TRENTOOL 3.4.0 beta – User Documentation

Patricia Wollstadt* Michael Lindner Raul Vicente Michael Wibral
Nicu Pampu Mario Martinez-Zarzuela

Version 0.93
`http://www.trentool.de/`

Contents

*`p.wollstadt@stud.uni-frankfurt.de`

List of Figures

List of Tables

List of Listings

1 Introduction

What is TRENTOOL? TRENTOOL (TRansfer ENtropy TOOLbox) is an open-source MATLAB toolbox that allows the user to easily handle the considerable complexity of transfer entropy (TE) estimation from time series. For the use with neural data TRENTOOL seamlessly integrates with the popular FieldTrip toolbox (Oostenveld, Fries, Maris, & Schoffelen, 2011). TRENTOOL provides the following features:

- Transfer entropy estimation (section 4.2)
- Reconstruction of interaction delays (section 4.3)
- Time-resolved transfer entropy estimation (section 4.4)
- Group analysis and statistics (section 4.6)
- Partial correction for multivariate interactions (section 4.7)

Implementation TRENTOOL is implemented in Mathworks® MATLAB® (The MathWorks Inc., Natick, MA, 2008) with some functions written in NVIDIA® CUDA™ C/C++ code (NVIDIA Corporation, 2013). TRENTOOL also makes use of the MATLAB toolboxes FieldTrip (Oostenveld et al., 2011) and TSTOOL (`http://www.dpi.physik.uni-goettingen.de/tstool/`).

The user interacts with TRENTOOL through MATLAB scripts (`.m`-files). TRENTOOL does not provide a graphical user interface.

Organization of this document This document provides a comprehensive user documentation of TRENTOOL's functionalities and intended analysis strategies. We will first give some information on the installation of TRENTOOL (section 2, *Installation and configuration*), next we will present some background information on TE estimation to allow the user to better understand the input parameters required for the use of TRENTOOL (section 3, *Background*). We will then discuss the general use of TRENTOOL and also provide instructions on how to use additional functionalities provided (section 4, *Using TRENTOOL*). Last, we give some example scripts that may be used as a first step in using TRENTOOL for data analysis (*Example scripts*). Also, most of the information provided in this documentation can be found in the individual function's help text using MATLAB's `help` function.

For in depth treatments of the TE measure and individual TRENTOOL functionalities refer to the following publications:

- Schreiber (2000): Introduction of TE
- Kraskov, Stögbauer, and Grassberger (2004): TE estimator implemented in TRENTOOL
- Lindner, Vicente, Priesemann, and Wibral (2011): Original publication of the TRENTOOL toolbox, main parts of this documentation were first published here
- Vicente, Wibral, Lindner, and Pipa (2011): TE estimation, application of TE to magnetoencephalographic data
- Wibral et al. (2011): TE estimation, application of TE to magnetoencephalographic data
- Wibral et al. (2013): Improved TE estimator, reconstruction of interaction delays

- Wibral et al. (2012): Graph algorithm for partial correction for multivariate interactions
- Wollstadt, Martinez-Zarzuela, Vicente, Diaz-Pernas, and Wibral (2013): TE estimation from non-stationary data using an ensemble approach presented in Gomez-Herrero et al. (2010)

2 Installation and configuration

Download You can download the current TRENTOOL version from `www.trentool.de`. TSTOOL functions that are required by TRENTOOL functions are included since TRENTOOL v3.3. Furthermore, FieldTrip needs to be downloaded separately from `http://fieldtrip.fcdonders.nl/` download.

Installation Unpack the downloaded archives and add TRENTOOL and FieldTrip to your MATLAB path by using the `addpath` command (see listings 1 and 2). You may use the command `restoredefaultpath` to reset your MATLAB path before adding the FieldTrip and TRENTOOL paths to avoid conflicting versions of toolboxes being added your path (see also `http://fieldtrip.fcdonders.nl/faq/should_i_add_fieldtrip_with_all_subdirectories_to_my_matlab_path`).

If you want to use NVIDIA CUDA GPU functionalities implemented in TRENTOOL (see section 4.4), after you have to compile the respective mex files by calling `install.m` from within MATLAB. NOTE: At this point, GPU functionality is only supported for **Linux OS** and GPU devices that support **CUDA**.

Listing 1: Generic MATLAB paths to FieldTrip and TRENTOOL.

```
restoredefaultpath;
addpath('/path/to/fieldtrip/fieldtrip-version');
ft_defaults;
addpath('/path/to/TRENTOOL/TRENTOOL3');
```

Listing 2: Concrete example for MATLAB paths to FieldTrip and TRENTOOL.

```
restoredefaultpath;
addpath('/data/common/FieldtripCurrent/fieldtrip-20150928');
ft_defaults;
addpath('/data/common/TRENTOOL_current/TRENTOOL3');
```

Dependencies and compatibility TRENTOOL depends on the MATLAB toolboxes FieldTrip and TSTOOL. TRENTOOL expects input data to be in the standard FieldTrip data format and makes use of some FieldTrip and TSTOOL functions. We will try to keep a compatibility with the current version of FieldTrip. However, as FieldTrip is continuously being updated (`http://fieldtrip.fcdonders.nl/tutorial/introduction`), we can not always guarantee the compatibility with the latest FieldTrip version. Please refer to the TRENTOOL homepage to check for the newest FieldTrip version that has been tested for compatibility with TRENTOOL. We recommend to use TRENTOOL with tested FieldTrip versions.

3 Background

3.1 Transfer entropy

Transfer entropy (TE) is a quantitative measure of information transfer between two processes `X` and `Y` (Schreiber, 2000). Schreiber's formulation of TE can be seen as the information theoretic implementation of Wiener's principle of observational causality (predictive information transfer) (Wiener, 1956; Vicente et al., 2011): In Wiener's definition an improvement of the prediction of the future of a time series $\{y_t\}$ from its own past by the incorporation of information from the past of a second time series $\{x_t\}$ is seen as a hint at a causal interaction from $\{x_t\}$ to $\{y_t\}$[1] (Fig. 1). One can formalized Wiener's principle as a conditional mutual information between the future of a process `Y`, y^+ and the past state $\mathbf{x}^-$ of a process `X`, conditioned on the past state $\mathbf{y}^-$ of process `Y` (Schreiber, 2000; Vicente et al., 2011) (here, time series $\{x_t\}$ and $\{y_t\}$ are seen as observed, scalar observations of the underlying processes `X` and `Y`):

$$TE(X \to Y) = I\left(y^+; \mathbf{x}^- | \mathbf{y}^-\right). \tag{1}$$

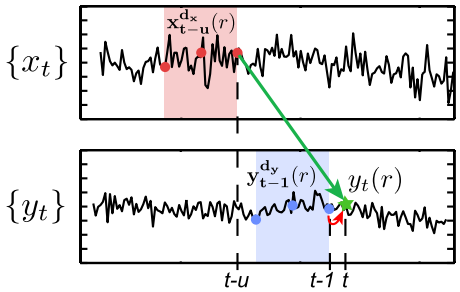

Figure 1: Transfer entropy (green arrow) between a source `X` and a target process `Y`. Red and blue boxes indicate past states of both processes, the green star indicates the future value of `Y`. u indicates the interaction delay from `X` to `Y`.

3.2 Notation and preliminaries

Notation We assume, that TE is measured between two coupled physical systems $\mathcal{X}$ and $\mathcal{Y}$, that produce observable time series $\{x_1, \ldots, x_t, \ldots, x_N\}$ and $\{y_1, \ldots, y_t, \ldots, y_N\}$ with $t \in \{1 \ldots N\}$. We formalize these time series as realizations of random processes `X`, `Y`, which in turn are collections of random variables, X_t and Y_t (sorted by the integer t). Each random variable X_t, at a specific time t, has a set of J possible outcomes $\mathcal{A}_{X_t} = \{a_1, \ldots, a_j, \ldots, a_J\}$, and their associated probability density function (PDF) $p_{X_t}(x_t = a_j)$.

State space reconstruction As TE is defined between states of the processes `X` and `Y`, we need to reconstruct these states from scalar observations $\{x_t\}$ and $\{y_t\}$. A state refers to a set

[1] Note, that TE is not a measure of "causality". The root cause of this difference is that there is no information transfer without causal interaction, but the reverse is not true, i.e. there can be causal interaction with zero information transfer (Ay & Polani, 2008; J. Lizier & Prokopenko, 2010; Chicharro & Andrzejak, 2009)

of variables, that fully characterizes all relevant parameters to sufficiently describe a process at any given moment in time. The set of all possible states of a system is called the state space. TRENTOOL reconstructs states from scalar time series using time-delay embedding (Takens, 1981) under the assumption, that each process can be approximated by a Markov process. Time-delay embedding for an observation at time t takes the form

$$\mathbf{x}_t^{d_x} = \left(x_t, x_{t-\tau}, \ldots, x_{t-(d_x-1)\tau}\right), \tag{2}$$

where d_x is the embedding dimension and τ the embedding delay. TRENTOOL optimizes both parameters according to Ragwitz' criterion (Ragwitz & Kantz, 2002), where d_x and τ are jointly optimized to minimize the prediction error of a local predictor, that predicts the future of each signal from its past.

Stationarity TE or other information theoretic functionals are calculated from the PDFs of the quantities involved. PDFs are typically not known *a priori* and have to be estimated from multiple observed realizations of a random variable. How these realizations are obtained from data depends on whether the process in question is stationary or non-stationary. Stationarity of a process means that PDFs of the random variables that form the random process do not change over time, such that $p_{X_t}(x_t = a_j) = p_{X_s}(x_s = a_j),\ \forall s, t \in \mathbb{N}$. Any PDF $p_{X_t}(\cdot)$ may then be estimated from one observation of process `X` by means of collecting realizations $(x_1, \ldots, x_N)$ *over time*. For processes that are non-stationary, i.e. $p_{X_t}(x_t = a_j) \neq p_{X_s}(x_s = a_j),\ \forall s, t \in \mathbb{N}$, temporal pooling is not applicable as PDFs vary over time t and every random variable X_t is associated with a different PDF $p_{X_t}(\cdot)$.

TRENTOOL implements TE estimators for both stationary and non-stationary time series. The estimator assuming stationarity pools observations over time for PDF estimation. For non-stationary processes it is possible to repeat a process in time and pool observations over repetitions (see Gomez-Herrero et al. (2010) and Wollstadt et al. (2013)). We call the repeated observations of a process an ensemble of time series. For an observation at time t from a repetition $r = 1, \ldots, R$, we write $x_t(r)$. Note, that t now refers to a time point t relative to the onset T of repetition r. If we choose the number of repetitions large enough, i.e. there is a sufficiently large set $\mathcal{R} = |R|$ of time points T, at which the process is repeated, we can assume that PDFs $p_{X_{T+t}}(\cdot)$ at time point t relative to the onset of the repetition at T are equal over all R repetitions. We may obtain a reliable estimation of $p_{X_{T+t}}(\cdot)$ from this ensemble by evaluating $p_\cdot(\cdot)$ over all observations $x_{T+t}, \forall T \in \mathcal{R}$.

3.3 Practical TE estimation in TRENTOOL

Transfer entropy functional TRENTOOL allows for TE estimation from both stationary and non-stationary time series. We here present the TE functional TE_{SPO} for TE estimation from an ensemble of time series (Lindner et al., 2011; Vicente et al., 2011; Wibral et al., 2013; Wollstadt et al., 2013). The TE_{SPO} estimator allows for TE estimation from non-stationary ensemble data as well as for TE estimation from stationary time series as a special case:

$$TE_{SPO}\left(\mathtt{X} \to \mathtt{Y}, t, u\right) = \sum_{\substack{y_t(r), \mathbf{y}_{t-1}^{d_y}(r), \mathbf{x}_{t-u}^{d_x}(r) \\ \in \mathcal{A}_{Y_t, \mathbf{Y}_{t-1}^{d_y}, \mathbf{X}_{t-u}^{d_x}}}} p\left(y_t(r), \mathbf{y}_{t-1}^{d_y}(r), \mathbf{x}_{t-u}^{d_x}(r)\right) \log \frac{p\left(y_t(r) | \mathbf{y}_{t-1}^{d_y}(r), \mathbf{x}_{t-u}^{d_x}(r)\right)}{p\left(y_t(r) | \mathbf{y}_{t-1}^{d_y}(r)\right)}, \tag{3}$$

where $y_t(r)$ denotes the future observation of $\mathtt{Y}$ in repetition $r = 1, \ldots, R$; $\mathbf{y}_{t-1}^{d_y}(r)$ denotes the past state of $\mathtt{Y}$ in repetition r and $\mathbf{x}_{t-u}^{d_x}(r)$ denotes the past state of $\mathtt{X}$ in repetition r. u is the delay of the information transfer between processes $\mathtt{X}$ and $\mathtt{Y}$ (Wibral et al., 2013). In order to estimate the involved PDFs, necessary realizations of the respective random variables may be obtained through *ensemble evaluation* over repetitions r or through evaluation over time (for stationary time series). For a completely stationary time series, the estimator may be evaluated over one repetition and all available time points. For non-stationary time series, the estimator may be iteratively evaluated over all repetitions at each time point. Also, the estimator allows to combine both approaches by pooling observations over repetitions and over time windows for which local stationarity can be assumed.

Transfer entropy estimator For practical TE estimation in TRENTOOL using equation 3, we proceed by first rewriting the estimator as a sum of four individual Shannon entropies:

$$TE_{SPO}\left(\mathtt{X} \to \mathtt{Y}, t, u\right) = H\left(\mathbf{Y}_{t-1}^{d_Y}, \mathbf{X}_{t-u}^{d_X}\right) - H\left(Y_t, \mathbf{Y}_{t-1}^{d_Y}, \mathbf{X}_{t-u}^{d_X}\right) + H\left(Y_t, \mathbf{Y}_{t-1}^{d_Y}\right) - H\left(\mathbf{Y}_{t-1}^{d_Y}\right), \tag{4}$$

The Shannon differential entropies H can be efficiently estimated using nearest neighbor techniques (Kozachenko & Leonenko, 1987; Victor, 2002). TRENTOOL uses a modified Kraskov-Stögbauer-Grassberger estimator for transfer entropy (Kraskov et al., 2004):

$$\begin{aligned} TE_{SPO}\left(\mathtt{X} \to \mathtt{Y}, u, t_0, \Delta t\right) = \\ \psi\left(k\right) + \langle \psi\left(n_{\mathbf{y_{t-1}^{d_y}}(r)} + 1\right) \\ -\psi\left(n_{y_t(r)\ \mathbf{y_{t-1}^{d_y}}(r)} + 1\right) \\ -\psi\left(n_{\mathbf{y_{t-1}^{d_y}}(r)\ \mathbf{x_{t-u}^{d_x}}(r)} + 1\right)\rangle_{r,t}, \\ \forall r, t_0 - \frac{\Delta}{2} \leq t \leq t_0 + \frac{\Delta}{2}, \end{aligned} \tag{5}$$

where ψ denotes the digamma function and the angle brackets ($< \cdot >_r$) indicate an averaging

over points in different repetitions r in time window t ($t \in [t^-; t^+]$ and $t^- \leq t \leq t^+$, where $\Delta t = t^+ - t^-$). The distances to the k-th nearest neighbor in the highest dimensional space (spanned by $X_t, \mathbf{Y}_{t-1}^{d_y}, \mathbf{X}_{t-u}^{d_x}$) define the radius of the spheres for the counting of the number of points ($n.$) in these spheres around each state vector ($\cdot$) involved.

