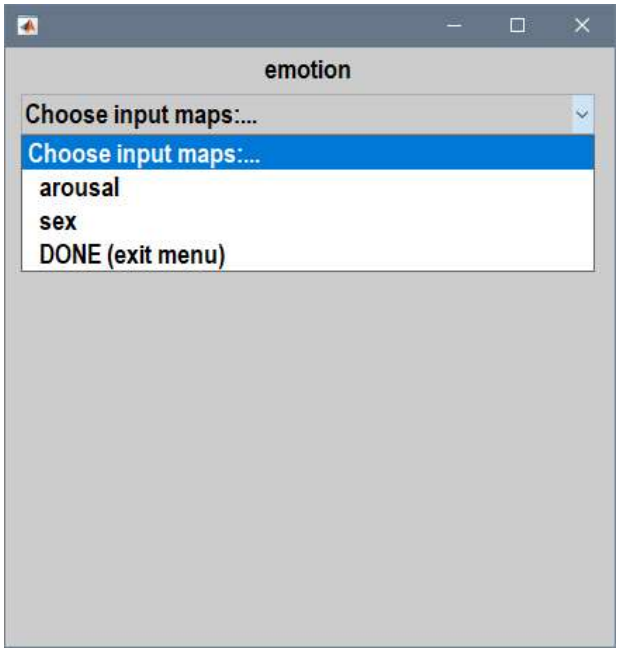


Select the option that is most appropriate. If 'One file for all data' is chosen, one is asked to select this file. If 'One file in each dir' is chosen, one is asked to indicate the name of the similarity maps file, one in each input folder.

In the next step, one is asked to select the maps that will be included in the analysis. These are the maps that were stored in the similarity map files (recall that arrays of maps may be saved in these files). In the example below, three similarity maps were included in the files. At least one map must be selected at this step:



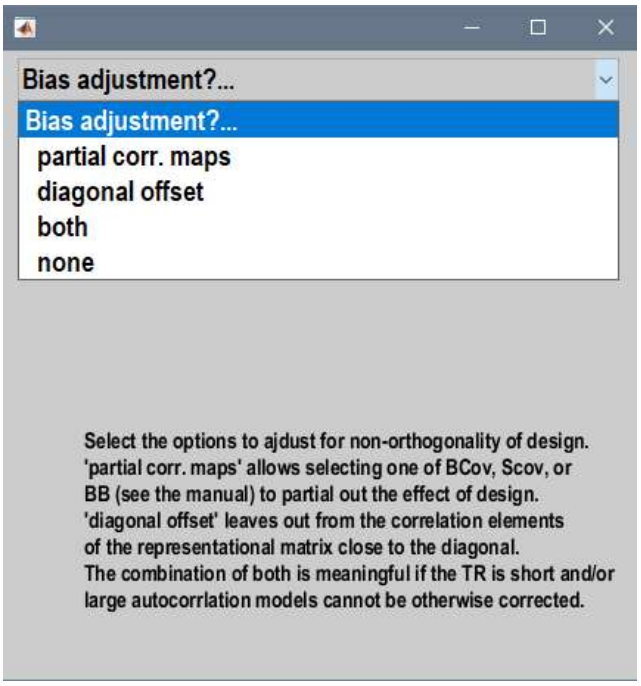
Here, after choosing 'emotion', one has the option to add other maps to the analysis or exit the menu:



For each chosen map, a (partial) correlation analysis will be conducted on the data. One chooses 'DONE (exit menu)' to indicate that one has selected all desired maps. There is no requirement to use all maps saved in the representation similarity files in the analysis. All chosen maps must refer to the same number of stimuli.

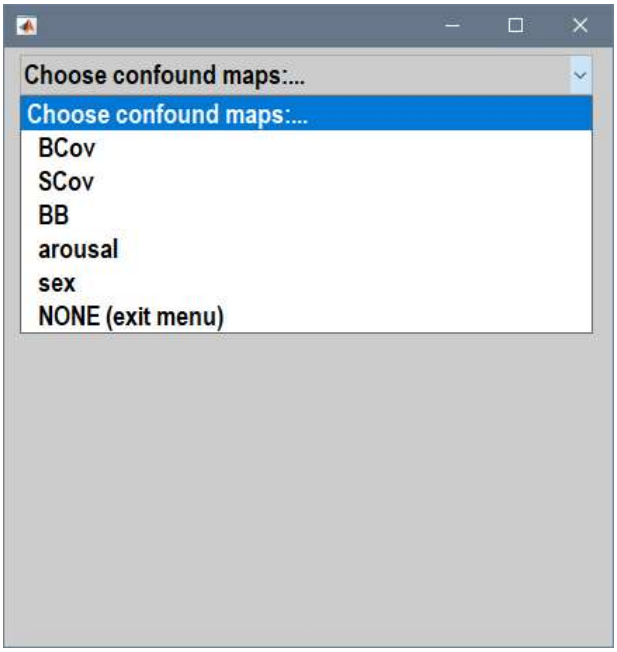
After having chosen 'DONE (exit menu)', one is asked to specify the options for the adjustment of the correlation to redress possible bias due to the design. One has two strategies to do this: use similarity maps as confounds in a partial correlation (such as BCov,

SCov, or BB – see Viviani 2021 for explanations), and to exclude off-diagonal elements from the similarity matrix. A third strategy includes both methods simultaneously.



Important: the off-diagonal offset strategy only makes sense if the design matrix contains a column for each trial, and these columns are arranged in temporal order. Note also that the combination of confound similarity maps such as BCov, SCov, and BB with an off-diagonal offset only makes sense if there is enough information away from the diagonal in these similarity maps to provide a predictor with sufficient signal. You would expect in general the signal in BCov, SCov, and BB to vanish or reduce to noise with increasing distance from the main diagonal, when trials enter the design in temporal order.

