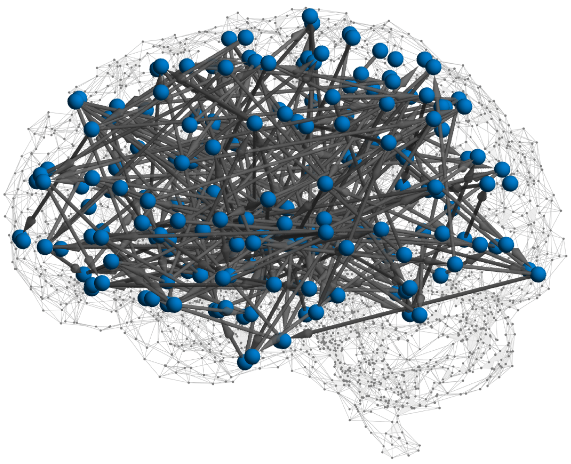

regression DCM (rDCM)

Toolbox Manual

Stefan Frässle

August 28, 2020

Contents

1 Introduction

The regression dynamic causal modeling (rDCM) toobox implements a novel variant of DCM for fMRI (Friston et al., 2003) that enables extremely efficient inference on effective (i.e., directed) connectivity among brain regions. Due to its computational efficiency, inversion of large network models becomes feasible. This manual provides an overview of the most important aspects of the rDCM toolbox and aims to help you quick start your own analyses.

In brief, the approach rests on reformulating the model inversion of a linear (static) DCM (comprising endogenous connections and driving inputs) in the time domain into a Bayesian linear regression like model in the frequency domain. Under this reformulation and some additional modifications to the original DCM framework (for more details, see Frässle et al., 2017), one can derive analytical Variational Bayesian (VB) update equations for the model parameters and hyperparameters. Additionally, the VB approach provides an estimate of the negative free energy, which serves as a lower-bound approximation to the log model evidence. The simple nature of these update equations means that model inversion is extremely fast.

rDCM was first introduced in Frässle et al. (2017), and then further extended by incorporating sparsity constraints in Frässle et al. (2018). Please make sure to cite these two papers whenever you use the rDCM toolbox.

The toolbox is part of the Translational Algorithms for Psychiatry-Advancing Science (TAPAS) toolbox, which is released under the terms of the GNU General Public License (GPL), version 3.0 or later. You can redistribute and/or modify the code under the terms of the GPL. For further information see COPYING or `http://www.gnu.org/licenses/`

2 Important Note

The rDCM toolbox is in an early stage of development and the method is still subject to limitations. Due to these limitations, the requirements of rDCM in terms of fMRI data quality (i.e., fast repetition time (TR), high signal-to-noise ratio (SNR)) are high - as shown in simulation studies (Frässle et al., 2017; 2018). For data that does not meet these conditions, the method might not give reliable results. It remains the responsibility of the user to ensure that his/her dataset fulfills the requirements. Please consult the main toolbox references (see below) for more detailed information.

3 Installation

After downloading the rDCM toolbox, move the main folder with all its contents to a location of your choice. Then open Matlab and add the toolbox to your Matlab path.

4 Documentation

Documentation is contained in this manual and throughout the code. Please also resort to the Wiki page of the toolbox, which contains an FAQ section with frequently asked questions, for further information. Note that this Wiki page is in a developmental stage and will be constantly updated and extended in the future. Finally, in case your question is not covered in the material mentioned above, you can submit a request via the issues system on our GitHub release page for TAPAS (`https://github.com/translationalneuromodeling/tapas/issues`).

5 Main functions

This section provides a brief overview of important functions in rDCM. The main function is **tapas_rdcm_estimate(...)**, which is essentially a wrapper that calls all the relevant functions related to data preparation, model preparation and model inversion.

5.1 tapas_rdcm_model_specification

The function **tapas_rdcm_model_specification(...)** serves as a helper function for setting up the DCM structure required by the main function **tapas_rdcm_estimate(...)**. This function is meant to facilitate the specification of (whole-brain) DCMs for users less familiar with Matlab. Experienced users may set up DCM structures using their own Matlab code.