Equation 5 is the most flexible formulation of the estimator implemented in TRENTOOL. It allows to obtain neighbor counts from pooling over time t and/or pooling over repetitions r. We implemented two TE estimation strategies derived from this estimator. The first allows for mere temporal pooling (eq. 6, estimating TE for each repetition r separately, see section 4.2, *TRENTOOL core functions*). The second allows for mere ensemble pooling (eq. 7) or a combination of ensemble and temporal pooling (eq. 5, see section 4.4, *Estimating time-resolved transfer entropy*).

$$\begin{aligned} TE_{SPO}\left(\mathtt{X} \rightarrow \mathtt{Y}, u, r_i\right) =& \psi\left(k\right) + \langle \psi\left(n_{\mathbf{y_{t-1}^{d_y}}(r_i)} + 1\right) \\ & - \psi\left(n_{y_t(r_i)\ \mathbf{y_{t-1}^{d_y}}(r_i)} + 1\right) \\ & - \psi\left(n_{\mathbf{y_{t-1}^{d_y}}(r_i)\ \mathbf{x_{t-u}^{d_x}}(r_i)} + 1\right)\rangle_t,\ \forall t, \end{aligned} \tag{6}$$

$$\begin{aligned} TE_{SPO}\left(\mathtt{X} \rightarrow \mathtt{Y}, u, t\right) =& \psi\left(k\right) + \langle \psi\left(n_{\mathbf{y_{t-1}^{d_y}}(r)} + 1\right) \\ & - \psi\left(n_{y_t(r)\ \mathbf{y_{t-1}^{d_y}}(r)} + 1\right) \\ & - \psi\left(n_{\mathbf{y_{t-1}^{d_y}}(r)\ \mathbf{x_{t-u}^{d_x}}(r)} + 1\right)\rangle_r,\ \forall r, \end{aligned} \tag{7}$$

TRENTOOL analysis steps TRENTOOL uses the estimator in equations 5 for TE estimation. To estimate TE from scalar time series using the estimators, TRENTOOL proceeds as follows:

1. Parameters for state space reconstruction are optimized for a given pair of time series according to Ragwitz' criterion (Ragwitz & Kantz, 2002)
2. Time series are embedded using optimized parameters
3. Embedded time series are pooled over time and/or repetitions to form a state space for nearest neighbor searching
4. State spaces are searched using nearest neighbor techniques (Kraskov et al., 2004)
5. Nearest neighbor counts resulting from nearest neighbor searches are used for TE calculation according to eq. 5
6. Resulting TE values are tested for statistical significance using non-parametric permutation testing to account for the bias introduced by the estimator (Lindner et al., 2011; Vicente et al., 2011)

7. Test for volume conduction using a shift test (Lindner et al., 2011) or additional conditioning (Faes, Nollo, & Porta, 2013)

This is the core functionality for TE estimation in TRENTOOL. The practical use of TE estimation in TRENTOOL is described in more detail in section 4.2, *TRENTOOL core functions*. Additional functionalities provided by the toolbox are described in the next subsection.

3.4 Additional functionalities implemented in TRENTOOL

TRENTOOL provides additional functionalities for TE estimation and statistical testing, especially for TE estimation from neural data:

- Reconstruction of interaction delays (Wibral et al., 2013)
- Estimation of time-resolved TE from an ensemble of time series (Wollstadt et al., 2013; Gomez-Herrero et al., 2010)
- Group analysis and statistical testing (Lindner et al., 2011)
- Graph algorithm for partial correction for multivariate interactions between more than two analyzed time series (Wibral et al., 2012)

Background information on the presented functionalities can be found in the respective articles. The following subsections provide short introductions to each functionality to enable the user to chose the appropriate analysis steps for a given data set.

3.4.1 Reconstruction of interaction delays

We recently showed, that the TE functional implemented in TRENTOOL (eq. 3) allows for the reconstruction of interaction delays between the two analyzed processes (Wibral et al., 2013). We showed that the estimated TE value becomes maximal when the parameter u is equal to the true interaction delay δ. Thus, our estimator may be used to recover δ from a set of assumed candidate values u:

$$\delta = arg \max_{u} \; TE_{SPO}(X \rightarrow Y, u) \tag{8}$$

This functionality is implemented in TRENTOOL by looping over TRENTOOL's core functions using varying values for u. Using TRENTOOL for interaction delay reconstruction is described in more detail in section 4.3, *Reconstruction of interaction delays*.

3.4.2 Estimating TE from an ensemble of time series

Gomez-Herrero and colleagues showed that it is possible to estimate the necessary probability density functions for TE calculation from an ensemble of time series (Gomez-Herrero et al., 2010). We implemented the ensemble method for TE estimation in TRENTOOL, allowing the treatment of non-stationary time series and the estimation of time-resolved TE (Wollstadt et al., 2013). Using this ensemble method circumvents the necessity of temporal pooling by using ensemble pooling over repeated observations. The use of the ensemble method is described in

more detail in section 4.4, *Estimating time-resolved transfer entropy from an ensemble of time series.*

3.4.3 Group analysis

TRENTOOL allows for the statistical comparison of TE values between two sets of data (e.g. two conditions or subject groups in an experiment) (Lindner et al., 2011). TRENTOOL choses common estimation parameters over all data sets to be analyzed, which are then used in individual TE estimation. Using common parameters for TE estimation allows for a statistical comparison of estimated TE values between groups, because statistical differences can be attributed to differences in information transfer rather than differences in the estimation parameters. Group differences are tested using a non-parametric permutation test (Maris & Oostenveld, 2007; Lindner et al., 2011). The use of the group analysis is described in more detail in section 4.6, *Group analysis in TRENTOOL.*

3.4.4 Graph correction for multivariate effects

TRENTOOL estimates TE between *pairs* of processes. However, if a system consists of more than two processes, multivariate interactions between these processes may take place. If multivariate interactions occur, the iterative analysis of pairs of processes, i.e. bivariate analysis, may return spurious information transfer (Kamiński, Ding, Truccolo, & Bressler, 2001; Blinowska, Kuś, & Kamiński, 2004; J. T. Lizier & Rubinov, 2013). Spurious information transfer may result from one of two interaction schemes: cascade effects or common drive (Figure 2). TRENTOOL allows for a partial post-hoc correction of these spurious results using a graph-based algorithm on the network of information transfer in a multivariate system (Wibral et al., 2012). This post-hoc correction may be used on any information transfer network resulting from *neural data* as it uses assumption valid for neural systems only. The use of the post-hoc correction is described in more detail in section 4.7, *Graph correction for cascade effects and simple common drive.*

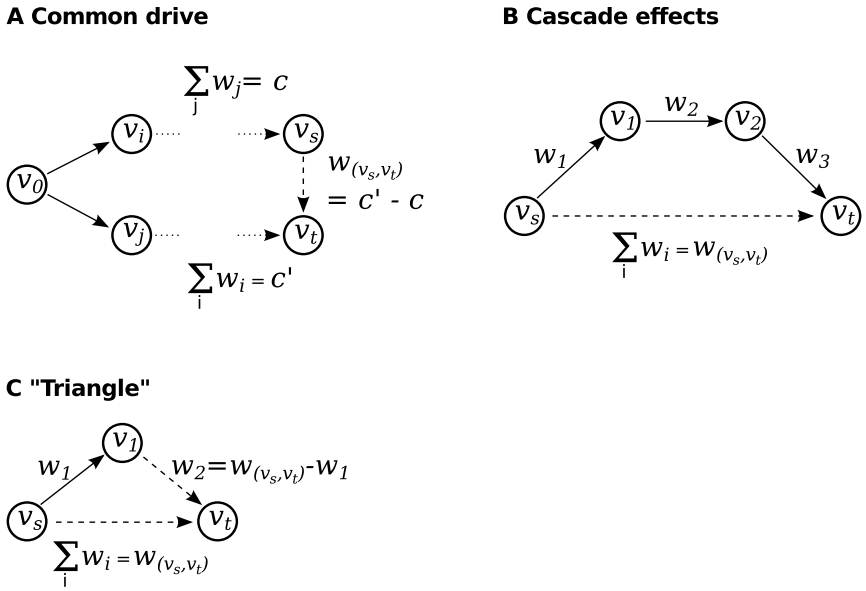

Figure 2: Network coupling schemes, that may lead to the detection of spurious information transfer. **A** Common drive: Processes v_s and v_t are driven by process v_0 with summed delays c and c' respectively. Bivariate analysis may detect a spurious information transfer between v_s and v_t. **B** Cascade effect: Information is transferred from v_s to v_t via nodes v_1 and v_2. Bivariate analysis may detect a spurious direct information transfer between v_s and v_t. **C** "Triangle": In a bivariately analyzed triangle, cascade effects ($v_s \overset{w_{(v_s,v_t)}}{\rightarrow} v_t$) and common drive effect ($v_1 \overset{w_{(v_1,v_t)}}{\rightarrow} v_t$) will not be distinguishable.

4 Using TRENTOOL

4.1 Overview

Analysis strategies TRENTOOL provides core functions for TE estimation as described in section 3.3. These core functions may be combined with the additional functionalities presented in section 3.4. Whether functionalities are applicable to a given set of data depends on the experimental design.

All TRENTOOL functionalities are accessible via calls to the respective MATLAB functions and may be combined into *analysis scripts*. Examples for analysis scripts for different analysis strategies can be found in section 5.

TRENTOOL core functions TRENTOOL's core functions for TE estimation provide all necessary functionalities to estimate TE from a set of time series. The core functions implement two main analysis steps (Figure 4):

1. data preparation (optimization of embedding parameters)
2. TE estimation from prepared data (data embedding using optimized parameters and TE calculation from nearest neighbor statistics)

Depending on the user's analysis script, the user calls functions for data preparation and TE estimation directly (Figure 3, panel A and B) or calls wrapper functions, that encapsulate both analysis steps and provide additional functionality (Figure 3, panel C and D). In general we recommend the use of the work flow for interaction delay reconstruction presented in section 4.3. This work flow may be used both for single subject analysis (TE estimation for individual subjects or conditions) as well as for group statistics (comparison of TE values between two sets of data, see section 4.6).

Organization of the following subsections We will first present TRENTOOL's core functions for data preparation and TE estimation for a single set of times series (e.g. a single subject measured in one experimental condition). We will then present analysis strategies building on the core functions in section 3.4. Example analysis scripts are provided in section 5, *Example Scripts*.

The following subsections are structured as follows: Each subsection describes one TRENTOOL analysis strategy. For each strategy, we first give a brief overview of the implementation of the pipeline, followed by detailed descriptions of the functions, users interact with. Detailed descriptions of input and output parameters are given in table form in section 6, *Appendix*.

Input data The main input to TRENTOOL consists of data to be processed and additional function parameters that specify how the data are processed. Data may be passed to TRENTOOL as a structure in the FieldTrip raw data format (table 1, see also (Oostenveld et al., 2011) and `http://fieldtrip.fcdonders.nl/`). Function parameters are passed within a MATLAB structure, following the general FieldTrip coding convention. Throughout this documentation, we will refer to these parameter structures as `cfgXXX`, where `XXX` is replaced by an abbreviation for the function that is called with the cfg structure in question.

Table 1
Input data in TRENTOOL: Fields in a FieldTrip raw data structure (TRENTOOL Version 3.3)

field name	dimension	data type	units	description
`trial`	{no. trials}[no. channels x no. samples]	cell array of double arrays		data for each trial, where each trial holds samples from all channels
`time`	{no. trials}[1 x no. samples]	cell array of double arrays	seconds	time indices for individual trials
`label`	{no. channels x 1}	cell of strings		labels of channels

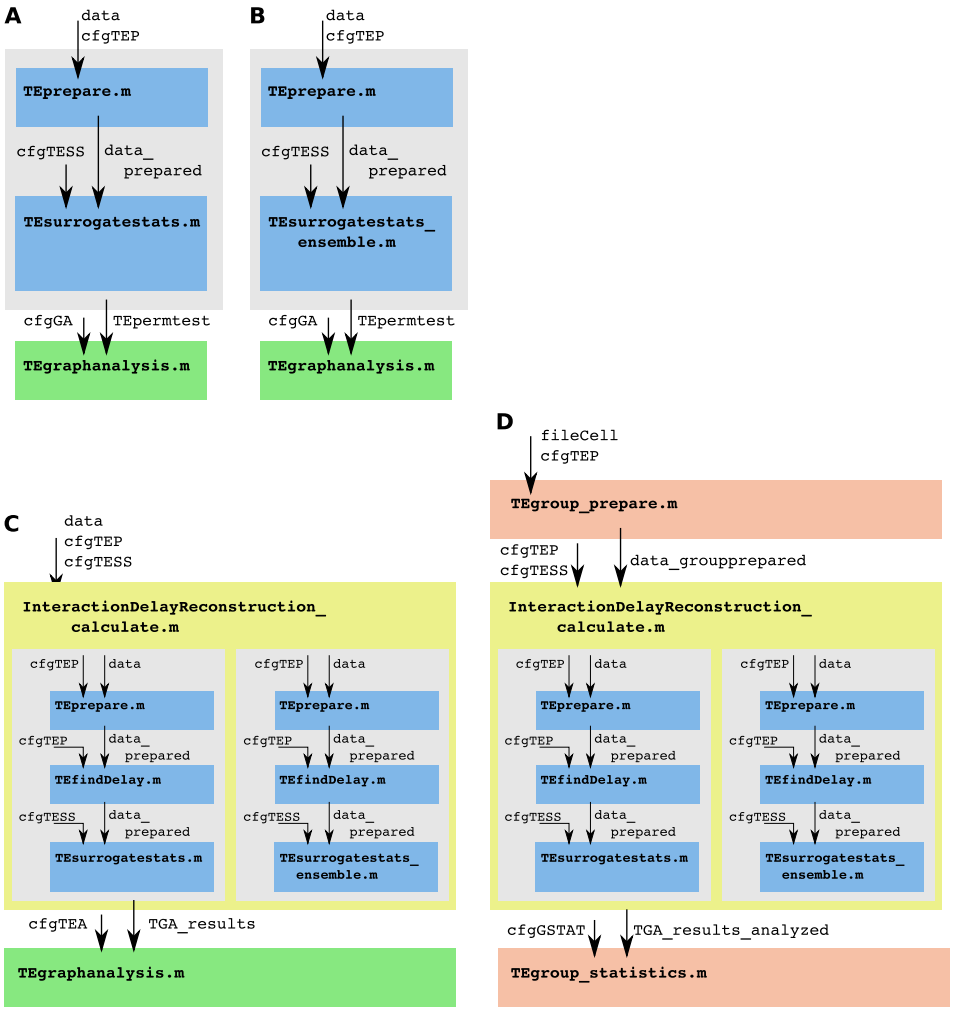

Figure 3: Overview of TRENTOOL functions: (A) TRENTOOL core functions for TE estimation (sec. 4.2); (B) TRENTOOL core functions for TE estimation using the ensemble method (sec. 4.4); (C) Reconstruction of interaction delays (sec. 4.3): The function `InteractionDelayReconstruction_calculate.m` calls functions `TEprepare.m`, `TEfindDelay.m`, and `TEsurrogatestats.m` or `TEsurrogatestats_ensemble.m` to prepare the data, reconstruct interaction delays and perform surrogate statistics on estimated TE values. (D) Group analysis (sec. 4.6: The function `TEgroup_prepare.m` prepares data sets for group analysis, TE is estimated using the functionality presented in (C), function `TEgroup_stats.m` tests TE values for significant differences; (A-C) TE estimates may be partially corrected for spurious information flow due to multivariate interactions using a graph algorithm (`TEgraphanalysis.m`, green boxes, see section 4.7).