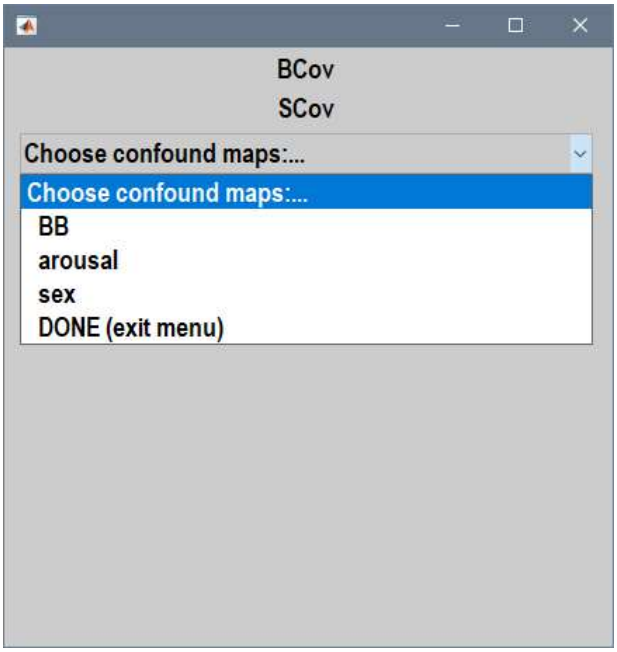
To illustrate, we will choose the option 'both' here. One is then first asked to specify the maps to include in the partial correlation as confounds:



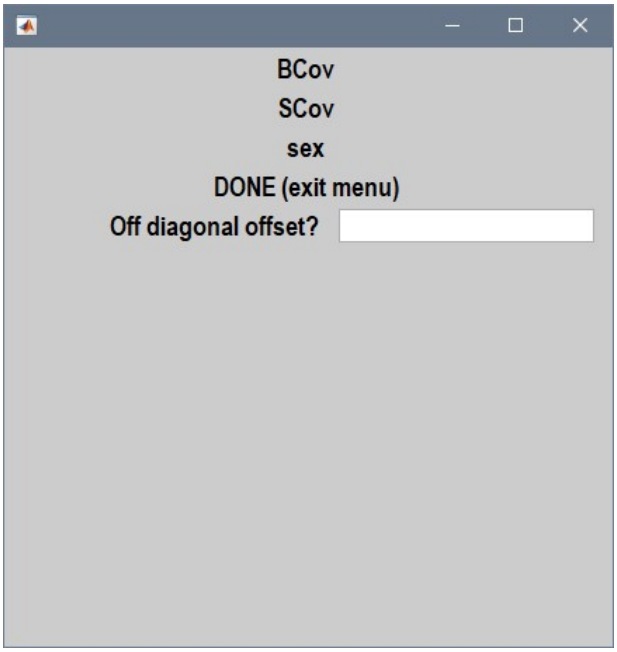
Three possible adjustment for bias are offered at this step: BCov, SCov, and BB (Viviani 2021). BCov is the theoretical covariance of model coefficients when the estimated

autocorrelation term corresponds to the real autocorrelation term. This option constrains using beta images as input volumes below. SCov and BB are similar (with BB tending to be more aggressive) and estimate the covariance of coefficients by averaging it over the whole volume. It is a good idea to select two of these terms, such as BCov and SCov, or BCov and BB, but not SCov and BB simultaneously (given that they are usually very similar).

After having chosen BCov and SCov, it is still possible to choose arousal and sex:

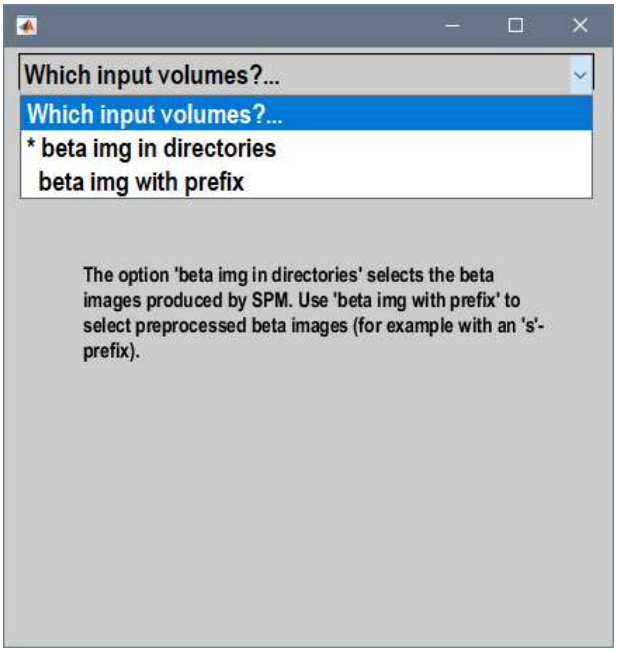


This is because arousal and sex were not included similarity maps at the previous step. The logic here is as when including confound covariates in linear regression. Suppose that we are interested in detecting representation of emotion defined by the emotion in question and its valence. The stimuli will also differ in arousal values and the sex of the person in the stimuli. Adding these features at this step adjusts for representational similarity accidentally induced by these features of the stimuli, viewed as confounds. As before, one chooses 'DONE (exit menu)' when the selection is complete. Here, after adding sex as a confound, we exited the menu. We are then asked to specify the off-diagonal offset:

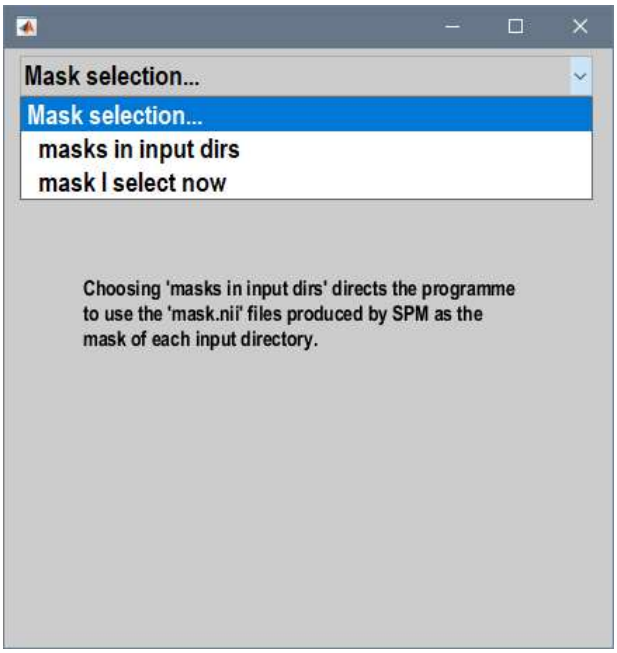


The off-diagonal offset is a positive integer. The value zero has no effect; negative values are not accepted. The appropriate value will generally be a small value such as 1 or 2. See Viviani (2021) for a description and justification of using an off-diagonal offset; remember that this option only makes sense if there is one regressor in the design matrix per trial, and these regressors are arranged in the temporal order of the occurrence of the trials.

The user is then asked to specify the beta images in the input directories:



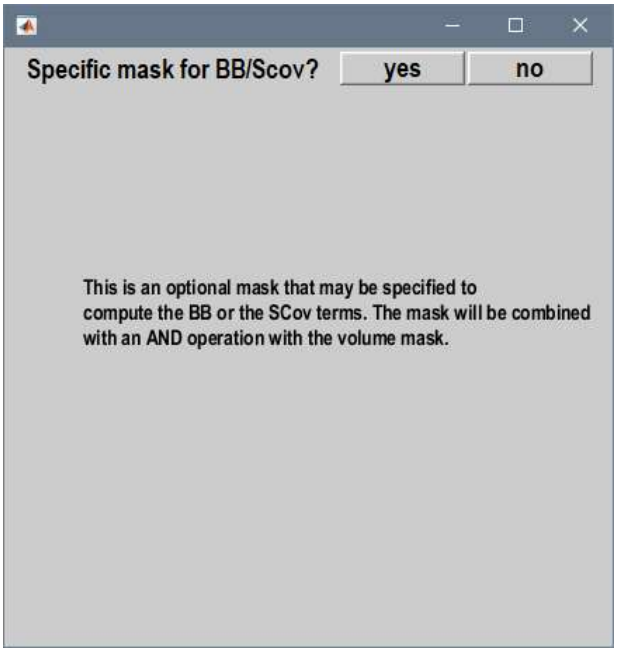
The suggested default value, 'beta img in directories', simply selected the beta images computed by SPM in the input folder. The option 'beta img with prefix' is provided to select preprocessed beta images, for example smoothed images. SPM denotes these files with a prefix. If you choose this latter option, you are asked to specify the prefix (the default value 's' is the one that SPM prepends to volumes if the user does not indicate otherwise). However, the BCov confound option is only compatible with choosing beta images here.



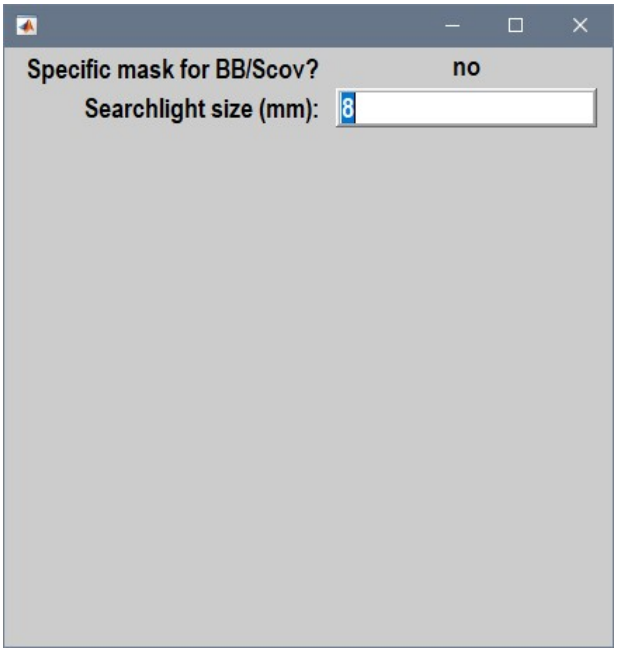
The option that follows refers to the mask in which the representational analysis is conducted. If one chooses 'masks in input dirs', the files mask.nii produced by the SPM

analysis in each folder will be used. Otherwise, the user is prompted to select the mask for all subjects.

A separate mask may be specified for SCov and BB, if these maps were selected. This is because SCov and BB are estimates of covariance of coefficient terms obtained by summing their value in voxels over the whole volume. Giving a separate mask allows one to be selective about the voxels where SCov and BB are estimated (for example, gray and white matter). If a separate mask is given, it will be combined with the mask of the analysis with an AND operator.



One is then prompted to indicate the searchlight size (in mm):



The searchlight will be included in the estimated rsa volume if the number of voxels within it is at least 27. You can change this number by altering the `minsearchlightsize` field in rsa_defaults.m. (You can also get maps of the number of voxels in your searchlight by calling `rsa_rsm_searchlight` with the `Method` field of the arguments set manually to `'slsize'` – see the reference section below).