5.1.1 Usage

The function is called using

[DCM] = tapas_rdcm_model_specification(Y, U, args)

5.1.2 Input arguments

- **Y:** data structure. Mandatory fields are the following: Y.y (data) and Y.dt (repetition time [s]). Data Y.y need to be a matrix with dimensionality [NxR], with N denoting the number of data points (per region) and R representing the number of regions. Optionally, the field Y.name can be used to provide region names.

- **U:** input structure. Mandatory fields are the following: U.u (inputs). Inputs U.u need to be a matrix with dimensionality [16*NxU], with N denoting the number of data points (per region), U representing the number of experimental inputs, and 16 representing the microtime resolution (Note that if inputs are provided as a matrix with dimensionality [NxU], **tapas_rdcm_model_specification(...)** will automatically upsample the inputs). Optionally, the field U.name can be used to provide input names.

- **args:** arguments for model specification. May contain custom network architectures (i.e., endogenous connectivity and driving inputs) that overwrite the default settings.

5.1.3 Outputs

- **DCM:** model structure (i.e., DCM structure, following the nomenclature from SPM). In the DCM structure, all information regarding the model specification is stored; for instance, the connectivity and driving input structure, experimental inputs, number of regions and scans, etc. The DCM structure serves as input to the main function of the toolbox **tapas_rdcm_estimate(...)**.

5.2 tapas_rdcm_set_options

The function **tapas_rdcm_set_options(...)** specifies the settings for an rDCM analysis and generates the fixed hemodynamic response function (HRF) which is used as a convolution kernel from neuronal signal to blood oxygen level dependent (BOLD) signal.

5.2.1 Usage

The function is called using

```
[ options ] = tapas_rdcm_set_options(DCM, input_options, type)
```

5.2.2 Input arguments

- **DCM:** model structure (i.e., DCM structure, following the nomenclature from SPM). In the DCM structure, all information regarding the model specification is stored; for instance, the connectivity and driving input structure, experimental inputs, number of regions and scans, etc.
- **input_options:** estimation options (if empty, default settings are used)
- **type:** string which contains either 'r' for empirical data or 's' for synthetic data

5.2.3 Outputs

This is an overview of only the most important fields of **options**:

- **options.type:** 'r' for empirical data or 's' for synthetic data
- **options.SNR:** signal-to-noise ratio of synthetic fMRI data (for simulations)
- **options.y_dt:** repetition time (TR)
- **options.padding:** zero-padding of Fourier-transformed data (default: no padding)
- **options.visualize:** visualize model inversion results of rDCM
- **options.compute_signal:** generate predicted BOLD signal time series based on the posterior estimates

- **options.evalCp:** generate and store the full covariance matrix (default: none). Only recommended for small DCMs because this matrix becomes very large for whole-brain models with many parameters. Note that region-wise covariance matrices (which contain all information) are stored automatically.
- **options.p0_all:** grid for optimizing the sparsity hyperparameter p_0 (relevant only for rDCM with sparsity constraints). Needs to be specified in **input_options**, otherwise set to default ($p_0 = [0.05 : 0.05 : 0.95]$).
- **options.iter:** number of permutations (of regressors) per region (relevant only for rDCM with sparsity constraints). Needs to be specified in **input_options**, otherwise set to default (iter = 100).
- **options.restrictInputs:** (1) regions that receive driving inputs are known / fixed as specified in DCM.c or (0) prune connectivity and driving input parameters simultaneously from full A- and C-matrices (relevant only for rDCM with sparsity constraints). Needs to be specified in **input_options**, otherwise set to default (restrictInputs = 1).
- **options.DCM:** copy of the original DCM structure
- **options.h:** fixed hemodynamic response function (HRF)

5.3 tapas_rdcm_create_regressors

The function **tapas_rdcm_create_regressors(...)** prepares the "design matrix" (i.e., the set of predictors) and the fMRI data. To this end, BOLD signal time series and driving inputs are transformed into the frequency domain using a (fast) Fourier transformation (FFT).