4.2 TRENTOOL core functions

The TRENTOOL core functions realize the two analysis steps of data preparation and TE estimation from prepared data (Fig. 4). The functions comprise of `TEprepare.m` for data prepa-

ration and `TEsurrogatestats` for TE estimation from prepared data (following the approach lined out in section 3, *Background*). Note that TRENTOOL analyses *pairs* of signals only, i.e. TRENTOOL does not estimate multivariate TE from a set of time series, but estimates TE for pairs of time series iteratively (pairs of time series have to be defined by the user, see next subsection).

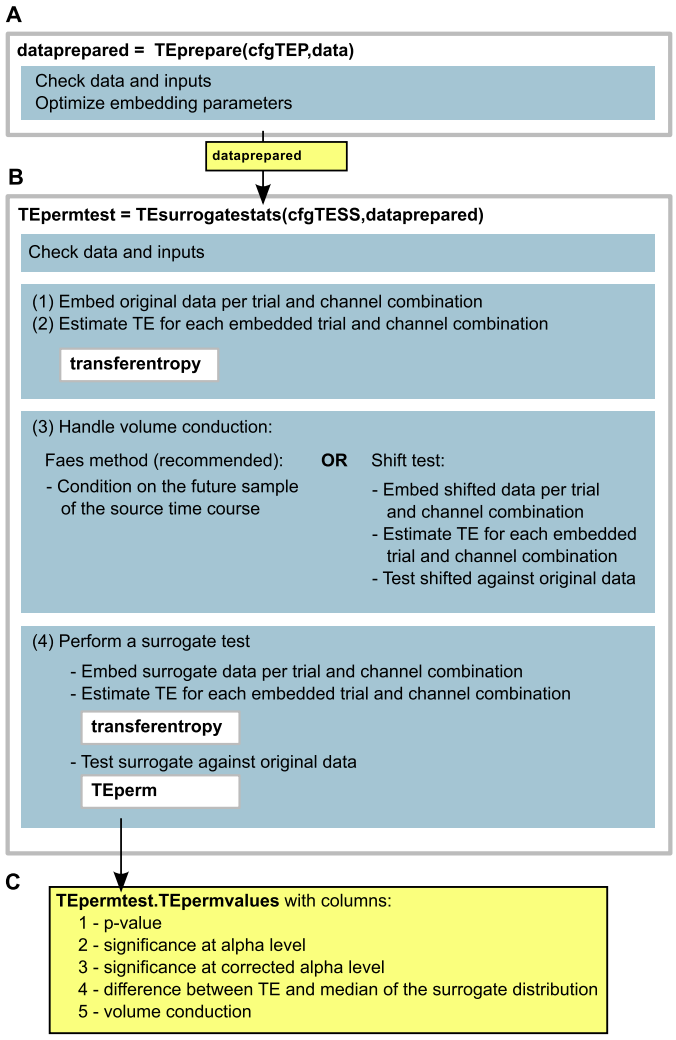

Figure 4: TRENTOOL core functions. (A) Data preparation and optimization of embedding parameters using `TEprepare.m`; (B) TE estimation and statistical testing from prepared data: (1) Embedding of original data and (2) TE estimation per trial and channel combination, (3) optional shift test (defined in `cfgTESS.shifttest`), (4) test against surrogate data using trial shuffling; (C) Output structure `TEpermtest`, containing the field `.TEpermvalues` holding TE results for each channel combination (see table 7).

4.2.1 `TEprepare.m`

Description `TEprepare.m` checks the input provided by the user and optimizes the embedding dimension d and embedding delay τ for subsequent TE estimation using Ragwitz' criterion ((Ragwitz & Kantz, 2002)).

Usage `data_prepared = TEprepare(cfgTEP,data);`

Input `data` contains data in the FieldTrip raw data format (table 1), `cfgTEP` contains a configuration structure with analysis parameters (table 4).

`cfgTEP` has to contain information regarding the combinations of signals the user wishes to analyze. Signal combinations can be defined by the user as a 2 by n cell array of strings, that contains n pairs of signal labels (`cfgTEP.sgncmb`). Alternatively, the user can provide a cell array of strings, that contains a list of signal labels from which TRENTOOL constructs all pairs of signal combinations (`cfgTEP.channel`).

If the user wants to use the ensemble method for TE analysis (see section 4.4), the field `cfgTEP.ensemblemethod` has to be set to `'yes'`. Parameters for TE estimation (`cfgTESS`) have to be changed accordingly (see next subsection and table 6).

Output Results (optimized embedding parameters, parameters for TE estimation) are appended to the data structure as a separate field `.TEprepare` (table 5). The resulting data structure `data_prepared` may be used as input to `TEsurrogatestats.m` or `TEsurrogatestats_ensemble.m` for TE estimation (see below).

4.2.2 `TEsurrogatestats.m`

Description `TEsurrogatestats.m` estimates TE from the prepared data returned by `TEprepare.m`, using the optimized embedding parameters. `TEsurrogatestats` performs the following analysis steps on the data:

- Check input data and parameters
- Estimate TE for original data:
  - Embed individual trials using optimized parameters
  - Perform nearest neighbor searches on reconstructed state spaces
  - Calculate TE from neighbor counts
- Optional: Perform a shift test or use extra conditioning to handle volume conduction (see below)
- Estimate TE for surrogate data (analogous to original data)
- Perform a statistical test of TE values from original data against surrogate TE values
- Save results

Usage `TEpermtest = TEsurrogatestats(cfgTESS,data_prepared);`

Input `data_prepared` contains data prepared by `TEprepare.m`, `cfgTESS` contains a configuration structure with analysis parameters (table 6).

It is recommended to use the embedding parameters optimized by `TEprepare.m` (see section 4.2.1) and not set `cfgTESS.dim` and `cfgTESS.tau` manually (see table 6).

The user may chose to perform a shift test on the data (`cfgTESS.shifttest = 'yes'`), to test for volume conduction in the data (Lindner et al., 2011). Alternatively, the user may use an extra conditioning (`cfgTESS.extracond = 'Faes_Method'`) (Faes et al., 2013), which includes the future sample of the source at the prediction time into the state vector of the past of of the target to condition on it (such that $TE = I\left(y^+; \mathbf{x}^- | \mathbf{y}^-; x^+\right)$). In principle, this removes any volume conduction effect and makes the conduction of a shift test unnecessary, thus the shift test and Faes Method are mutually exclusive. We recommend the use of extra conditioning over the shift test.

For the creation of surrogate data for statistical testing (`cfgTEP.surrogatetype`) one of the following strategies may be chosen (Lindner et al., 2011):

Parameter	**Strategy**	**Example** (original: 1 2 3 4 5 6)
`trialshuffling`	trial(n+1)	2 3 4 5 6 1
`trialreverse`	reverse of trial(n)	6 5 4 3 2 1
`blockresampling`	cuts trial(n) at random point and resamples the trial	4 5 6 1 2 3
`blockreverse1`	reverse after blockresampling	3 2 1 6 5 4
`blockreverse2`	reverse first block after blockresampling	6 5 4 1 2 3
`blockreverse3`	reverse second block after blockresampling	4 5 6 3 2 1
`swapneighbors`	pair odd trials with the higher neighbor and 3even with the lower neighbor	2 1 4 3 6 5

Output The results of TE estimation and statistical testing are returned to the user as a structure `TEpermtest` (see table 7). This structure contains the results of the statistical testing and is also save to disk as `*_TEpermtest_output.mat'`, using the string in `cfgTESS.fileidout` as prefix. Additionally, raw TE values are saved to disk (structure `TEresult`, table 8) as `*_TE_output.mat'`, again using `cfgTESS.fileidout` as prefix.

4.3 Reconstruction of interaction delays

TRENTOOL allows for the reconstruction of interaction delays between time series by scanning over assumed interaction delays u (Wibral et al., 2013). The true interaction delay may be estimated by finding the maximum TE value over all scanned interaction delays. This functionality is provided within the function `InteractionDelayReconstruction_calculate.m` (Figure

5): The function estimates TE for each channel combination and assumed delay u using the TRENTOOL core functions (section 4.2). Estimated TE values are passed to the function `InteractionDelayReconstruction_analyze.m`, that finds the maximum TE value over all u for each channel combination (eq. 8). TE values at this interaction delay are then tested for statistical significance using the functions `TEsurrogatestats.m` or `TEsurrogatestats_ensemble.m` (section 4.4, *Estimating time-resolved transfer entropy from an ensemble of time series*).

The approach for interaction delay reconstruction may also be used together with group analysis functionalities (section 4.6, *Group analysis in TRENTOOL*). For an example analysis, see example scripts 3 and 4.

Note: Using the functions `InteractionDelayReconstruction_*` is the recommended analysis strategy for most neural data sets.

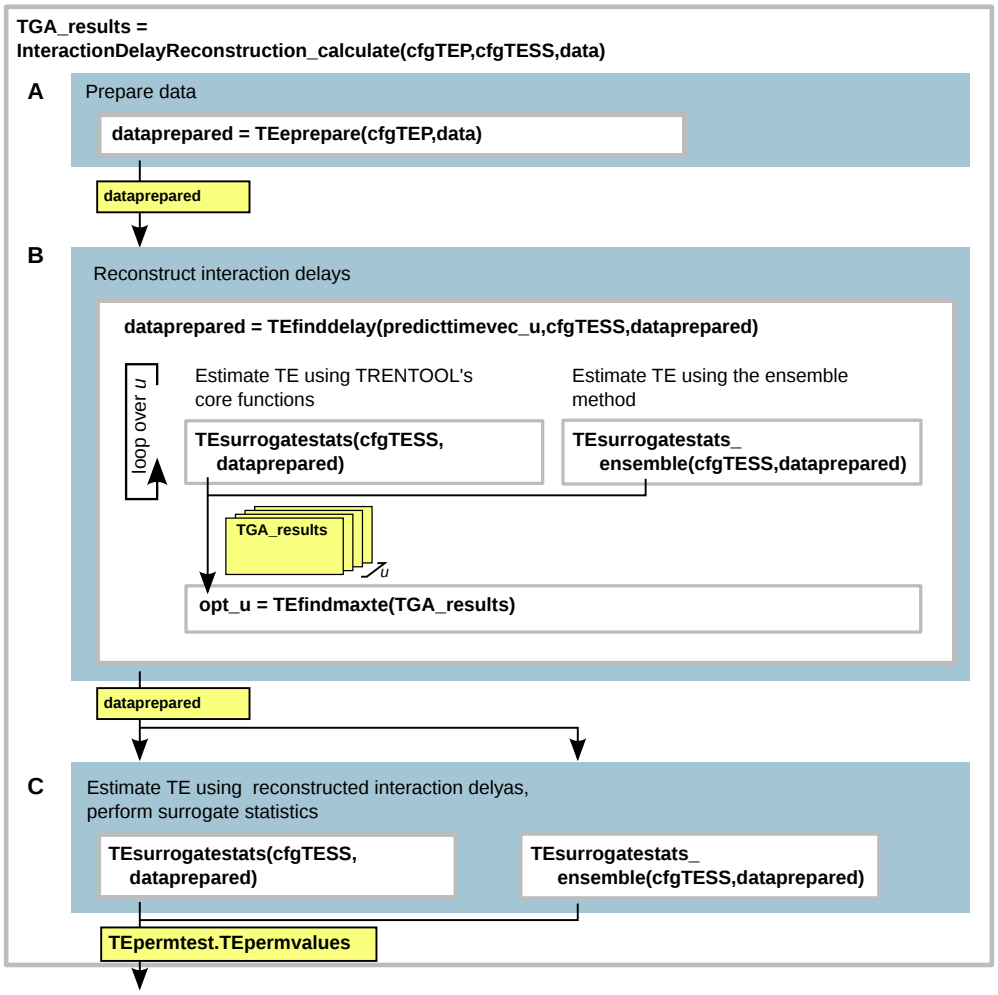

Figure 5: TRENTOOL work flow for interaction delay reconstruction. The function `InteractionDelayReconstruction_calculate` first calls `TEprepare` on the input data (A). Prepared data is then passed to `TEfindDelay` (B) to reconstruct the optimal interaction delay by looping over TE estimation using varying values for u. TE may be estimated using `TEsurrogatestats` (see section 4.2) or `TEsurrogatestats_ensemble` (see section 4.4), while using `TEsurrogatestats_ensemble` allows the user to additionally calculate time-resolved TE. The function `TEfindmaxte` finds the u that maximizes TE and adds this u_{opt} to every channel combination in the `dataprepared` structure. In a third step (C) final TE values are estimated from `dataprepared`, using the reconstructed interaction delays and a statistical test is performed.

4.3.1 `InteractionDelayReconstruction_calculate.m`

Description Wrapper function that estimates TE using optimized values for the interaction delay(see eq. 3). The function (1) prepares the data (calls `TEprepare.m`); (2) optimizes interaction delays by estimating TE for all assumed values u and finding the value u that maximizes

TE; (3) performes a statistical test on TE values estimated using the optimized interaction delay Figure 5). TE values are estimated by the functions `TEsurrogatestats.m` (see also section 4.2) or `TEsurrogatestats_ensemble.m` (see also section 4.4).