Finally, one is prompted to specify a model name and the output folder. Both values are optional. If no model name or the default value 'AUTO' is selected, a model name will be created automatically based on the choices of the previous step. If no output folder is selected, the correlation maps will be saved in the input folders.

The call terminates at this point returning in the variable `args` the chosen options, which will look similar to the one show here:

```
args =
              MFile: 'rsa_rsm_searchlight'
            Version: '1.0'
         SPMVersion: 'SPM12'
          Timestamp: '01-Apr-2021 13:11:01'
        Directories: [7×76 char]
             Method: 'Pearson'
       BrainMapType: 'sscp'
            MapFile: 'D:\Scratch\EmotionStudy\RSM.mat'
             MapSel: {'emotion'}
        MapSelPcorr: {'BCov'  'SCov'  'sex'}
      OffDiagOffset: 1
          BetaFiles: []
            BetaIdx: []
           MaskFile: []
      MaskConfound: []
     SearchlightDef: 'sphere'
    SearchlightSize: 8
          ModelName: []
          OutputDir: ''
```

At this point, entering `str = rsa_rsm_searchlight(args)` at the MATLAB prompt will execute the analysis. Before doing that, however, note that there are a number of fields in the variable `args` that do not correspond to any specification of the previous call to this function. These fields correspond to options that have been given a default value, but that may be changed manually before invoking `rsa_rsm_searchlight`. These fields are: Method, BrainMapType, SearchlightDef, and BetaIdx. See the reference section for information about these fields.

Any field may be changed manually before invoking `rsa_rsm_searchlight`. For example, `BrainMapType` determines how the similarity matrix for the brain signal pattern is computed. This setting may be changed manually (see the Reference section below for a list of all available settings):

```
> args.BrainMapType = 'cov';
```

Output

After calling `str = rsa_rsm_searchlight(args)`, the variable `str` contains some informative data on the representational analysis. This variable is a MATLAB associative array (`struct`) with the following fields:

`ModelName`	the name of the model. The parametric maps of the correspondence are saved to disk with this name, followed by the name of the map.
`MapNames`	the names of the map for which the RSA was computed.
`Rsms`	the array of the similarity maps used in the analysis. Each element of the array contains the name of the map, its type, and the map.

`Mns`	the average volume correlation of each map. This is a diagnostic value to assess the bias of the representational similarity analysis.
`Cmx`	the correlation matrix of the off-diagonal elements of the similarity maps used in the analysis. This assesses the extent of the collinearity between these off-diagonal elements.
`Output`	the output directory
`Args`	the argument settings used to compute the analysis
`diagn()`	a function that displays a summary of the possible remaining bias of the correspondence analysis for the similarity maps. For example, if the output was saved in the variable `str`, then entering `str.diagn()` at the prompt will print this summary
`getmap(mname)`	a function to obtain the (average) similarity map indicated by mname. Use `disp()` to retrieve the names of the used maps. For example, `imagesc(str.getmap('emotion'))` displays the (average) similarity map for emotion in the example of the previous section.
`disp()`	prints out a summary of the settings and maps in the RSA
`boxplot()`	a function that displays boxplots of the average volume correlations. Average correlations with zero in the interquartile range are normally acceptable
`mapcorr()`	computes the average concordance between the off-diagonal terms of two named maps. The maps that may be used here are those named by `disp()` (i.e., the maps that were used in the analysis). The concordance is computed based on the settings of the present RSA analysis (method, off-diagonal offset, etc.). For example, `str.mapcorr('BCov', 'emotion')` prints out the average concordance between the emotion similarity maps and the theoretical covariance of the model coefficients.

The correspondence maps (the partial correlation between the similarity of the stimuli and of the brain response) are saved in the output directory, or (if no output directory was given) in the directories of the beta images.

Differences between the present package and the representation similarity software described in Nili et al. (2014)

To assess concordance of similarity, the present package computes the sum of squares and cross products (sscp) of the coefficients of the modelled events involved in the analysis in the voxels of the searchlight. It then computes a (partial) correlation between the off-diagonal elements of this matrix and the off-diagonal elements of the similarity maps of the stimuli.

At the time of writing, the software described in Nili et al. (2014) uses Matlab's `pdist` function to compute the dissimilarity map for the data. This function computes 1 – **P**, where **P** is the correlation matrix of the input after row centering (i.e., by voxel), to represent dissimilarity. To assess concordance, it computes the *rank* correlation between the off-diagonal elements of 1 – **P** and those of the dissimilarity maps of the stimuli.

The present package avoids rank correlation because it leads to inefficient adjustment of confound terms (it may be specified in the `Method` field of the argument structure to the

function, however). Transformation of the sscp into a correlation matrix is also available as a setting of the `BrainMapType` field of the argument structure (see the Reference section below).

Reference

Fields to specify the parameters of the analysis

Fields for which the descriptor '(opt)' is given are optional and may remain empty.

`MFile` (read only) the m file containing the software to compute the analysis

`Version` (read only) the version of the software

`SPMVersion` (read only) the SPM version used

`Timestamp` (read only) the time the function was called

`Directories` the folders containing the beta images and other data (char matrix)

`Method` the method used to assess the correspondence between the similarity maps of the stimuli and the similarity of the brain response signal. Possible values are 'Pearson', 'Spearman', or 'regression'. The use of 'Spearman' when adjusting for bias is not recommended. If this field is set to 'slsize', then maps of the number of voxels in the searchlight are computed instead of a similarity analysis. If empty, defaults to 'Pearson'. The code that does this is centralized in `rsa_algo.m` for inspection.