5.3.1 Usage

The function is called using

[X, Y, DCM, args] = tapas_rdcm_create_regressors(DCM, options)

5.3.2 Input arguments

- **DCM:** model structure (i.e., DCM structure, *see above*)
- **options:** estimation options

5.3.3 Outputs

- **Y:** dependent variable (i.e., Fourier-transformed temporal derivative of BOLD signal)
- **X:** design matrix / set of regressors (i.e., Fourier-transformed BOLD signal and driving inputs)
- **DCM:** model structure (i.e., DCM structure, *see above*)
- **args:** arguments for inference

5.4 tapas_rdcm_ridge

The function **tapas_rdcm_ridge(...)** implements the VB update equations for the original formulation of the rDCM framework as introduced in Frässle et al. (2017). Specifically, the function implements Eq. (21) and Eqs. (23-28) from the paper for the model parameters & hyperparameters and the negative free energy, respectively.

5.4.1 Usage

The function is called using

[output] = tapas_rdcm_ridge(DCM, X, Y, args)

5.4.2 Input arguments

The input arguments of function **tapas_rdcm_ridge(...)** are the outputs obtained from function **tapas_rdcm_create_regressors(...)**:

- **DCM:** model structure (i.e., DCM structure, *see above*)
- **Y:** dependent variable (i.e., Fourier-transformed temporal derivative of BOLD signal)
- **X:** design matrix / set of regressors (i.e., Fourier-transformed BOLD signal and driving inputs)
- **args:** arguments for inference

5.4.3 Outputs

This is an overview of only the most important fields of **output**:

- **output.Ep:** posterior means of connectivity (A-matrix) and driving input (C-matrix) parameters
- **output.sN:** region-wise posterior covariance matrices
- **output.t:** posterior estimate of noise precision
- **output.aN** & **bN:** posterior shape and rate parameter of Gamma distribution over measurement noise
- **output.Ip:** posterior probability of Bernoulli distribution over binary indicator variables (for original rDCM, without sparsity constraints, equal to 1 for all present connections)
- **output.logF:** negative free energy
- **output.priors:** prior settings utilized for model inversion
- **output.inversion:** rDCM function / variant used for model inversion

5.5 tapas_rdcm_sparse

The function **tapas_rdcm_sparse(...)** implements the VB update equations for the rDCM framework comprising sparsity constraints as introduced in Frässle et al. (2018). Specifically, the function implements Eq. (15) and Eqs. (17-23) from the paper for the model parameters & hyperparameters and the negative free energy, respectively.

5.5.1 Usage

The function is called using

[output] = tapas_rdcm_sparse(DCM, X, Y, args)

5.5.2 Input arguments

The input arguments of function **tapas_rdcm_sparse(...)** are the outputs obtained from function **tapas_rdcm_create_regressors(...)**:

- **DCM:** model structure (i.e., DCM structure, *see above*)
- **Y:** dependent variable (i.e., Fourier-transformed temporal derivative of BOLD signal)
- **X:** design matrix / set of regressors (i.e., Fourier-transformed BOLD signal and driving inputs)
- **args:** arguments for inference

5.5.3 Outputs

This is an overview of only the most important fields of **output**:

- **output.Ep:** posterior means of connectivity (A-matrix) and driving input (C-matrix) parameters
- **output.sN:** region-wise posterior covariance matrices
- **output.t:** posterior estimate of noise precision
- **output.aN** & **bN:** posterior shape and rate parameter of Gamma distribution over measurement noise
- **output.Ip:** posterior probability of Bernoulli distribution over binary indicator variables
- **output.logF:** negative free energy
- **output.priors:** prior settings utilized for model inversion
- **output.inversion:** rDCM function / variant utilized for model inversion

5.6 tapas_rdcm_estimate

The function **tapas_rdcm_estimate(...)** is the main function of the rDCM toolbox. It is essentially a wrapper that calls all relevant functions necessary to perform an rDCM analysis.

5.6.1 Usage

The function is called using

[output] = tapas_rdcm_estimate(DCM, type, options, methods)

5.6.2 Input arguments

- **DCM:** model structure (i.e., DCM structure, *see above*)
- **type:** string which contains either 'r' for empirical data or 's' for synthetic data
- **options:** estimation options (if empty, default settings are used)
- **methods:** rDCM variant: 1 = original, 2 = with sparsity constraints

5.6.3 Outputs

Most fields of the output structure of **tapas_rdcm_estimate(...)** are the same as those of the main functions for model inversion - that is, **tapas_rdcm_ridge(...)** for the original rDCM and **tapas_rdcm_sparse(...)** for rDCM with sparsity constraints (*for details, see above*). Additional output fields are related to the predicted BOLD signal time series and some basic statistics.