Usage `TGA_results = InteractionDelayReconstruction_calculate(cfgTEP,cfgTESS, data);`

Input `data` contains input data in the FieldTrip raw data format (see section 4.1 and table 1), `cfgTEP` (table 4) and `cfgTESS` (table 6) contain configuration structures with analysis parameters for functions `TEprepare.m` and `TEsurrogatestats.m`, that are called inside the function.

Output The interaction delay and the associated statistics are returned as a structure `TE_permtest` similar to the structure returned by `TEsurrogatestats.m` (table 7). Additionally, the table in the field `.TEpermvalues` contains a sixth column that provides the optimal u for each channel combination. Final results (output `TEpermtest`) are saved to disk using the string in `cfgTESS.fileidout` as prefix. These results may be visualized using the function `InteractionDelayReconstruction\_plotting` (see next subsection).

4.3.2 `InteractionDelayReconstruction_plotting.m`

Description Function plots final results from TE analysis with `InteractionDelayReconstruction_calculate.m`. The function plots TE values against assumed delays u for each channel combination in the data set.

Usage `InteractionDelayReconstruction_plotting(cfgUPLOT, TEpermtest);`

Input `cfgUPLOT` (table 14) contains a configuration structure with plotting options. `TE_permtest` is the output of `InteractionDelayReconstruction_calculate.m`.

Output The function plots normalized TE values (TE/TE_{max}) for each channel combination against assumed interaction delays u.

4.4 Estimating time-resolved transfer entropy from an ensemble of time series

TRENTOOL allows for the estimation of TE from an ensemble of time series (see (Wollstadt et al., 2013) and (Gomez-Herrero et al., 2010)). By using this ensemble approach, TRENTOOL allows for the estimation of TE from non-stationary processes and furthermore enables TE estimation in a time-resolved fashion. The ensemble approach is computationally demanding, thus TRENTOOL requires the use of a graphics processing unit (GPU) to speed up TE estimation when using the ensemble approach.

TRENTOOL realizes the time-resolved estimation of TE by using user-defined analysis windows. Analysis windows can be defined by looping over calls to `TEprepare.m` and `TEsurrogatestats_ensemble.m` with varying times of interests specified in `cfgTEP.toi` (see example script 3). Furthermore, time-resolved TE estimation may be combined with the reconstruction of interaction delays (see below) and group statistics functionalities (see sections 4.6, *Group analysis in TRENTOOL*). For both analysis strategies, the user has to loop over calls to `InteractionDelayReconstruction_calculate.m` using varying analysis windows specified in `cfgTEP.toi` to estimate time-resolved TE (example script 4).

Using the ensemble method in TRENTOOL is similar to the usage of TRENTOOL's core functions (section *TRENTOOL core functions*), with the exception that TE estimation is handled by the function `TEsurrogatestats_ensemble.m` instead of `TEsurrogatestats.m`. Here, we will present the usage of `TEsurrogatestats_ensemble.m` only as the use of `TEprepare.m` for data preparation prior to TE estimation remains the same as described in section 4.2. For an example analysis, see 4.

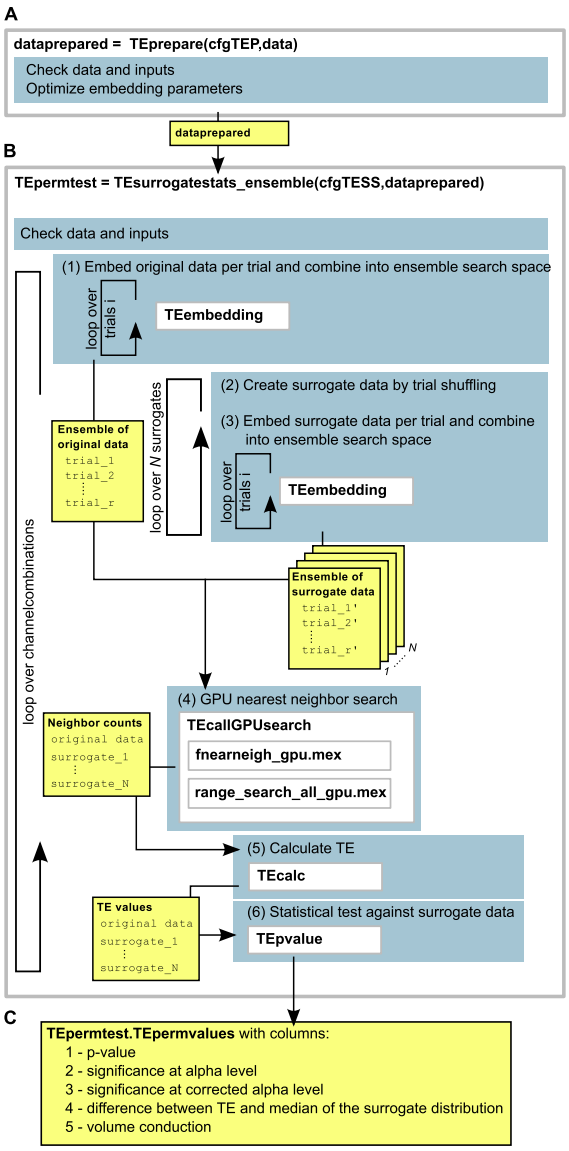

Figure 6: Using the ensemble method in TRENTOOL. (A) Data preparation and optimization of embedding parameters using `TEprepare.m`; (B) TE Estimation using a GPU for nearest neighbor searches; (C) Output structure `TEpermtest` containing the field `.TEpermvalues` holding TE results for each channel combination (see table 7).

4.4.1 `TEsurrogatestats_ensemble.m`

Description The function estimates TE from the prepared data returned by `TEprepare.m`, using the optimized embedding parameters. The function estimates TE in six steps (Figure 6):

1. original data is embedded per repetition and repetitions are concatenated forming the

ensemble search space

2. N sets of surrogate data are created from the original data by shuffling the repetitions of the target time series,
3. each surrogate dataset is embedded per repetition and concatenated forming one ensemble search space
4. all embedded original and surrogate data ensembles are concatenated and passed to a wrapper function that calls the GPU search functions to perform neighbor counts in parallel for each ensemble
5. TE values are calculated for original and surrogate data from the ensemble neighbor counts using the KSG-estimator (Kraskov et al., 2004),
6. TE values for original data are tested statistically against the distribution of surrogate TE values.

Usage `TEpermtest = TEsurrogatestats_ensemble(cfgTESS,data_prepared);`

Input `data_prepared` contains data prepared by `TEprepare.m`, `cfgTESS` contains a configuration structure with analysis parameters (table 6). Note, that the use of the GPU has to be specified in the field `cfgTEP.ensemblemethod`, when calling `TEprepare.m` for data preparation. Additionally, the GPU's hardware parameters have to be provided to `TEsurrogatestats_ensemble.m` (`cfgTESS.GPUmemsize`, `cfgTESS.numthreads`, `cfgTESS.maxgriddim`).

When using the ensemble method, the use of the Faes-Method is mandatory (`cfgTESS.extracond = 'Faes_Method'`) and the shift test has to be switched off (`cfgTESS.shifttest = 'no';`). The use of the Faes method prevents the detection of information transfer due to volume conduction (a shift test can not be used for the ensemble method as TE values are no longer calculated per trial).

Additionally, when using the ensemble method, the strategy for the creation of surrogate data (section 4.2.1) has to be set to `'trialperm'` (`cfgTEP.surrogatetype = 'trialperm'`), which simply permutes trials.

Output The output structure `TEpermtest` is similar to the output of `TEsurrogatestats.m` (table 7).

4.4.2 Combining time-resolved transfer entropy estimation with interaction delay reconstruction

The ensemble approach for TE estimation may be combined with the reconstruction of interaction delays. To combine both approaches, the user may follow the interaction delay reconstruction work flow (Figure 5) and specify the use of the ensemble method in the configuration structure for TEprepare (`cfgTEP.ensemblemethod = 'yes'`). The function `InteractionDelayReconstruction_calculate.m` will use `TEsurrogatestats_ensemble.m` instead of `TEsurrogatestats.m` for TE estimation. The rest of the work flow remains unchanged. For an example analysis, see 4.

4.5 Binomial Test for Single Subject Results

TE results for single data sets (returned by `InteractionDelayReconstruction_calculate`,section 4.3, or by `TEsurrogatestats`/`TEsurrogatestats_ensemble` directly, sections and 4.4) may be combined into a group result using a binomial test. The binomial test tests for deviations from an expected distribution of occurrences of a binary random variable. Thus, the binomial test tests for every pair of sources whether the number of significant information transfers over subjects is significantly higher than expected over a binomial distribution $B(n, p)$. The parameters of the distribution are set as follows:

- n: number of subjects,
- p: significance level of the individual tests for significance of information transfer against surrogate data in single subjects.

Binomial testing of single subject result is provided in the function `TEsurrogate_binomstats.m`.

4.5.1 `TEsurrogate_binomstats.m`

Description Performs a binomial test over the results from single subject analysis for each pair of signals in the data. The function tests if significant information transfer between two signals is found significantly more often than expected under a binomial distribution $B(n, p)$.

Usage `TEbinomstats = TEsurrogate_binomstats(cfgBS,fileCell);`

Input `fileCell` contains a cell array with file names, where each file holds an output structure returned by `InteractionDelayReconstruction_calculate`, `TEsurrogatestats`, or `TEsurrogatestats_ensemble` *or* a cell array of output structures. `cfgBS` contains a configuration structure with a field `alpha` that specifies the alpha level for the binomial test.

Output The function returns a structure `TEbinomstats` (table **??**) with statistical results for every signal combination as well as information about the CDF of the binomial distribution with parameters n and p.

4.6 Group analysis in TRENTOOL

TRENTOOL allows to statistically test for differences in TE values between two sets of data: either data obtained from two groups, data obtained from two conditions in one group, or data obtained from two conditions in a single subject (Figures 8 and 7) (Lindner et al., 2011). If two conditions are compared within a single subject, a sufficient number of trials has to be recorded for each condition (see also section 6.1 in the appendix). To not introduce a bias in the raw TE values and thus to make TE values comparable in a statistical test between groups, identical embedding dimensions have to be used for TE estimation in individual data sets. To find a common embedding dimension over all data sets, the function `TEgroup_prepare.m` has to be used on all data sets that are supposed to enter the analysis. The function calls `TEprepare.m` on each data set and finds the common, i.e., maximum embedding dimension over all data. This common embedding dimension is then used in the following TE estimation, using the work flow for delay

reconstruction (section 4.3). Resulting TE values can then be tested for statistical differences between both groups of data using `TEgroup_stats.m` or `TE_conditionstatssingle.m`. See also the example script 5.

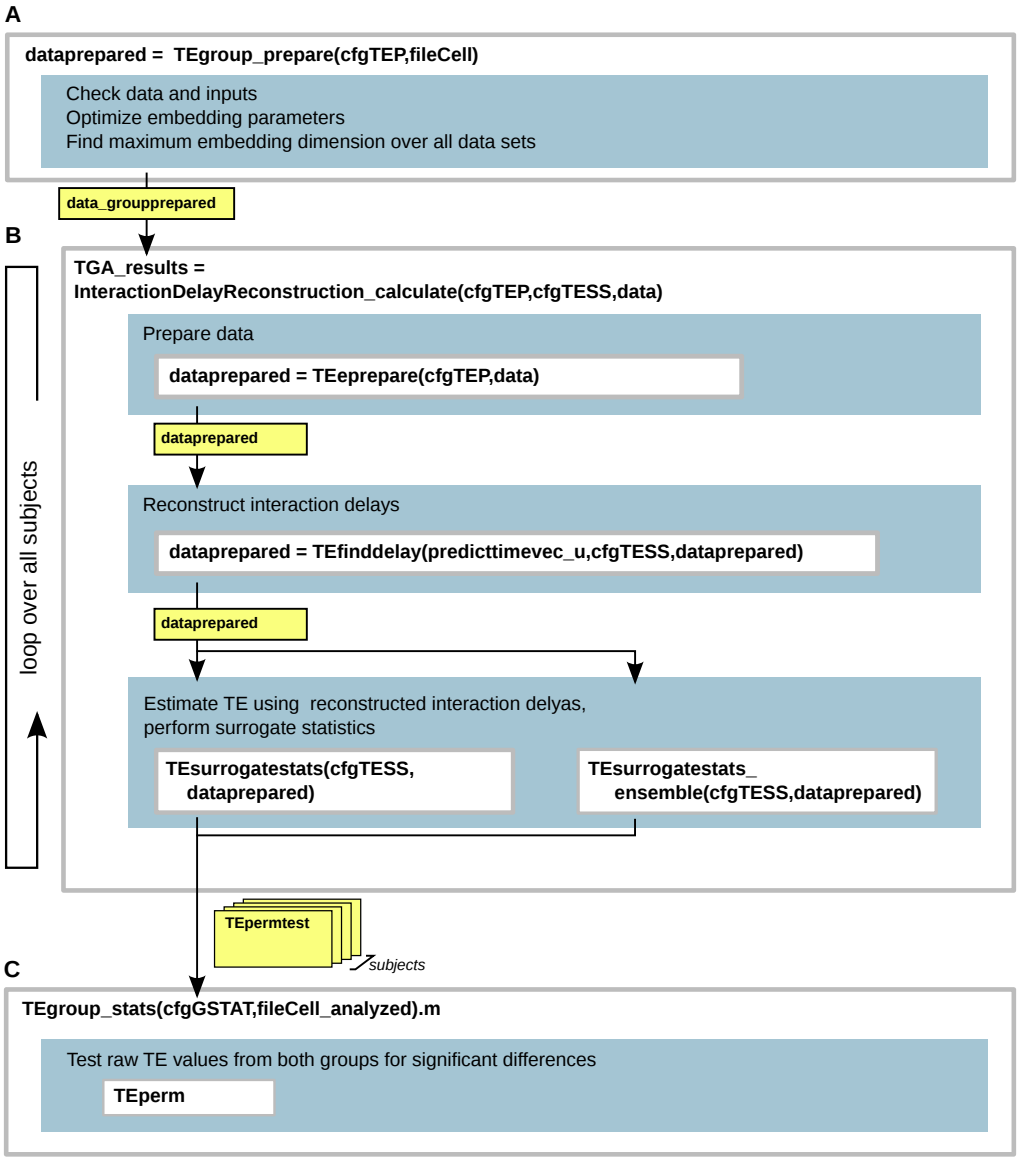

Figure 7: TRENTOOL work flow for group analysis: (A) All data sets have to be prepared for group analysis using the function `TEgroup_prepare`, the function finds the maximum embedding dimension over all data sets/channel combinations and writes results to a field `grouppre_pare` in the data; (B) TE is estimated from group-prepared single subject data, the function `InteractionDelayReconstruction_calculate.m` recognizes data prepared for group analysis and uses group parameters from `data_grouppprepared.grouppprepare` for TE analysis; (C) Raw TE values are compared between both groups by means of permutation testing, results are written to disk.