`BrainMapType` one of 'sscp' (for sum of squares and cross-products), 'cov' (for covariance), 'cor' (for correlation). This value specifies how the similarity map of the brain signal is formed: sum of squares and cross-products, covariance, or correlation. If 'cor' is specified, also the confound maps BB and SCov are transformed to correlations. This field is ignored if Method is 'regression'. Defaults to 'sscp'. If $\hat{\mathbf{B}}$ is the matrix of the estimated coefficients from the model (of size number of beta coefficients × voxels in the searchlight matrix), then the similarity maps are computed as follows:

$$\text{sscp: } v^{-1}\hat{\mathbf{B}}\hat{\mathbf{B}}' == v^{-1}\sum_i \hat{\boldsymbol{\beta}}_i\hat{\boldsymbol{\beta}}_i'$$

$$\text{cov: } \mathbf{S} = v^{-1}\sum_i (\hat{\boldsymbol{\beta}}_i - \bar{\boldsymbol{\beta}})(\hat{\boldsymbol{\beta}}_i - \bar{\boldsymbol{\beta}})',\ \bar{\boldsymbol{\beta}} = v^{-1}\sum_i \hat{\boldsymbol{\beta}}_i$$

$$\text{cor: } \mathbf{P} = \mathbf{D}^{-1/2}\mathbf{S}\mathbf{D}^{-1/2},$$

where i indexes voxels in the searchlight, $\hat{\boldsymbol{\beta}}_i$ is the vector of coefficient estimates at the first level in voxel i, v is the number of voxels in the searchlight, and $\mathbf{D}$ is the diagonal matrix from $\mathbf{S}$.

`MapFile` the name of the similarity maps file, i.e. the file containing the similarity maps in each input folder. If this name is a full path file name, the same maps will be used in the whole input set. If it is a simple name (without path), it will direct the function to select this file to read in the similarity input maps in each input directory.

`MapSel` (opt) a cell array of names specifying which maps from the map file should be used in the correlation. Defaults to all maps (if empty). May

also contain 'BCov', 'SCov', 'BB', if these are not specified in `MapSelPcorr`. The map names specified here must not have been specified in `MapSelPcorr`.

`MapSelPcorr` (opt) a cell array of names specifying which maps from the map file should be partialled out in the correlation. Specify 'BCov', 'SCov', 'BB' to partial out these terms. Defaults to none (if empty). If the Method is 'regression', the terms specified here are simply added to the model (there is no difference with terms specified in `MapSel`). The map names specified here must not have been specified in `MapSel`. These terms are computed as follows:

BCov: $\mathbf{X}^{-}\mathbf{X}^{-\prime}$

SCov: $\mathbf{S} = m^{-1}\sum_i (\hat{\boldsymbol{\beta}}_i - \overline{\boldsymbol{\beta}})(\hat{\boldsymbol{\beta}}_i - \overline{\boldsymbol{\beta}})'$, $\overline{\boldsymbol{\beta}} = m^{-1}\sum_i \hat{\boldsymbol{\beta}}_i$

BB: $m^{-1}\sum_i \hat{\boldsymbol{\beta}}_i \hat{\boldsymbol{\beta}}_i'$

where $\mathbf{X}^{-}$ is the pseudoinverse of the filtered, pre-whitened design matrix, *i* indexes voxels in the volume, *m* is the number of voxels in the volume, and $\hat{\boldsymbol{\beta}}_i$ is the vector of coefficient estimates in voxel *i*. These matrices are normalized to unitary diagonal values when `BrainMapType` is 'cor'.

`OffDiagOffset` (opt) off-diagonal offset. Defaults to zero. Zero selects all off-diagonal terms of the mx (but not the diagonal). Numbers larger than zero exclude the band of elements near the diagonal. Must be a positive value or zero.

`BetaFiles` (opt) If empty (default), directs the function to look for the beta images in the directories specified in Directories as input. If a simple string, it specifies a prefix to be prepended to the names of the beta images in these directories (for example, in case the images were preliminary smoothed). If a cell array of one element containing a string, a regular expression to select the nifti volumes as input to the rsa.

`BetaIdx` (opt) indices to identify the beta images from which the similarity of the brain signal is computed. For example, to select 24 beta images from the third to number 26, enter `3:26` here. When empty, the first beta images up to the order of the selected similarity maps of the stimuli will be selected. This field is useful if the design matrix contains predictors that do not correspond to the trials where the similarity analysis applies (such as confound covariates). Including indices here allows selecting a specific subset of beta images. If the model only contains regressors for the stimuli or the classes of the stimuli, the empty default will be fine because SPM puts confounding covariates such as movement covariates and the constant term at the end of the design matrix. This field must be empty if the representational similarity structure contains a non-empty idx field, which conveys the same information.

`MaskFile`	(opt) mask file. If empty, the file mask.nii is used in each of the input directories (mask.nii is an SPM-generated file). Otherwise, the name of the volume mask to be used in the analysis.
`MaskConfound`	(opt) the mask used to compute confound terms such as BB or SCov. If empty, MaskFile is used.
`SearchlightDef`	(opt) the shape of the searchlight. Possible values are 'sphere' and 'box'. If empty, defaults to 'sphere'.
`SearchlightSize`	(opt) searchlight size (in mm.). If empty defaults to 8. Must be a positive value at least vox size*2.
`ModelName`	(opt) a model name of the analysis to be included in the output. If empty, a model name will be automatically generated based on the parameters of the analysis.
`OutputDir`	(opt) the directory where the correlation volumes are written. If empty, the output is written in the same directories as the input files.

Similarity map structure

This structure is created by `rsa_create_rsm.m` and managed by the functions in `rsa_rsmio.m`. An array of these stuctures is returned in the field `rsms` in the output of `rsa_rsm_searchlight`.