- **output.allParam:** summary of all connectivity parameters
- **output.statistics:** statistics over model parameters - for instance, mean-squared error and number of sign errors between estimated and true (simulations) or VBL (empirical data) parameter values
- **output.signal:** measured / true and predicted BOLD signal time courses
- **output.residuals:** residuals of predicted BOLD signal time courses and mean-squared error (MSE) between measured / true and predicted signal
- **output.inputs:** copy of experimental inputs specified in the DCM structure
- **output.time:** run-time of rDCM analysis
- **output.rngSeed:** seed of the random number generator (important for reproducing model inversion results)
- **output.ver:** version number

6 Application to task or resting-state fMRI

While rDCM was originally designed to work with experimentally controlled perturbations (i.e., task data), it is also possible to fit the model to resting-state fMRI data. This can be achieved by "switching off" driving inputs (i.e., setting all input parameters to zero) and, in order to explain activity in any given region, relying on measured data (in the Fourier domain) in regions from which afferent connections are received. This means that, in contrast to stochastic variants of DCM, the generative model does not explicitly represent endogenous fluctuations in neuronal activity and has no concept of stochastic "innovations" or similar ways noise can drive neuronal activity intrinsically. Instead, endogenous fluctuations of BOLD signal in any given region are predicted as a linear mixture of intrinsic fluctuations from other regions. A demonstration of the face validity and construct validity of rDCM for resting-state fMRI data can be found in Frässle et al. (2020).

In order to specify a resting-state model for rDCM analysis, a DCM structure has to be prepared that does not contain a field related to driving inputs (i.e., DCM.U). The most convenient way to specify such a resting-state model is by calling the model specification function **tapas_rdcm_model_specification(...)** without any driving inputs:

```
[ DCM ] = tapas_rdcm_model_specification(Y, [ ], [ ])
```

Otherwise, it is also possible to specify the DCM structure using your own Matlab code. Once a DCM structure for resting-state fMRI data (i.e., without input field DCM.U) has been specified, the structure can be passed on to the main function **tapas_rdcm_estimate(...)**. This function will then automatically make all necessary adjustments to prepare the DCM for resting-state fMRI data before running the model inversion.

7 Tutorial

The rDCM toolbox contains a simple tutorial **tapas_rdcm_tutorial()** that illustrates how rDCM can be used. In brief, the script generates synthetic fMRI data from a DCM with known network architecture and parameter values. Here, the network architecture is based on the S50 structure introduced in Smith et al. (2011) and illustrated in Fig. 1.

The script then demonstrates an rDCM analysis, aiming to recover the "true" (data-generating) network architecture by pruning a fully-connected network to the most essential connections (given the synthetic data). Performance of rDCM is then assessed by computing sensitivity and specificity of recovering the true network architecture. We here detail the individual steps.