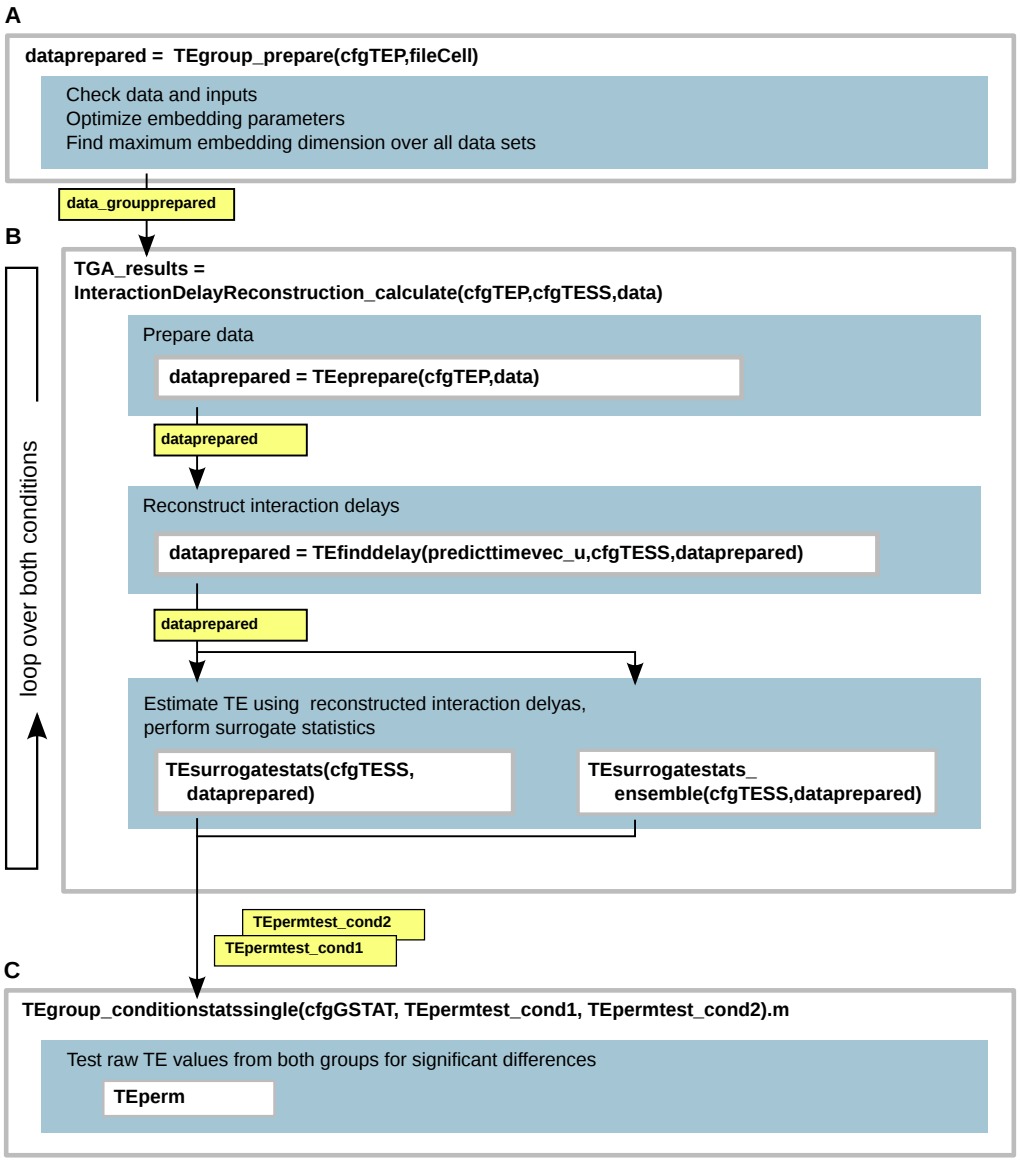

Figure 8: TRENTOOL work flow for comparison of two conditions, withing a single subject: (A) Both data sets (one for each condition) have to be prepared for group analysis using the function `TEgroup_prepare`, the function finds the maximum embedding dimension over all data sets/channel combinations and writes results to a field `groupprepare` in the data; (B) TE is estimated from group-prepared data for both conditions, the function `InteractionDelayReconstruction_calculate.m` recognizes data prepared for group analysis and uses group parameters from `data_groupprepared.groupprepare` for TE analysis; (C) Raw TE values are compared between both conditions by means of permutation testing, results are written to disk.

4.6.1 `TEgroup_prepare.m`

Description The function prepares all data for group analysis (from both conditions/groups) by finding the group embedding dimension to be used for subsequent TE estimation. All data sets are loaded by `TEgroup_prepare.m` and passed to `TEprepare.m` individually. From all individually optimal embedding parameters, the maximum embedding dimension is kept for subsequent TE analysis. Furthermore, the function checks whether all data sets use the same channel combination and assumed interaction delays u to be scanned during TE estimation.

Usage `data_groupprepared = TEgroup_prepare(cfgTEP,fileCell);`

Input `fileCell` contains a cell array of file names (not paths, i.e. the function needs to be called from the folder containing the data sets), `cfgTEP` contains a configuration structure with analysis parameters (table 4).

Output Results from `TEgroup_prepare.m` are added to all data sets as an extra field `.groupprepare` (table 9). Data are then saved to a user specified location.

4.6.2 Transfer entropy estimation

In its current version, TRENTOOL requires the use of the function `InteractionDelayReconstruction_calculate.m` for TE estimation in group analysis. The function is called individually for each data set, prepared for group analysis. The work flow for TE analysis is the same as in single subject analysis (see section 4.3, *Reconstruction of interaction delays*). Both the core TRENTOOL functions and the ensemble method for TE estimation (section 4.4) may be used.

The function `InteractionDelayReconstruction_calculate.m` recognizes data prepared for group analysis and replaces relevant fields in `cfgTEP` with the group parameters found in `data\_grouprepared.groupprepare` (embedding dimension and values for u to be scanned). The function finds the optimal u and estimates TE values using the optimal u and group embedding parameters. TE estimates for individual data sets are returned to the user.

4.6.3 `TEgroup_stats.m`

Description The function compares raw TE values estimated for all data sets and channel combinations by means of permutation testing (Maris & Oostenveld, 2007; Lindner et al., 2011).

Usage `TEgroup_stats(cfgGSTAT,fileCell_analyzed);`

Input `fileCell` contains a cell array of names of files that have been prepared for group analysis by running `TEgroup_prepare.m` and have been analyzed individually, using the interaction delay reconstruction work flow (see 4.3). `cfgGSTAT` contains a configuration structure with analysis parameters (table 10).

Subjects are assigned to groups using a design matrix specified in `cfgGSTAT.design` (table 2). The design matrix specifies group membership for statistical testing.

Table 2
Example design matrix to be used in `cfgGSTAT.design` as a parameter for statistical testing in `TEgroup_stats.m`. The first row holds the indices of data sets (in this case, two conditions are compared), the second row encodes the condition of the respective data set.

dataset	(`uvar`)	1	2	3	4	5	1	2	3	4	5
condition	(`ivar`)	1	1	1	1	1	2	2	2	2	2

Output The function `TEgroup_stats.m` writes two files with suffixes `TE_output.mat` and `TEpermtestgroup_output.mat` to disc (location is specified in `cfgGSTAT.fileidout`). `TE_output.mat` contains the raw TE values of both sets of data (`TEresult1` and `TEresult2`) as well as the mean over both sets (`TEresultmean`). `*TEpermtestgroup_output.mat` contains the structure `TEpermtestgroup`, that holds the result of the statistical testing. Results are provided in the same fashion as for single subject analysis (see table 7), i.e. for each channel combination results of the permutation test are given in table form (`.TEpermvalues`). Additionally, the output structure contains a field `.TEgroupprepare`, that provides the results of `TEgroup_prepare.m`.

4.6.4 `TEgroup_conditionstatssingle.m`

Description The function compares raw TE values estimated for two conditions in a single subject by means of permutation testing (Maris & Oostenveld, 2007; Lindner et al., 2011). See also Figure 8.

Usage `TEgroup_conditionstatssingle(cfgCSTAT,TEpermtest_cond1,TEpermtest_cond2);`

Input `TEpermtest_cond1` and `TEpermtest_cond2` are two output structures that contain TE values estimated from two data sets, where each data set represents one of two conditions, recorded in a single subject. TE values have to be estimated with the group analysis work flow (data preparation with `TEgroup_prepare.m`, TE estimation with `InteractionDelayReconstruction_calculate.m`). `cfgCSTAT` contains a configuration structure with analysis parameters (table 11).

Output The function `TEgroup_conditionstatssingle.m` writes a files with suffix `TEpermtestcondsingle_output.mat` to disc (location is specified in `cfgCSTAT.fileidout`). The output is astructure `TEpermtestcondsingle`, with the statistical testing. Results are provided in the same fashion as for single subject analysis (see table 7), i.e. for each channel combination results of the permutation test are in a table (`.TEpermvalues`). Additionally, the output structure contains information on the statistical test and a field `.TEgroupprepare`, that provides the results of `TEgroup_prepare.m`.

4.7 Graph correction for cascade effects and simple common drive

TRENTOOL allows for the partial correction of spurious information flow that may be introduced by bivariate analysis of a highly multivariate system. The description of the applied algorithm can be found in (Wollstadt et al., 2013). The algorithm is provided in TRENTOOL as the function `TEgraphanalysis.m` (Figure 9) and may be called on the output of any function that returns TE estimates: `TEsurrogatestats.m`, `TEsurrogatestats_ensemble.m`, and `InteractionDelayReconstruction_calculate.m` (Figure 4).

`TEgraphanalysis.m`

Description The function corrects the network of significant information transfer for multivariate effects, namely cascade effects and simple common drive effects. The function tagss respective edges (significant information transfer between two time series) as potentially spurious and returns a modified output structure to the user.

Usage `TEpermtest_GA = TEgraphanalysis(cfgGA,TEpermtest);`

Input `TEpermtest` can contain any structure with the field `.TEpermvalues`, that is returned by one of the functions that handle TE estimation: `TEsurrogatestats.m`, `TEsurrogatestats_ensemble.m`, or `InteractionDelayReconstruction_calculate.m`. `cfgGA` contains a configuration structure with analysis parameters (table 12).

Output The function returns the results structure that was used as input with a modified table in field `.TEpermvalues` and a field `.graphanalysis` with information on the information transfer network (table 13).

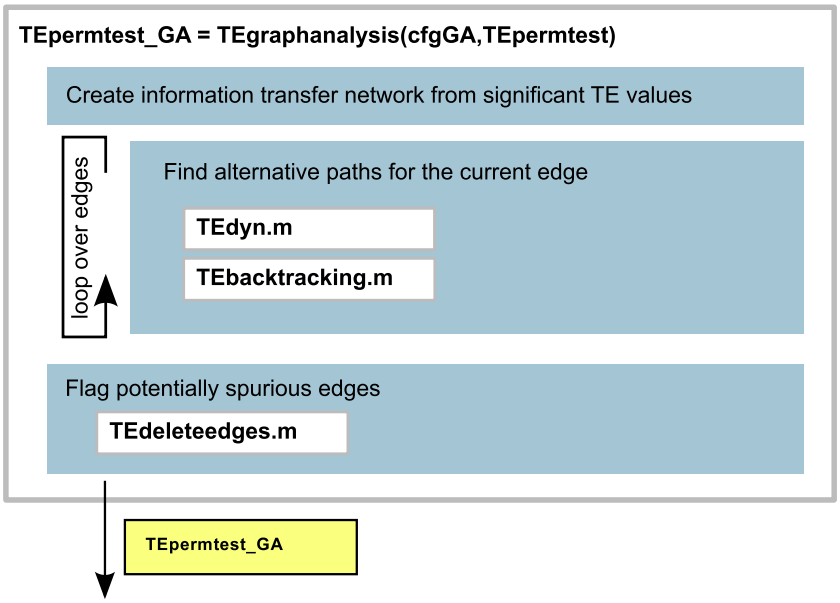

Figure 9: Post-hoc correction of spurious information transfer using a graph-based algorithm (correction for cascade effects and simple common drive).

4.8 Plotting of TRENTOOL results

In the current version, TRENTOOL offers two options to visualize results from TE estimation.

4.8.1 `TEplot2D.m`

Description `TEplot2D.m` plots the resulting network of information transfer for human neuronal data. The function plots results returned by functions `TEsurrogatestats.m`, `TEsurrogatestats_ensemble.m`, `InteractionDelayReconstruction_calculate.m` or `TEgroup_stats.m`. Information transfer is plotted as arrows between sources or channels into a 2D head model.

Usage `InteractionDelayReconstruction_plot(cfg2DPLOT,TEresult_analyzed);`

Input `cfg2DPLOT` contains a configuration structure with plotting parameters (table 15), `TEresult_analyzed` contains the structure `TEpermtest` returned by one of the above mentioned functions for TE analysis.

Output The function plots the network of information transfer using a user-provided layout of the sources.

4.8.2 TRENTOOL2BrainNet.m

Description TRENTOOL2BrainNet.m exports results of TE estimation into a file format readable by BrainNet Viewer (http://www.nitrc.org/projects/bnv/, (Xia, Wang, & He, 2013)). The function plots results returned by functions TEsurrogatestats.m, TEsurrogatestats_ensemble.m, InteractionDelayReconstruction_calculate.m or TEgroup_stats.m. The function produces node and edge files, that can be read into BrainNet Viewer to produce 3D visualization of information transfer networks using MNI coordinates.

Usage TRENTOOL2BrainNet(cfgBN,TEpermtest);

Input cfgBN contains a configuration structure with export parameters (table 16), TEpermtest contains the structure TEpermtest returned by one of the above mentioned functions for TE analysis.

Output The function writes two files to disk: *.node and *.edge. *.node contains information regarding source positions in the 3D template in BrainNet Viewer. *.edge contains a connectivity matrix that defines the edges to be plotted by BrainNet Viewer (the label ordering corresponds to the ordering of sources in the *.node-file). See also the BrainNet Viewer manual (http://www.nitrc.org/docman/view.php/504/1280/BrainNetViewerManual1.42.pdf).