`name`	name of the map
`type`	either 'similarity' or 'converted', for map loaded from the Nili et al. software
`RSM`	the matrix containing pairwise similarity indices. Only the upper triangular portion of this matrix is used
`idx`	a field to select the beta images to which the RSM refers (see the `BetaIdx` field of the argument structure for an explanation)

The following fields are saved to the similarity map file but are not used in the rest of this package:

`data`	the data used to compute the RSM (for reference)
`distf`	the distance function used to compute the RSM (for reference)

Computational settings

These settings are set in the function `rsa_defaults.m`. This file may be changed directly in code to alter these settings.

`minsearchlightsize`	the minimum number of voxels for the searchlight to be included in the volume. The preset value is 27. Searchlights containing less than this number of voxels will be excluded from the estimated volume. Searchlights with small voxel numbers occur at the edge of the volume mask. Ignored if the size of the searchlight specified in the `SearchlightSize` of the argument structure is less than 2 voxels, as in this case the searchlight is explicitly set to small.
`offdiagtol`	the off-diagonal minimum value for a representational map RSM to be considered as carrying information. The preset value is

	0.001. If the upper triangular portion of the similarity map only contains values that are less than this value, the RSM is rejected and an error is thrown.
`predscaletol`	the maximal size of the eigenvalue in the upper triangular part of similarity maps. The preset value is 10000. This is a check on the scale of the data when SCov or BB are used. When coefficient estimates are unstable, violation of this tolerance value may be triggered. Increase this value in code to continue if appropriate.
`ranktol`	tolerance value to declare the similarity maps to be collinear. The preset value is 0.00001. Violation of this tolerance value may be triggered if some of the maps are almost identical.

Acknowledgments

We gratefully acknowledge the work of the members and collaborators of the Wellcome Centre for Human Neuroimaging who created the SPM package and made it available under the GNU General Public Licence, and of Guillaume Flandin in particular for writing the searchlight sampling function on which this package relies.

This work was conducted within the framework of the "Austrian NeuroCloud", supported by the Austrian Federal Ministry of Education, Science and Research.

References

Kriegeskorte, N., Goebel, R., Bandettini, P., 2006. Information-based functional brain mapping. Proc. Natl Acad. Sci. USA 103, 3863.3868.

Kriegeskorte, N., Mur, M., Bandettini, P., 2008. Representational similarity analysis. Connecting the branches of systems neuroscience. Frontiers Sys. Neurosci. doi: 10.3389/neuro.06.004.2008.

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., Kriegeskorte, N., 2014. A toolbox for representational similarity analysis. PLoS Comp. Biol. 10, e1003553. Software available at https://www.mrc-cbu.cam.ac.uk/methods-and-resources/toolboxes/ (last visited 16 Sept. 2020).

Viviani, R. 2021. Overcoming bias in representational similarity analysis. arXiv preprint arXiv:2102.08931.