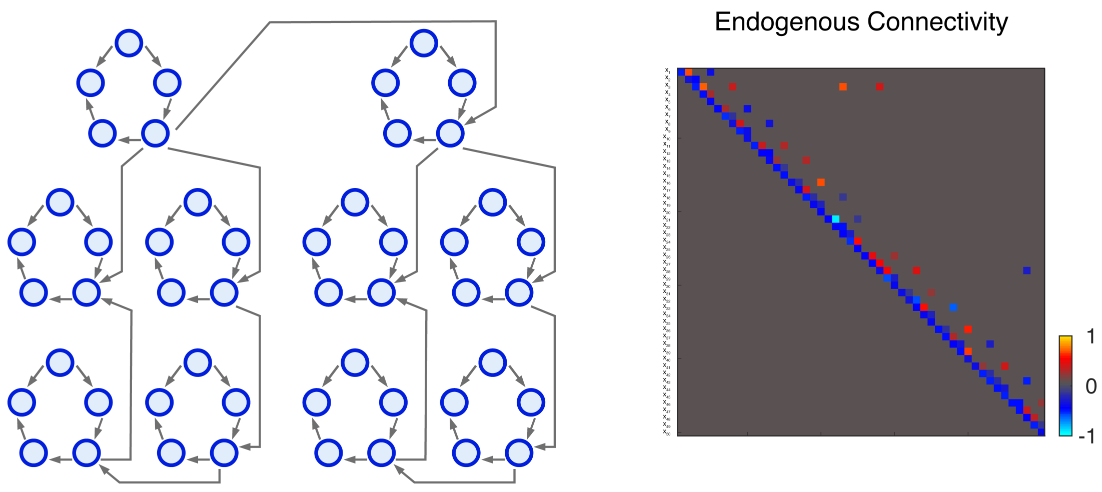

Figure 1: Connectivity architecture of large-scale network used in tutorial. The model comprises 50 nodes and was initially introduced as the S50 network structure in Smith et al. (2011). This model consists of 10 local sub-networks (each comprising 5 nodes connected in a ring via unidirectional links; although not with cyclic directionality of influences) that were connected via one long-range connection to mimic the small-world architecture of the human brain. *Figure taken from Frässle et al. (2018).*

First, we generate synthetic fMRI data for the exemplary network architecture.

```
% generate synthetic data (for simulations)
if ( strcmp(type,'s') )
        fprintf('Generate synthetic data\n')
        DCM = tapas_rdcm_generate(DCM, options, options.SNR);
end
```

and set the options for our analysis

```
% create the options file
options = tapas_rdcm_set_options(DCM, options, type);
```

The DCM can then be reformulated as a GLM-like model in the frequency domain using the function **tapas_rdcm_create_regressors(DCM, options)**.

Upon specification of the model, inversion in rDCM (here: with sparsity constraints) is performed for each value of the hyperparameter p_0 individually (here, for computational reasons, we use only one value). The optimal p_0 value can then be chosen to maximize the negative free energy (i.e., Maximum likelihood II / Empirical Bayes; Bishop, 2006).

```
% iterate over p0 values
for p0_counter = 1:length(options.p0_all)

    % specify p0
    args.p0_temp = options.p0_all(p0_counter);

    % specify the number of permutations (per brain region)
    if ( isfield(options,'iter') ), args.iter = options.iter; end

                    ⋮

    % output progress
    if ( ~isfield(DCM,'M') || ~isfield(DCM.M,'noprint') || ~DCM.M.noprint )
      msg = sprintf('Processing p0: %d/%d', p0_counter, length(options.p0_all));
      fprintf([reverseStr, msg]);
      reverseStr = repmat(sprintf('\b'), 1, length(msg));
    end

    % rDCM (with sparsity constraints)
    output_temp = tapas_rdcm_sparse(DCM, X, Y, args);
    output_all{p0_counter} = output_temp{1};

    % get the negative free energy
    F_all(p0_counter) = output_all{p0_counter}.logF;

end

% find optimal hyperparameter settings (p0)
[~, F_max_ind] = max(F_all);

% asign results
output = output_all{F_max_ind};
```

Once model inversion is completed, we can plot the results using the command

```
% visualize the results
tapas_rdcm_visualize(output, DCM, options, plot_regions, 2)
```

This yields the plots in Fig. 2, showing the estimated and "true" (data-generating) connectivity pattern (A-matrix), as well as the predicted and measured / synthetic BOLD signal time series. For empirical data, where the ground-truth connectivity and parameter values are unknown, the function **tapas_rdcm_visualize(output, DCM, options, plot_regions, plot_mode)** plots the posterior probability of the Bernoulli distribution over binary indicator variables.