5 Example scripts

5.1 Example analysis: Interaction delay reconstruction

The following example script uses the functions `InteractionDelayReconstruction_*` for the reconstruction of interaction delays together with the core TRENTOOL functions for TE estimation. The example script `example_script_CPUmethod.m` and data `exampledata/Lorenz_1->2_45ms.mat` can be found in the current TRENTOOL version.

Listing 3: Example script for interaction delay reconstruction in TRENTOOL, using the core TRENTOOL functions for TE estimation.

```
%% set paths

addpath('../TRENTOOL3')
addpath('../fieldtrip-20150928');
ft_defaults;

%% define data paths

OutputDataPath = '../results/';
InputDataPath  = 'exampledata/Lorenz_1->2_45ms.mat';

load(InputDataPath);


%% define cfg for TEprepare.m

cfgTEP = [];

% data
cfgTEP.toi                 = [min(data.time{1,1}),max(data.time{1,1})]; % time of
    interest
cfgTEP.sgncmb              = {'A1' 'A2'};  % channels to be analyzed

% scanning of interaction delays u
cfgTEP.predicttimemin_u    = 40;      % minimum u to be scanned
cfgTEP.predicttimemax_u    = 50;          % maximum u to be scanned
cfgTEP.predicttimestepsize = 1;           % time steps between u's to be scanned

% estimator
cfgTEP.TEcalctype  = 'VW_ds'; % use the new TE estimator (Wibral, 2013)

% ACT estimation and constraints on allowed ACT(autocorelation time)
cfgTEP.actthrvalue = 100;   % threshold for ACT
cfgTEP.maxlag      = 1000;
cfgTEP.minnrtrials = 15;   % minimum acceptable number of trials

% optimizing embedding
cfgTEP.optimizemethod ='ragwitz';  % criterion used
cfgTEP.ragdim         = 2:9;       % criterion dimension
cfgTEP.ragtaurange    = [0.2 0.4]; % range for tau
cfgTEP.ragtausteps    = 5;        % steps for ragwitz tau steps
cfgTEP.repPred        = 100;      % size(data.trial{1,1},2)*(3/4);

% kernel-based TE estimation
cfgTEP.flagNei = 'Mass' ;           % neigbour analyse type
cfgTEP.sizeNei = 4;                 % neigbours to analyse


```

```
%% define cfg for TEsurrogatestats_ensemble.m

cfgTESS = [];

% use individual dimensions for embedding
cfgTESS.optdimusage = 'indivdim';

% statistical and shift testing
cfgTESS.tail           = 1;
cfgTESS.numpermutation = 5e4;
cfgTESS.shifttesttype  ='TEshift>TE';
cfgTESS.surrogatetype  = 'trialshuffling';

% results file name
cfgTESS.fileidout  = strcat(OutputDataPath,'Lorenzdata_1->2_');

%% calculation - scan over specified values for u

TGA_results = InteractionDelayReconstruction_calculate(cfgTEP,cfgTESS,data);

save([OutputDataPath 'Lorenz_1->2_TGA_results.mat'],'TGA_results');

%% optional: perform a post hoc correction for cascade effects and simple common
    drive effects

cfgGA = [];

cfgGA.threshold = 3;
cfgGA.cmc = 1;

TGA_results_GA = TEgraphanalysis(cfgGA,TGA_results_analyzed);

save([OutputDataPath 'Lorenz_1->2_TGA_results_analyzed_GA.mat'],'TGA_results_GA');



%% plotting

load('exampledata/Lorenz_layout.mat');

cfgPLOT = [];

cfgPLOT.layout        = lay_Lorenz;             % see fieldtrip's
    ft_prepare_layout.m
cfgPLOT.electrodes    = 'highlights';
cfgPLOT.statstype     = 1;              % 1: corrected; 2:uncorrected; 3: 1-pval;
    4:rawdistance
cfgPLOT.alpha         = 0.05;
cfgPLOT.arrowpos      = 1;
cfgPLOT.showlabels    = 'yes';
cfgPLOT.electrodes    = 'on';
cfgPLOT.hlmarker      = 'o';
cfgPLOT.hlcolor       = [0 0 0];
cfgPLOT.hlmarkersize  = 4;
cfgPLOT.arrowcolorpos = [1 0 0];

figure;
TEplot2D(cfgPLOT,TGA_results_analyzed_GA)
```

5.2 Example analysis: Interaction delay reconstruction with ensemble method for TE estimation

The following example script uses the ensemble method for TE estimation together with functions `InteractionDelayReconstruction_*` for the reconstruction of interaction delays. Note, that in comparison to the first example script, the ensemble method allows for the analysis of shorter time windows of interest (in the example, '`toi`' is set to 120 msec). These short analysis windows can typically not be realized without the use of the ensemble method as the number of data points is not sufficient.

To account for the shorter analysis windows and the reduction in data points per repetition, parameters for the embedding optimization have to be adjusted (e.g. `cfgTEP.maxlag` or `cfgTEP.repPred`).

The example script `example_script_ensemblemethod.m` and data `exampledata/Lorenz_1->2_45ms.mat` can be found in the current TRENTOOL version.

Listing 4: Example script for interaction delay reconstruction in TRENTOOL.

```
%% set paths

addpath('../TRENTOOL3')             % note: this script assumes, you have already compiled the GPU mex-files using 'install.m'
addpath('../fieldtrip-20150928');
ft_defaults;

%% define data paths

OutputDataPath = '../results/';
InputDataPath  = 'exampledata/Lorenz_1->2_45ms.mat';

load(InputDataPath);

%% define cfg for TEprepare.m

cfgTEP = [];

% data
cfgTEP.toi                 = [min(data.time{1,1}),max(data.time{1,1})]; % time of interest
cfgTEP.sgncmb              = {'A1' 'A2'};  % channels to be analyzed

% scanning of interaction delays u
cfgTEP.predicttimemin_u    = 40;                    % minimum u to be scanned
cfgTEP.predicttimemax_u    = 50;          % maximum u to be scanned
cfgTEP.predicttimestepsize = 1;           % time steps between u's to be scanned

% estimator
cfgTEP.TEcalctype  = 'VW_ds'; % use the new TE estimator (Wibral, 2013)

% use ensemble method
cfgTEP.ensemblemethod = 'yes';

% ACT estimation and constraints on allowed ACT(autocorelation time)
cfgTEP.actthrvalue = 40;   % threshold for ACT
cfgTEP.maxlag      = 100;
cfgTEP.minnrtrials = 15;   % minimum acceptable number of trials

% optimizing embedding
cfgTEP.optimizemethod ='ragwitz';  % criterion used
cfgTEP.ragdim         = 2:8;       % criterion dimension
```

```
cfgTEP.ragtaurange    = [0.2 0.4]; % range for tau
cfgTEP.ragtausteps    = 15;        % steps for ragwitz tau steps
cfgTEP.repPred        = 100;       % points used for optimization the embedding
    dimensions

% kernel-based TE estimation
cfgTEP.flagNei = 'Mass' ;           % neigbour analyse type
cfgTEP.sizeNei = 4;                 % neigbours to analyse


%% define cfg for TEsurrogatestats_ensemble.m

cfgTESS = [];

% use individual dimensions for embedding
cfgTESS.optdimusage = 'indivdim';

% surrogate testing
cfgTESS.tail           = 1;
cfgTESS.surrogatetype  = 'trialperm';
cfgTESS.numpermutation = 100;

% GPU specifications
cfgTESS.GPUmemsize     = 4200;
cfgTESS.numthreads     = 512;
cfgTESS.maxgriddim     = 65535;

% volume conduction
cfgTESS.extracond      = 'Faes_Method';
cfgTESS.shifttest      = 'no';

% calculation of mutual information (MI)
cfgTESS.MIcalc = 1;

% results file name
cfgTESS.fileidout  = strcat(OutputDataPath,'Lorenzdata_1->2_ensemble_');


%% calculation - scan over specified values for u

TGA_results = InteractionDelayReconstruction_calculate(cfgTEP,cfgTESS,data);

save([OutputDataPath 'TGA_results.mat'],'TGA_results');


%% optional: perform a post hoc correction for cascade effects and simple common
    drive effects

cfgGA = [];

cfgGA.threshold = 3;
cfgGA.cmc = 1;

TGA_results_GA = TEgraphanalysis(cfgGA,TGA_results);

save([OutputDataPath 'Lorenz_1->2_TGA_results_analyzed_GA.mat'],'TGA_results_GA');



%% plotting

load('exampledata/Lorenz_layout.mat');
```

```

cfgPLOT = [];

cfgPLOT.layout        = lay_Lorenz;             % see fieldtrip's
    ft_prepare_layout.m
cfgPLOT.electrodes    = 'highlights';
cfgPLOT.statstype     = 1;              % 1: corrected; 2:uncorrected; 3: 1-pval;
    4:rawdistance
cfgPLOT.alpha         = 0.05;
cfgPLOT.arrowpos      = 1;
cfgPLOT.showlabels    = 'yes';
cfgPLOT.electrodes    = 'on';
cfgPLOT.hlmarker      = 'o';
cfgPLOT.hlcolor       = [0 0 0];
cfgPLOT.hlmarkersize  = 4;
cfgPLOT.arrowcolorpos = [1 0 0];

figure;
TEplot2D(cfgPLOT,TGA_results_analyzed_GA)
```

5.3 Example group analysis

The following analysis scripts show a comparisons of TE values between two groups of data (e.g. from two pools of subjects or from two conditions recorded in different subjects) and a comparison of TE values estimated under two conditions in a single subject.

Listing 5: Example script for group analysis in TRENTOOL.

```
%% set paths

TRENTOOLpath = './TRENTOOL3';
addpath(TRENTOOLpath);
addpath('./fieldtrip-20150928');
ft_defaults;

%% get data

% find relevant mat files and collect file names in a cell structure
files    = dir('subj_*.mat');
fileCell = {files(:).name};

%% create cfg (as you would for single subject analysis)

cfgTEGP                  = [];
cfgTEGP.TEcalctype       = 'VW_ds';

% get channel label from file, analyze all possible combinations
load(files(1).name);
cfgTEP.channel             = data.label;

% specify u to be scanned
cfgTEGP.predicttimemin_u    = 5;
cfgTEGP.predicttimemax_u    = 17;
cfgTEGP.predicttimestepsize = 2;

% use the whole trial for analysis
cfgTEP.toi = [min(data.time{1,1}),max(data.time{1,1})]; % time of interest

% ACT estimation and constraints on allowed ACT(autocorelation time)
```

```
cfgTEGP.maxlag      = 3*cfgTEGP.predicttimemax_u;   % range for ACT computation
cfgTEGP.actthrvalue = 40;                           % treshold for ACT
cfgTEGP.minnrtrials = 12;                           % minimum acceptable number of
     trials

% optimizing embedding
cfgTEGP.optimizemethod ='ragwitz';   % criterion used
cfgTEGP.ragdim         = 4:8;        % criterion dimension
cfgTEGP.ragtaurange    = [0.2 0.4];  % range for tau [0.5 1]
cfgTEGP.ragtausteps    = 15;         % steps for ragwitz tau steps
cfgTEGP.repPred        = 100;        % points used for optimization the embedding
    dimensions

% kernel-based TE estimation
cfgTEGP.flagNei = 'Mass' ;           % neigbour analyse type
cfgTEGP.sizeNei = 4;                 % neigbours to analyse

cfgTEGP.ensemblemethod  = 'no';
cfgTEGP.outputpath  = '/results/';

%% run TEgroup_prepare

TEgroup_prepare(cfgTEGP,fileCell);

%% run analysis on group prepared data

cfgTESS               = [];
cfgTESS.shifttest     = 'yes';
cfgTESS.optdimusage   = 'indivdim';   % dimension to use
cfgTESS.surrogatetype = 'trialshuffling';

% go to output path of previous analysis step (this is where TEgroup_prepare saved
     the prepared data)
cd(cfgTEGP.outputpath)

% load and analyze prepared data individually
for i=1:length(fileCell)

    load(fileCell(i).name)
    cfgTESS.fileidout = strcat(cfgTEGP.outputpath,fileCell(i).name(1:5));

    TEresult = InteractionDelayReconstruction_calculate(cfgTEGP,cfgTESS,data);

    save([cfgTEGP.outputpath fileCell(i).name(1:end-4) '_calc.mat'],'TEresult');
    clear TEresult data;

end

%% perform graph analysis (optional)

files = dir('*_calc.mat');


% settings for graph analysis
cfgGA = [];
cfgGA.threshold = 1;
cfgGA.cmc       = 1;

for i=1:length(files)

    load(files(i).name);

```

```
    % correct for spurious information transfer
    TEresult_GA = TEgraphanalysis(cfgGA,TEresult);

    save([cfgTEGP.outputpath files(i).name(1:end-9) '_result.mat'],'TEresult_GA');

end


%% run group statistics

% collect individually analyzed data in a file cell
cd(cfgTEGP.outputpath);
fileCell_TEpermtest = dir('*_result.mat');
fileCell_TEpermtest = {fileCell_TEpermtest(:).name};

% define analysis parameters
cfgGSTAT = [];
cfgGSTAT.design     = [1 2 3 4 5 6 7 8; 1 1 1 1 2 2 2 2];
cfgGSTAT.uvar       = 1;
cfgGSTAT.ivar       = 2;
cfgGSTAT.permstatstype = 'indepsamplesT';
cfgGSTAT.fileidout = 'test_groupstats';

% run group statistics
TEgroup_stats(cfgGSTAT,fileCell_TEpermtest);
```