Licence

```
% The software in this distribution (rsa_algorithms.m, rsa_create_rsm.m, rsa_defaults.m,
% rsa_rsmio.m, rsa_rsm_searchlight.m, rsa_utils.m)
% is copyright, distributed under the GNU General Public Licence.
%_____________________________________________________________________________
%
% This code is free but copyright software, distributed under
% the terms of the GNU General Public Licence as published by the Free
% Software Foundation (either version 2, as given below, or at your
% option, any later version).
%
% For more details on "copyleft", see http://www.gnu.org/copyleft/
%
% This software is supplied as is:
% No formal support or maintenance is provided or implied.
%_____________________________________________________________________________
%
%                    GNU GENERAL PUBLIC LICENSE
%                       Version 2, June 1991
%
%  Copyright (C) 1989, 1991 Free Software Foundation, Inc.
%      59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
%  Everyone is permitted to copy and distribute verbatim copies
%  of this license document, but changing it is not allowed.
%
%                            Preamble
%
%   The licenses for most software are designed to take away your
% freedom to share and change it.  By contrast, the GNU General Public
% License is intended to guarantee your freedom to share and change free
% software--to make sure the software is free for all its users.  This
% General Public License applies to most of the Free Software
% Foundation's software and to any other program whose authors commit to
% using it.  (Some other Free Software Foundation software is covered by
% the GNU Library General Public License instead.)  You can apply it to
% your programs, too.
%
%   When we speak of free software, we are referring to freedom, not
% price.  Our General Public Licenses are designed to make sure that you
% have the freedom to distribute copies of free software (and charge for
% this service if you wish), that you receive source code or can get it
% if you want it, that you can change the software or use pieces of it
% in new free programs; and that you know you can do these things.
%
%   To protect your rights, we need to make restrictions that forbid
% anyone to deny you these rights or to ask you to surrender the rights.
% These restrictions translate to certain responsibilities for you if you
% distribute copies of the software, or if you modify it.
%
%   For example, if you distribute copies of such a program, whether
% gratis or for a fee, you must give the recipients all the rights that
% you have.  You must make sure that they, too, receive or can get the
% source code.  And you must show them these terms so they know their
% rights.
%
%   We protect your rights with two steps: (1) copyright the software, and
% (2) offer you this license which gives you legal permission to copy,
% distribute and/or modify the software.
%
%   Also, for each author's protection and ours, we want to make certain
% that everyone understands that there is no warranty for this free
% software.  If the software is modified by someone else and passed on, we
% want its recipients to know that what they have is not the original, so
% that any problems introduced by others will not reflect on the original
% authors' reputations.
%
%   Finally, any free program is threatened constantly by software
% patents.  We wish to avoid the danger that redistributors of a free
% program will individually obtain patent licenses, in effect making the
% program proprietary.  To prevent this, we have made it clear that any
% patent must be licensed for everyone's free use or not licensed at all.
%
%   The precise terms and conditions for copying, distribution and
% modification follow.
%
%                            ----------------
%
```

```
%                    GNU GENERAL PUBLIC LICENSE
%     TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
%
%   0. This License applies to any program or other work which contains
% a notice placed by the copyright holder saying it may be distributed
% under the terms of this General Public License.  The "Program", below,
% refers to any such program or work, and a "work based on the Program"
% means either the Program or any derivative work under copyright law:
% that is to say, a work containing the Program or a portion of it,
% either verbatim or with modifications and/or translated into another
% language.  (Hereinafter, translation is included without limitation in
% the term "modification".)  Each licensee is addressed as "you".
%
% Activities other than copying, distribution and modification are not
% covered by this License; they are outside its scope.  The act of
% running the Program is not restricted, and the output from the Program
% is covered only if its contents constitute a work based on the
% Program (independent of having been made by running the Program).
% Whether that is true depends on what the Program does.
%
%   1. You may copy and distribute verbatim copies of the Program's
% source code as you receive it, in any medium, provided that you
% conspicuously and appropriately publish on each copy an appropriate
% copyright notice and disclaimer of warranty; keep intact all the
% notices that refer to this License and to the absence of any warranty;
% and give any other recipients of the Program a copy of this License
% along with the Program.
%
% You may charge a fee for the physical act of transferring a copy, and
% you may at your option offer warranty protection in exchange for a fee.
%
%   2. You may modify your copy or copies of the Program or any portion
% of it, thus forming a work based on the Program, and copy and
% distribute such modifications or work under the terms of Section 1
% above, provided that you also meet all of these conditions:
%
%     a) You must cause the modified files to carry prominent notices
%     stating that you changed the files and the date of any change.
%
%     b) You must cause any work that you distribute or publish, that in
%     whole or in part contains or is derived from the Program or any
%     part thereof, to be licensed as a whole at no charge to all third
%     parties under the terms of this License.
%
%     c) If the modified program normally reads commands interactively
%     when run, you must cause it, when started running for such
%     interactive use in the most ordinary way, to print or display an
%     announcement including an appropriate copyright notice and a
%     notice that there is no warranty (or else, saying that you provide
%     a warranty) and that users may redistribute the program under
%     these conditions, and telling the user how to view a copy of this
%     License.  (Exception: if the Program itself is interactive but
%     does not normally print such an announcement, your work based on
%     the Program is not required to print an announcement.)
%
% These requirements apply to the modified work as a whole.  If
% identifiable sections of that work are not derived from the Program,
% and can be reasonably considered independent and separate works in
% themselves, then this License, and its terms, do not apply to those
% sections when you distribute them as separate works.  But when you
% distribute the same sections as part of a whole which is a work based
% on the Program, the distribution of the whole must be on the terms of
% this License, whose permissions for other licensees extend to the
% entire whole, and thus to each and every part regardless of who wrote it.
%
% Thus, it is not the intent of this section to claim rights or contest
% your rights to work written entirely by you; rather, the intent is to
% exercise the right to control the distribution of derivative or
% collective works based on the Program.
%
% In addition, mere aggregation of another work not based on the Program
% with the Program (or with a work based on the Program) on a volume of
% a storage or distribution medium does not bring the other work under
% the scope of this License.
%
%   3. You may copy and distribute the Program (or a work based on it,
% under Section 2) in object code or executable form under the terms of
% Sections 1 and 2 above provided that you also do one of the following:
```

```
%
%     a) Accompany it with the complete corresponding machine-readable
%     source code, which must be distributed under the terms of Sections
%     1 and 2 above on a medium customarily used for software interchange; or,
%
%     b) Accompany it with a written offer, valid for at least three
%     years, to give any third party, for a charge no more than your
%     cost of physically performing source distribution, a complete
%     machine-readable copy of the corresponding source code, to be
%     distributed under the terms of Sections 1 and 2 above on a medium
%     customarily used for software interchange; or,
%
%     c) Accompany it with the information you received as to the offer
%     to distribute corresponding source code.  (This alternative is
%     allowed only for noncommercial distribution and only if you
%     received the program in object code or executable form with such
%     an offer, in accord with Subsection b above.)
%
% The source code for a work means the preferred form of the work for
% making modifications to it.  For an executable work, complete source
% code means all the source code for all modules it contains, plus any
% associated interface definition files, plus the scripts used to
% control compilation and installation of the executable.  However, as a
% special exception, the source code distributed need not include
% anything that is normally distributed (in either source or binary
% form) with the major components (compiler, kernel, and so on) of the
% operating system on which the executable runs, unless that component
% itself accompanies the executable.
%
% If distribution of executable or object code is made by offering
% access to copy from a designated place, then offering equivalent
% access to copy the source code from the same place counts as
% distribution of the source code, even though third parties are not
% compelled to copy the source along with the object code.
%
%   4. You may not copy, modify, sublicense, or distribute the Program
% except as expressly provided under this License.  Any attempt
% otherwise to copy, modify, sublicense or distribute the Program is
% void, and will automatically terminate your rights under this License.
% However, parties who have received copies, or rights, from you under
% this License will not have their licenses terminated so long as such
% parties remain in full compliance.
%
%   5. You are not required to accept this License, since you have not
% signed it.  However, nothing else grants you permission to modify or
% distribute the Program or its derivative works.  These actions are
% prohibited by law if you do not accept this License.  Therefore, by
% modifying or distributing the Program (or any work based on the
% Program), you indicate your acceptance of this License to do so, and
% all its terms and conditions for copying, distributing or modifying
% the Program or works based on it.
%
%   6. Each time you redistribute the Program (or any work based on the
% Program), the recipient automatically receives a license from the
% original licensor to copy, distribute or modify the Program subject to
% these terms and conditions.  You may not impose any further
% restrictions on the recipients' exercise of the rights granted herein.
% You are not responsible for enforcing compliance by third parties to
% this License.
%
%   7. If, as a consequence of a court judgment or allegation of patent
% infringement or for any other reason (not limited to patent issues),
% conditions are imposed on you (whether by court order, agreement or
% otherwise) that contradict the conditions of this License, they do not
% excuse you from the conditions of this License.  If you cannot
% distribute so as to satisfy simultaneously your obligations under this
% License and any other pertinent obligations, then as a consequence you
% may not distribute the Program at all.  For example, if a patent
% license would not permit royalty-free redistribution of the Program by
% all those who receive copies directly or indirectly through you, then
% the only way you could satisfy both it and this License would be to
% refrain entirely from distribution of the Program.
%
% If any portion of this section is held invalid or unenforceable under
% any particular circumstance, the balance of the section is intended to
% apply and the section as a whole is intended to apply in other
% circumstances.
%
```