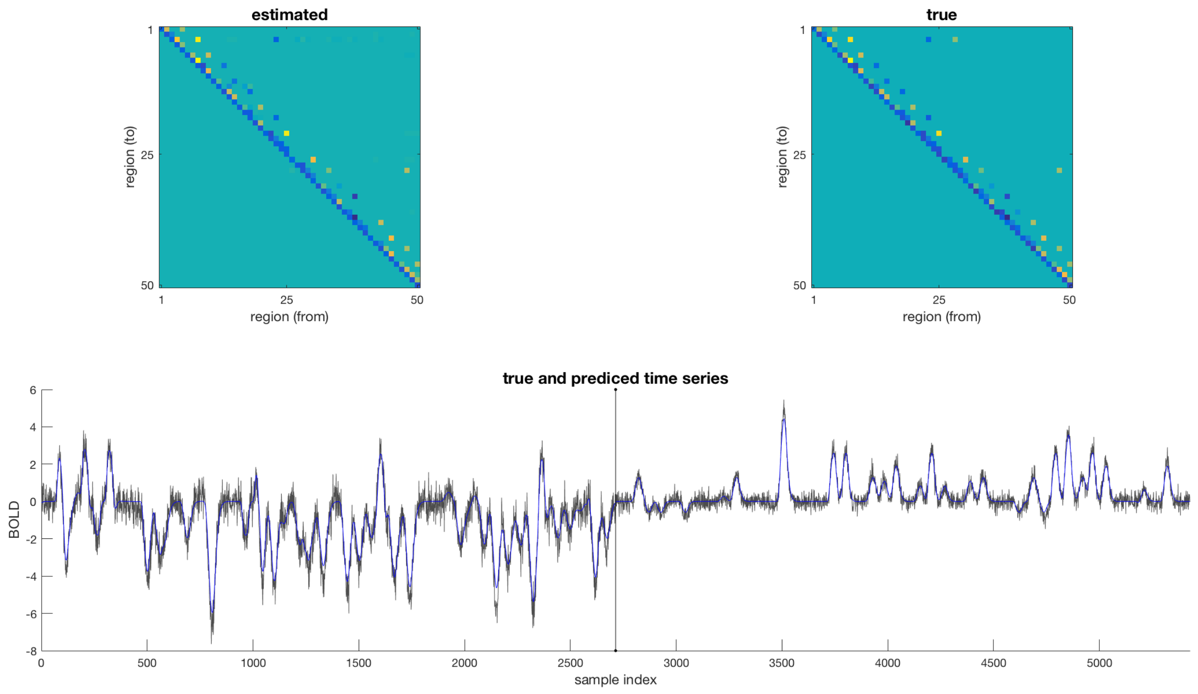

Figure 2: Model inversion results of rDCM using sparsity constraints. (*Top, left:*) Estimated and (*Top, right:*) true (data-generating) network architecture and parameter values. The estimated connectivity profile has been obtained by starting from a fully connected A-matrix and then pruning the network to the most essential connections. (*Bottom:*) Predicted data (blue) and measured/synthetic data (gray) for two regions of interest (as defined in "plot_regions").

Finally, the performance of rDCM for this particular simulation is assessed in terms of the sensitivity and specificity of recovering the "true" (data-generating) network architecture and the root-mean-squared-error (RMSE) between the true and estimated parameter values. The output reads as follows:

```
========================================
Regression dynamic causal modeling (rDCM)
========================================

Run model inversion
Processing p0: 1/1
Processing region: 50/50
Finalize results
Elapsed time is 521.670545 seconds.

Summary
- - - - - - - - - - - - - - - -

Accuracy of model architecture recovery:
Sensitivity: 0.918 - Specificity: 0.992
Root mean squared error (RMSE): 0.106
```

References

Main toolbox references

Frässle, S.*, Lomakina, E.I.*, Razi, A., Friston, K.J., Buhmann, J.M., Stephan, K.E., 2017. Regression DCM for fMRI. *NeuroImage* 155, 406-421.

Frässle, S.*, Lomakina, E.I.*, Kasper, L., Manjaly Z.M., Leff, A., Pruessmann, K.P., Buhmann, J.M., Stephan, K.E., 2018. A generative model of whole-brain effective connectivity. *NeuroImage* 179, 505-529.

Frässle, S., Harrison, S.J., Heinzle, J., Clementz, B.A., Tamminga, C.A., Sweeney, J.A., Gershon, E.S., Keshavan, M.S., Pearlson, G.D., Powers, A., Stephan, K.E., 2020. Regression dynamic causal modeling for resting-state fMRI. *bioRxiv*, https://doi.org/10.1101/2020.08.12.247536.

Further references

Bishop, C.M., 2006. Pattern Recognition and Machine Learning, vol. 12. *Springer*, New York., p. 105, 13, 47.

Friston, K., Harrison, L., Penny, W., 2003. Dynamic causal modelling. *NeuroImage* 19, 1273-1302.

Smith, S.M., Miller, K.L., Salimi-Khorshidi, G., Webster, M., Beckmann, C.F., Nichols, T.E., Ramsey, J.D., Woolrich, M.W., 2011. Network modelling methods for FMRI. *NeuroImage* 54, 875-891.