Listing 6: Example script for the comparison of two conditions within a single subject in TRENTOOL.

```
%% set paths

TRENTOOLpath = './TRENTOOL3';
addpath(TRENTOOLpath);
addpath('./fieldtrip-20150928');
ft_defaults;

%% get data

% find relevant mat files and collect file names in a cell structure
fileCell = {'subj_A_cond_1.mat' 'subj_A_cond_2.mat'};

%% create cfg (as you would for single subject analysis)

cfgTEGP                  = [];
cfgTEGP.TEcalctype       = 'VW_ds';

% get channel label from file, analyze all possible combinations
load(files(1).name);
cfgTEP.channel             = data.label;

% specify u to be scanned
cfgTEGP.predicttimemin_u    = 5;
cfgTEGP.predicttimemax_u    = 17;
cfgTEGP.predicttimestepsize = 2;

% use the whole trial for analysis
cfgTEP.toi = [min(data.time{1,1}),max(data.time{1,1})]; % time of interest

% ACT estimation and constraints on allowed ACT(autocorelation time)
cfgTEGP.maxlag      = 3*cfgTEGP.predicttimemax_u;   % range for ACT computation
cfgTEGP.actthrvalue = 40;                           % treshold for ACT
```

```
cfgTEGP.minnrtrials = 12;                          % minimum acceptable number of
    trials

% optimizing embedding
cfgTEGP.optimizemethod ='ragwitz';   % criterion used
cfgTEGP.ragdim         = 4:8;        % criterion dimension
cfgTEGP.ragtaurange    = [0.2 0.4];  % range for tau [0.5 1]
cfgTEGP.ragtausteps    = 15;         % steps for ragwitz tau steps
cfgTEGP.repPred        = 100;        % points used for optimization the embedding
    dimensions

% kernel-based TE estimation
cfgTEGP.flagNei = 'Mass' ;           % neigbour analyse type
cfgTEGP.sizeNei = 4;                 % neigbours to analyse

cfgTEGP.ensemblemethod  = 'no';
cfgTEGP.outputpath  = '/results/';

%% run TEgroup_prepare

TEgroup_prepare(cfgTEGP,fileCell);

%% run analysis on group prepared data

cfgTESS               = [];
cfgTESS.shifttest     = 'yes';
cfgTESS.optdimusage   = 'indivdim';   % dimension to use
cfgTESS.surrogatetype = 'trialshuffling';

% go to output path of previous analysis step (this is where TEgroup_prepare saved
    the prepared data)
cd(cfgTEGP.outputpath)

% load and analyze prepared data individually
for i=1:length(fileCell)

    load(fileCell(i).name)
    cfgTESS.fileidout = strcat(cfgTEGP.outputpath,fileCell(i).name(1:end-4));

    TEresult = InteractionDelayReconstruction_calculate(cfgTEGP,cfgTESS,data);

    save([cfgTEGP.outputpath fileCell(i).name(1:end-4) '_calc.mat'],'TEresult');
    clear TEresult data;

end

%% perform graph analysis (optional)

files = dir('*_calc.mat');


% settings for graph analysis
cfgGA = [];
cfgGA.threshold = 1;
cfgGA.cmc       = 1;

for i=1:length(files)

    load(files(i).name);

    % correct for spurious information transfer
    TEresult_GA = TEgraphanalysis(cfgGA,TEresult);
```

```
92
93      save([cfgTEGP.outputpath files(i).name(1:end-9) '_result.mat'],'TEresult_GA');
94
95  end
96
97
98  %% run compare TE values from both conditions
99
100 % collect individually analyzed data in a file cell
101 cd(cfgTEGP.outputpath);
102 files_TEpermtest = dir('*_result.mat');
103 cond1 = load(files_TEpermtest(1).name);
104 cond2 = load(files_TEpermtest(2).name);
105
106
107
108 % define analysis parameters
109 cfgCSTAT = [];
110 cfgCSTAT.permstatstype = 'indepsamplesT';
111 cfgCSTAT.alpha         = 0.05;
112 cfgCSTAT.tail          = 1;
113 cfgCSTAT.fileidout = 'test_condstatssingle';
114
115 % run statistics
116 TEgroup_conditionstatssingle(cfgCSTAT, cond1.TEpermtest, cond2.TEpermtest);
```

6 Appendix

6.1 Choosing the number of permutations for permutations testing

A number of permutations for statistical testing is required as a configuration parameter `.numpermutation` for a series of TRENTOOL functions (e.g. `TEsurrogatestats.m`, `TEsurrogatestats_ensemble.m`, `TEgroup_stats.m`). The number of permutations specifies how often two data sets are pooled and two new sets are drawn to calculate the test statistic and to then form the Null distribution. The `.numpermutation` parameter can be set by the user or it can be omitted, in which case it is calculated automatically on the basis of the requested alpha level and the number of data sets. This calculation is done by `\TEchecknumperm.m` and works as follows: The required number of permutations is chosen such that the minimum p-value that can be possibly calculated from the Null distribution is smaller than the alpha level (i.e. it is theoretically possible to obtain significant results). The minimum possible p-value is $1/numpermutation$, thus `.numpermutation` is chosen such that the following equations holds:

$$\frac{1}{numpermutation} < \frac{1}{\alpha/c},$$

where α is the requested alpha level of the statistical test, corrected for the number of multiple comparisons c. To be on the safe side, TRENTOOL multiplies the number of strictly necessary permutations by a factor 20.

In a second step, `.numpermutation` is checked for sanity. This check is done for values set by the user and for values calculated by `\TEchecknumperm.m`. It is checked, whether the number of permutations requested/required by the (corrected) alpha level, is theoretically possible given the number of data sets to be tested. For two data sets, with N_1 and N_2 observations in each group, the number of possible permutations is

$$\binom{N_1 + N_2}{min(N_1, N_2)},$$

for an independent test (determined by `.permstatstype`), and

$$2^{N_1} = 2^{N_2}$$

for an dependent test (`.permstatstype = 'depsamplesT'`).

The function checks if the number of possible permutations is larger than the number of requested permutations. If this is not the case, a warning is issued.

6.2 Setting output verbosity

The verbosity of console output can be set with the configuration parameter `cfgTEP.verbosity` (input to `TEprepare.m`). Each console output in TRENTOOL is associated with a level of detail. A output is only printed to the console if it has the same or a lesser level of detail as requested in `cfgTEP.verbosity`. See table 3 for possible settings.

Note that errors and warnings are issued even if console output is switched off (`'none'`).

Table 3
Output verbosity levels defined by `cfgTEP.verbosity`.

string	**description**
`'none'`	no output except for warnings and errors
`'info_major'`	notifications of major analysis steps (e.g. data preparation, reconstruction of interaction delays)
`'info_minor'`	notification of minor analysis steps (e.g. results of data preparation, generation of permutations), **default**
`'debug_coarse'`	debug information, includes technical details and intermediate results
`'debug_fine'`	fine debug information, includes technical details and detailed intermediate results

6.3 TRENTOOL Input/Output

The following tables provide detailed information on input and output of the most common TRENTOOL functions. Input parameters are mandatory if no default values are provided. Curly brackets indicate MATLAB cell arrays, square brackets indicate numeric arrays. Note, that MATLAB uses row-major order for multidimensional arrays.

Table 4
Prameters for the configuration structure `cfgTEP.` *of the functions* `TEprepare.m`, `TEgroup_prepare.m` *and* `InteractionDelayReconstruction_calculate` *(TRENTOOL Version 3.3)*

field name	default value	data type	units	description
`sgncmb`		string		cell array of channel pairs to be analyzed
`channel`		string		cell array of channel names, from which TRENTOOL will construct all possible pairs for later analysis
`toi`		double	seconds	1x2-vector with first and last time point of the time range of interest
`TEcalctype`	`'VW_ds'`	string		`'VW_ds'`: Estimator guaranteeing optimal self prediction (eq. 3, see (Wibral et al., 2013)). `'V'` and `'VW'` estimators are no longer supported
`predictiontime_u`		integer	milli-seconds	assumed information transfer delay u between source and target time series (Fig. 1)
`ensemblemethod`	`'no'`	string		Use of the ensemble-method for (time-resolved) TE estimation, see sec. 4.4 (Wollstadt et al., 2013) (`'yes'` or `'no'`).
`kth_neighbors`	`4`	integer		number of neighbors for fixed mass search (controls balance of bias/statistical errors)
`TheilerT`	`'ACT'`	integer or string 'ACT'		number of temporal neighbors excluded to avoid serial correlations (Theiler correction)
`maxlag`	`1000`	integer	samples	the range of lags for computing the ACT: from -MAXLAG to MAXLAG
`optimizemethod`		string		define method for parameter optimization: 'ragwitz'
`trialselect`	`'ACT'`	string		selecting trials: 'no' uses all available trials, 'range' uses trials from a separately defined range (fields `.trial_from` and `trial_to`), 'ACT' uses trials with an ACT lower than a given threshold (field `.actthreshold_value`)
`actthrvalue`		integer		max threshold for the ACT for trial selection
`trial_from`		integer		first trial in case of range selection of trials
`trial_to`		integer		last trial in case of range selection of trials
`minnrtrials`		integer		sets a minimum number of trials, that have to survive trial selection (if `cfgTEP.trialselect = 'ACT'` is set); if less trials survive trial selection, TE estimation is abortet
`verbosity`	`'info_minor'`	string		defines the verbosity of console output of TRENTOOL (see section 6.2)
Parameters needed for the use of Ragwitz' criterion for parameter optimization				
`ragdim`	`1 to 10`	integer		for Ragwitz: range of embedding dimensions to scan vector from 1 to n
`ragtaurange`		double	units of act	for Ragwitz: 1x2-vector of min and max embedding delays
`ragtausteps`	`10`	integer		for Ragwitz: number of equidistant steps in ragtaurange with a minimum of 5
`flagNei`		string		for Ragwitz: 'Range' or 'Mass' type of neighbor search
`sizeNei`		integer		for Ragwitz: Radius or mass for the neighbor search according to `flagNei`
`repPred`		integer		for Ragwitz: repPred represents the number of sample points for which the prediction is performed (it has to be smaller than $length(timeSeries) - (embeddingdimension - 1) * tau * ACT - u$)

Table 5
Results provided in the field `.TEprepare`*, added to the data by* `TEprepare.m` *(TRENTOOL Version 2.0)*

field name	dimension	data type	units	description
`channelcombi`	[no. channel combinations x 2]	integers		cell array with channel indices specifying the channel combinations to be analyzed
`channelcombilabel`	{no. channel combinations x 2 }	strings		cell array with channel labels specifying the channel combinations to be analyzed
`ACT`	[no. channel combinations x 2 x no. trials]	integers	samples	array with the values of the auto correlation decay times of the channelcombinations
`trials`	{no. channel combinations x 2} [1 x no. trials]	integers		cell array of integer arrays containing trials used for individual channel combinations for analysis
`ntrials`	[no. channel combinations x 2]	integers		integer array containing the number of trials used for individual channel combinations for analysis
`optdimmattrial`	[channelcombi x trial]	integer		array with optimal embedding dimension for each trial
`optdimmat`	[channelcombi x 1]	integer		array with optimal embedding dimension for each channel combination (over trials)
`optdim`	scalar	integer		max of the `optdimmat` which should be used as embedding dimension in the further steps
`timeindices`	[1 x 2]	integer	samples	timeindices in samples (from `cfg.toi`)
`u_in_samples`	scalar	integer	samples	interaction delay in sample points (from `cfg.predictionstime_u`)
`cfg`		structure		configuration structure `cfgTEP` (table 4) provided by the user
`maxact`	scalar	integer		maximum autocorrelation decay time of the targetchannels
`opttaumat`	[1 x channelcombi]	integer		optimal embedding delays tau for each channel combination over trials
`opttau`	scalar	integer		max of the opttaumat which should be used as embedding delay in the further steps

Table 6
Parameters for the configuration structure `cfgTESS` of the functions `TEsurrogatestats.m`, `TEsurrogatestats_ensemble.m` and `InteractionDelayReconstruction_calculate.m` (TRENTOOL Version 3.3)

field name	default value	data type	description
`optdimusage`		string	'maxdim' to use maximum of optimal embedding dimensions over all channels for all channels, or 'indivdim' to use the individual optimal dimension for each channel. In case of using ragwitz criterion also the optimal embedding delay tau per channelcombi is used.
`dim`	output TEprepare	integer	Value(s) for embedding dimension. This is automatically taken from the field `TEprepare` in the data (recommended!). Otherwise: scalar value (if `cfgTESS.optdimusage = 'maxdim'`) or vector of size [channelcombi x 1] (if `cfgTESS.optdimusage = 'indivdim'`).
`tau`	output TEprepare	integer	Embedding delay in units of act (x*act). This is automatically taken from the field `TEprepare` in the data (recommended!). Otherwise: scalar value (if `cfgTEP.optdimusage = 'maxdim'`) or vector of size [channelcombi x 1] (if `cfgTEP.optdimusage = 'indivdim'`). Otherwise:
`alpha`	0.05	double	Significance level for statisatical permutation test
`surrogatetype`		string	Strategy for surrogate data creation: 'trialshuffling', 'trialreverse', 'blockresampling', 'blockreverse1','blockreverse2', or 'blockreverse3' (see section 4.5).
`extracond`		string	Perform conditioning in tansfer entropy formula on additional variables. Values: `'Faes_Method'` ((Faes et al., 2013), see section 4.5)
`MIcalc`	1	integer	Determines whether mutual information is calculated additionally to TE (1) or not (0).
`shifttest`	'yes'	string	Perform shift test to identify instantaneous mixing between the signal pairs. Values: 'yes' or 'no'. Note: If `cfgTESS.extracond = 'Faes_Method'` is requested, `cfgT_ESS.shifttest` has to be set to 'no' as both methods are mutually exclusive (see section 4.5).
`shifttesttype`	'TE>TEshift'	string	The shift test can be calculated for the direction TE value of original data > TE values of shifted data (value = 'TE>TEshift') or for the other direction (value = 'TEshift>TE'). In this case the alpha is set to 0.1.
`shifttype`	'predicttime'	string	Shifting the data 'onesample' or the length of the 'predicttime'.
`numpermutation`	190100/500[a]	integer	Nr of permutations in permutation test.
`permstatstype`	'indepsamplesT	string	Type of the test statistic used: 'mean' to use the distribution of the mean differences, 'normmean' to use the distribution of the normalized mean differences and 'depsamplesT' or 'indepsamplesT' for distribution of the t-values.
`tail`	1	integer	1 tail or 2 tailed test of significance (for the permutation tests).
`correctm`	'FDR'	string	Correction method used for correction of the multiple comparison problem over all analyzed channel combinations - False discovery rate 'FDR' or Bonferroni correction 'BONF'.
`fileidout`		string	String for the first part of the output filename.
Additional hardware parameters needed when using the ensemble method for TE estimation			
`GPUmemsize`		integer	The memory of the GPU in MB (e.g. `cfg.GPUmemsize = 4200;`).
`numthreads`		integer	Max. number of threads that can be run in one block on the GPU.
`maxgriddim`		integer	Max. grid dimension of the GPU.

[a] The first value is the default for trial-wise TE estimation on the CPU, the second value is the default for ensemble TE estimation using a GPU.