% It is not the purpose of this section to induce you to infringe any
% patents or other property right claims or to contest validity of any
% such claims; this section has the sole purpose of protecting the
% integrity of the free software distribution system, which is
% implemented by public license practices. Many people have made
% generous contributions to the wide range of software distributed
% through that system in reliance on consistent application of that
% system; it is up to the author/donor to decide if he or she is willing
% to distribute software through any other system and a licensee cannot
% impose that choice.
%
% This section is intended to make thoroughly clear what is believed to
% be a consequence of the rest of this License.
%
%   8. If the distribution and/or use of the Program is restricted in
% certain countries either by patents or by copyrighted interfaces, the
% original copyright holder who places the Program under this License
% may add an explicit geographical distribution limitation excluding
% those countries, so that distribution is permitted only in or among
% countries not thus excluded. In such case, this License incorporates
% the limitation as if written in the body of this License.
%
%   9. The Free Software Foundation may publish revised and/or new versions
% of the General Public License from time to time. Such new versions will
% be similar in spirit to the present version, but may differ in detail to
% address new problems or concerns.
%
% Each version is given a distinguishing version number. If the Program
% specifies a version number of this License which applies to it and "any
% later version", you have the option of following the terms and conditions
% either of that version or of any later version published by the Free
% Software Foundation. If the Program does not specify a version number of
% this License, you may choose any version ever published by the Free Software
% Foundation.
%
%   10. If you wish to incorporate parts of the Program into other free
% programs whose distribution conditions are different, write to the author
% to ask for permission. For software which is copyrighted by the Free
% Software Foundation, write to the Free Software Foundation; we sometimes
% make exceptions for this. Our decision will be guided by the two goals
% of preserving the free status of all derivatives of our free software and
% of promoting the sharing and reuse of software generally.
%
%   NO WARRANTY
%
%   11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
% FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN
% OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
% PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
% OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS
% TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE
% PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
% REPAIR OR CORRECTION.
%
%   12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
% WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
% REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
% INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
% OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
% TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
% YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
% PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
% POSSIBILITY OF SUCH DAMAGES.
%
%   END OF TERMS AND CONDITIONS
%
%   ----------------
%
%   How to Apply These Terms to Your New Programs
%
%   If you develop a new program, and you want it to be of the greatest
% possible use to the public, the best way to achieve this is to make it
% free software which everyone can redistribute and change under these terms.
%
%   To do so, attach the following notices to the program. It is safest
% to attach them to the start of each source file to most effectively
% convey the exclusion of warranty; and each file should have at least

```
% the "copyright" line and a pointer to where the full notice is found.
%
%     <one line to give the program's name and a brief idea of what it does.>
%     Copyright (C) 19yy  <name of author>
%
%     This program is free software; you can redistribute it and/or modify
%     it under the terms of the GNU General Public License as published by
%     the Free Software Foundation; either version 2 of the License, or
%     (at your option) any later version.
%
%     This program is distributed in the hope that it will be useful,
%     but WITHOUT ANY WARRANTY; without even the implied warranty of
%     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%     GNU General Public License for more details.
%
%     You should have received a copy of the GNU General Public License
%     along with this program; if not, write to the Free Software
%     Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
%
%
% Also add information on how to contact you by electronic and paper mail.
%
% If the program is interactive, make it output a short notice like this
% when it starts in an interactive mode:
%
%     Gnomovision version 69, Copyright (C) 19yy name of author
%     Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
%     This is free software, and you are welcome to redistribute it
%     under certain conditions; type `show c' for details.
%
% The hypothetical commands `show w' and `show c' should show the appropriate
% parts of the General Public License.  Of course, the commands you use may
% be called something other than `show w' and `show c'; they could even be
% mouse-clicks or menu items--whatever suits your program.
%
% You should also get your employer (if you work as a programmer) or your
% school, if any, to sign a "copyright disclaimer" for the program, if
% necessary.  Here is a sample; alter the names:
%
%   Yoyodyne, Inc., hereby disclaims all copyright interest in the program
%   `Gnomovision' (which makes passes at compilers) written by James Hacker.
%
%   <signature of Ty Coon>, 1 April 1989
%   Ty Coon, President of Vice
%
% This General Public License does not permit incorporating your program into
% proprietary programs.  If your program is a subroutine library, you may
% consider it more useful to permit linking proprietary applications with the
% library.  If this is what you want to do, use the GNU Library General
% Public License instead of this License.
%_____________________________________________________________________________
% Copyright (C) 2020-21, Roberto Viviani, Institute of Psychology,
% University of Innsbruck
% See also acknowledgment of re-used code in spm12_searchlight.m
```