Table 7
Results provided in the structure `TEpermtest` saved to disk/returned by 'TEsurrogatestats.m' and 'TEsurrogatestats_ensemble.m', (TRENTOOL Version 3.3)

field name	dimension	data type	description
`dimord`		string	dimensions of TEpermvalues
`cfg`		structure	configuration file used to TE estimation (`cfgTESS`, see table 6)
`sgncmb`		cell array of strings	labels of channel combinations (source -> target)
`numpermutation`	scalar	integer	number of permutations used for statistical testing
`ACT`		structure	structure including `.act` with the ACT matrix ([channelcombi x 2 x trial])
`nr2cmc`	scalar	integer	number of of multiple comparisons that was corrected for
`TEprepare`		structure	results of the function TEprepare from the input data
`TEpermvalues`	[no. channel combinations x 5(6[a])]	doubles	matrix with results of permutation testing for individual channel combinations. For each channel combination (i.e. each row), the matrix holds the following information:

	Column	Description	Values
	1	p-value	0 to 1
	2	stat. significance	1 significant at the provided alpha level, 0 not significant
	3	stat. significance after CMC[b]	1 significant at the provided alpha level after CMC, 0 not significant after CMC
	4	mean difference	raw difference between empirical and surrogate data
	5	volume conduction	indicates the presence (1) or absense (0) of volume conduction (is set to NaN for ensemble method)
	6[a]	optimal u	u in msec, available only after was ran on data

[a] a sixth column containing the optimal u is added only if interaction delays are reconstructed for the data (using `InteractionDelayReconstruction_analyze.m`, see section 4.3)
[b] correction for multiple comparisons

Table 8
Results provided in the structure `TEresult` *saved to disk by 'TEsurrogatestats.m' and 'TEsurrogatestats_ensemble.m' (TRENTOOL Version 3.3)*

field name	dimension	data type	description
`TEmat`	[no. channel combinations x u x trial]	double	matrix of raw TE values
	[no. channel combinations]	double	vector of raw TE values if the ensemble method is used for TE estimation
`MImat`	[no. channel combinations x u x trial]	double	matrix of rax mutual information values
	[no. channel combinations]	double	vector of raw MI values if the ensemble method is used for TE estimation
`dimord`		string	dimension of TEmat and MImat
`cfg`		structure	configuration structure used for TE estimation (`cfgTESS`, see table 6)
`trials`			trial numbers selected from raw data set
`act`	[no. channel combinations x 2 x no. trials]	integer	ACT matrix
`sgncmb`		structure	results of the function TEprepare from the input data

Table 9
Fields in the structure `.groupprepare` *added to data prepared from group analysis by the function* `TEgroup_prepare.m` *(TRENTOOL Version 3.3)*

field name	description
`max_dim`	Maximum embedding dimension over all subjects, channels and u. This will be used for TE estimation in individual subjects.
`min_nrtrials`	Minimum number of trials over all subjects.
`code`	Code added to all subjects. This field is checked by `TEgroup_stats.m` to make sure, all data sets were prepared within the same run of `TEgroup_prepare.m`
`predicttimevec_u`	Vector with assumed interaction delays u. This will be used for TE estimation to overwrite any conflicting information.

Table 10
Parameters for the configuration structure `cfgGSTAT` of the function `TEgroup_stats.m` (TRENTOOL Version 3.3)

field name	default value	data type	description
`design`		integer array	design matrix with dimension 2 by no. data sets, where one row contains the data set number and one row encodes the independent variable (see example in table 2)
`uvar`		integer	row in `cfgGSTAT.design` that contains the data set number (unit variable)
`ivar`		integer	row in `cfgGSTAT.design` that contains the independent variable (encoding of group or condition)
`alpha`	0.05	double	significance level for statistical permutation test
`numpermutation`	190100	integer	number of permutations in permutation test
`permstatstype`	mean	string	statistic for permutation test: 'mean' usse the distribution of the differences between group means; 'depsamplesT' or 'indepsamplesT' uses the distribution of the (dependent or independent) t-values computed from both group means
`tail`	2	integer	one- or two tailed test for significance (1 or 2)
`correctm`	'FDR'	string	correction method used for correction of the multiple comparison problem - False discovery rate 'FDR' or Bonferroni correction 'BONF'
`fileidout`		string	output path and the first part of the output filename

Table 11
Parameters for the configuration structure `cfgCSTAT` of the function `TEgroup_condstatssingle.m` (TRENTOOL Version 3.3)

field name	default value	data type	description
`alpha`	0.05	double	significance level for statistical permutation test
`numpermutation`	190100	integer	number of permutations in permutation test
`permstatstype`	mean	string	statistic for permutation test: 'mean' usse the distribution of the differences between group means; 'depsamplesT' or 'indepsamplesT' uses the distribution of the (dependent or independent) t-values computed from both group means
`tail`	2	integer	one- or two tailed test for significance (1 or 2)
`correctm`	'FDR'	string	correction method used for correction of the multiple comparison problem - False discovery rate 'FDR' or Bonferroni correction 'BONF'
`fileidout`		string	output path and the first part of the output filename

Table 12
Parameters for the configuration structure `cfgGA` of the functions `TEgraphanalysis.m` (TRENTOOL Version 3.3)

field name	data type	units	description
`threshold`	integer	milli-seconds	tolerance that is used when looking for alternative paths
`cmc`	integer	bool	consider links significant after correction for multiple comparisons (1) or links significant without correction (0)

Table 13
Output of `TEgraphanalysis.m` (TRENTOOL Version 3.3): The table TEpermvalues is modified for links that are identified as potentially spurious; a substructure `graphanalysis` is added to the results structure.

Modifications in `TEpermvalues`:

Column		Set to
1	p-value	1
2	significance at the prescribed alpha	0
3	significance after correction for multiple comparisons	0
4	mean difference	NaN
5	link type	type of spurious interaction: 2 = cascade effect; 3 = cascade effect triangle; 4 = common drive link triangle.
6	interaction delay	0

Fields in `graphanalysis`:

field name	description
`n_vertices`	number of vertices
`n_edges`	number of edges
`density`	graph density, defined as $\frac{E}{(V*(V-1))}$, where V = `n_vertices` and E = `n_edges`
`threshold`	user provided threshold (see table 12)
`triangles`	list of triangles found in the graph (1x3-vector with indices of the three edges)

Table 14
Parameters for the configuration structure `cfgUPLOT` *of the function* `InteractionDelayReconstruction_plot.m` *(TRENTOOL Version 3.3). All parameters have to be given as strings.*

field name	default value	description
`standardize`	`'no'`	Plot standardized TE values (`'yes'`/`'no'`) .
`scaletype`	`'lin'`	Use log (`'log'`) or linear (`'lin'`) scaling.
`ch_per_fig`	8	Number of channels to be plotted per Figure (to avoid cluttered plots if a large number of channel combis is plotted).

Table 15
Parameters for the configuration structure `cfg2DPLOT` *of the function* `TEplot2D.m` *(TRENTOOL Version 3.3).*

field name	default value	data type	description
`statstype`		integer	Statistics used for plotting (determines which information transfer arrows are plotted), 1 = corrected; 2 = uncorrected; 3 = 1-pval; 4 = rawdistance.
`arrowpos`	2	integer	Position of arrowhead: 1 = end of the line; 2 = centre of the line.
`arrowcolorpos`	[1 0 0]	integer vector [1x3]	Arrow color in case of positive mean distances between raw TE value and surrogate distribution (MATLAB rgb color-code, values have to be between 0 and 1).
`arrowcolorneg`	[0 0 1]	integer vector [1x3]	Arrow color in case of positive mean distances between raw TE value and surrogate distribution (MATLAB rgb color-code, values have to be between 0 and 1).
`alinewidth`	2	integer	Linewidth of arrows in case of significance.
`electrodes`	`'on'`	string	plotting of electrodes, 'on', 'off', 'labels', 'numbers', 'highlights'.
`hcolor`	[0 0 1]	integer vector [1x3]	Color of head cartoon.
`hlinewidth`	2	integer	Linewidth of the head cartoon, nose and ears.
`emarker`	'o'	string (MATLAB marker symbol)	Marker symbol.
`ecolor`	[0 0 0]	integer vector [1x3]	Marker color (MATLAB rgb color-code, values have to be between 0 and 1).
`emarkersize`	2	integer	Marker size.
`efontsize`	8	integer (pt)	Font size of electrode labels/numbers (only if `cfg2DPLOT.electrodes = 'numbers'` or `'labels'`).
`efontcolor`	[0 0 0]	integer vector [1x3]	Font color of electrode labels/numbers (only if `cfg2DPLOT.electrodes = 'num_bers'` or `'labels'`, MATLAB rgb color-code, values have to be between 0 and 1).
`hlmarker`	'o'	string (MATLAB marker symbol)	Highlight marker symbol.
`hlcolor`	[1 0 0]	integer vector [1x3]	Highlight marker color (MATLAB rgb color-code, values have to be between 0 and 1).
`hlmarkersize`	4	integer	Highlight marker size.
`layout`		Specification of the layout which defines how the channels are arranged, may be one of two options:	
		layout structure	Pre-computed layout structure (see FieldTrip reference for `prepare_layout`).
		string	Name of an ascii layout file with extension `*.lay`.

Table 16
Parameters for the configuration structure `cfgBN` *of the function* `TRENTOOL2BrainNet.m` *(TRENTOOL Version 3.3).*

field name	data type	description
`MNIcoord`	integer array [nx3]	MNI coordinates (x,y,z) of n sources or channels.
`labels`	cell array [nx1]	Cell array containing strings with source/channel labels. **NOTE**: Labels should not contain blanks, this causes BrainNet Viewer to chrash. Also, make sure to use the same ordering of labels for the fields `.labels` and `.MNIcoord` and the TEpermvalues table in `TEpermtest` (you may use the label information in `TEpermtest.cfg.channel`).
`filename`	string	filename for output files `*.node` and `*.edge`, you may specify a filename only (string), which causes the function to save both files to the current folder, or you can specify a whole filepath ([filepath]/[filename]).
`nodeCol`	integer vector [nx1]	Optional. BrainNet gives you the option to color nodes according to the values in this vector, see BrainNet Manual. If you want the nodes to have the same color, provide a vector with ones or no vector at all.
`nodeSize`	integer vector [nx1]	Optional. BrainNet gives you the option to size nodes according to the values in this vector, see BrainNet Manual. If you want the nodes to have the same size, provide a vector with ones or no vector at all.
`edgeCol`	integer vector [nx1]	Optional. BrainNet gives you the option to color edges according to the values in this vector, see BrainNet Manual. If you want the edges to have the same color, provide a vector with ones or no vector at all.

References

Ay, N., & Polani, D. (2008). Information flows in causal networks. *Advances in Complex Systems*, *11*(1), 17–42.

Blinowska, K., Kuś, R., & Kamiński, M. (2004). Granger causality and information flow in multivariate processes. *Physical Review E*, *70*(5), 050902.

Chicharro, D., & Andrzejak, R. G. (2009, Aug). Reliable detection of directional couplings using rank statistics. *Phys Rev E Stat Nonlin Soft Matter Phys*, *80*(2 Pt 2), 026217.

Faes, L., Nollo, G., & Porta, A. (2013). Compensated transfer entropy as a tool for reliably estimating information transfer in physiological time series. *Entropy*, *15*(1), 198–219.

Gomez-Herrero, G., Wu, W., Rutanen, K., Soriano, M., Pipa, G., & Vicente, R. (2010). Assessing coupling dynamics from an ensemble of time series. *arXiv preprint arXiv:1008.0539*.

Kamiński, M., Ding, M., Truccolo, W., & Bressler, S. (2001). Evaluating causal relations in neural systems: Granger causality, directed transfer function and statistical assessment of significance. *Biological cybernetics*, *85*(2), 145–157.

Kozachenko, L., & Leonenko, N. (1987). Sample estimate of entropy of a random vector. *Probl. Inform. Transm.*, *23*, 95-100.

Kraskov, A., Stögbauer, H., & Grassberger, P. (2004, Jun). Estimating mutual information. *Phys Rev E Stat Nonlin Soft Matter Phys*, *69*(6 Pt 2), 066138.

Lindner, M., Vicente, R., Priesemann, V., & Wibral, M. (2011, Nov). TRENTOOL: A matlab open source toolbox to analyse information flow in time series data with transfer entropy. *BMC Neuroscience*, *12*(1), 119.

Lizier, J., & Prokopenko, M. (2010). Differentiating information transfer and causal effect. *Eur. Phys. J. B*, *73*, 605–615.

Lizier, J. T., & Rubinov, M. (2013). Multivariate construction of effective computational networks from observational data.

Maris, E., & Oostenveld, R. (2007). Nonparametric statistical testing of EEG-and MEG-data. *Journal of Neuroscience Methods*, *164*(1), 177-190. doi: 10.1016/j.jneumeth.2007.03.024

NVIDIA Corporation. (2013). *CUDA toolkit documentation.* Available: http://docs.nvidia.com/cuda. Accessed 7 November 2013.

Oostenveld, R., Fries, P., Maris, E., & Schoffelen, J.-M. (2011). Fieldtrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data. *Computational Intelligence and Neuroscience*, *2011*, 1–9. doi: 10.1155/2011/156869

Ragwitz, M., & Kantz, H. (2002, Apr). Markov models from data by simple nonlinear time series predictors in delay embedding spaces. *Phys. Rev. E*, *65*, 056201. Retrieved from `http://link.aps.org/doi/10.1103/PhysRevE.65.056201` doi: 10.1103/PhysRevE.65.056201

Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, *85*(2), 461–464.

Takens, F. (1981). Detecting strange attractors in turbulence. *Dynamical systems and turbulence, Warwick 1980*, 366–381.

Vicente, R., Wibral, M., Lindner, M., & Pipa, G. (2011). Transfer entropy - A model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, *30*(1), 45–67.

Victor, J. (2002). Binless strategies for estimation of information from neural data. *Phys Rev E Stat Nonlin Soft Matter Phys*, *72*, 051903.

Wibral, M., Pampu, N., Priesemann, V., Siebenhühner, F., Seiwert, H., Lindner, M., ... Vicente, R. (2013). Measuring information-transfer delays. *PloS one*, *8*(2), e55809.

Wibral, M., Rahm, B., Rieder, M., Lindner, M., Vicente, R., & Kaiser, J. (2011, Mar). Transfer entropy in magnetoencephalographic data: Quantifying information flow in cortical and cerebellar networks. *Progress in Biophysics and Molecular Biology*, *105*(1-2), 80–97.

Wibral, M., Wollstadt, P., Meyer, U., Pampu, N., Priesemann, V., & Vicente, R. (2012). Revisiting wiener's principle of causality – interaction-delay reconstruction using transfer entropy and multivariate analysis on delay-weighted graphs. *Conf Proc IEEE Eng Med Biol Soc*, *2012*, 3676–3679. doi: 10.1109/EMBC.2012.6346764

Wiener, N. (1956). *The theory of predictions.* Modern mathematics for engineers (1956) Beckenbach, Edwin F. NewYork: McGraw-Hill.

Wollstadt, P., Martinez-Zarzuela, M., Vicente, R., Diaz-Pernas, F. J., & Wibral, M. (2013). Efficient transfer entropy analysis of non-stationary neural time series. *PloS one*, *submitted*.

Xia, M., Wang, J., & He, Y. (2013). Brainnet viewer: a network visualization tool for human brain connectomics. *PloS one*, *8*(7), e68910.