

# Statistical Learning based Estimation of the Mutual Information (SLEMI) - R package

## User Manual

*T. Jetka, K. Nienaltowski, T. Winarski, S. Błoński, M. Komorowski*

*Version: May 6, 2019*

## Contents

<b>1</b>	<b>Preliminaries</b>	<b>2</b>
1.1	Requirements - Hardware . . . . .	2
1.2	Requirements - Software . . . . .	2
1.3	Installation . . . . .	3
<b>2</b>	<b>Structure of the package</b>	<b>4</b>
2.1	Input data . . . . .	6
2.2	Calculation of the information capacity . . . . .	6
2.3	Calculation of the mutual information . . . . .	7
2.4	Calculation of the probabilities of correct discrimination . . . . .	8
<b>3</b>	<b>Diagnostic procedures</b>	<b>9</b>
<b>4</b>	<b>Additional functionalities of the function <code>capacity_logreg_main()</code></b>	<b>9</b>
<b>5</b>	<b>Examples</b>	<b>10</b>
5.1	Minimal example . . . . .	10
5.2	Further step-by-step introductory examples . . . . .	15
5.3	Examples in MP and SI . . . . .	15
<b>6</b>	<b>List of all package's functions</b>	<b>15</b>

# 1 Preliminaries

## 1.1 Requirements - Hardware

- A 32 or 64 bit processor (recommended: 64bit)
- 1GHz processor (recommended: multicore for a comprehensive analysis)
- 2GB MB RAM (recommended: 4GB+, depends on the size of experimental data)

## 1.2 Requirements - Software

The main software requirement is the installation of the R environment (version:  $\geq 3.2$ ), which can be downloaded from R project website and is distributed for all common operating systems. We tested the package in R environment installed on Windows 7, 10; Mac OS X 10.11 - 10.13 and Ubuntu 18.04 with no significant differences in the performance. The use of a dedicated Integrated development environment (IDE), e.g. RStudio is recommended.

Apart from a base installation of R, SLEMI requires the following R packages:

1. for installation
  - devtools
2. for estimation
  - e1071
  - Hmisc
  - nnet
  - glmnet
  - caret
  - doParallel (if parallel computation are needed)
3. for visualisation
  - ggplot2
  - ggthemes
  - gridExtra
  - corrplot
4. for data handling
  - reshape2
  - stringr
  - plyr

Each of the above packages can be installed by executing

```
install.packages("name_of_a_package")
```

in the R console.

Importantly, during installation availability of the above packages will be verified and missing packages will be automatically installed.

### 1.3 Installation

The package can be directly installed from GitHub. For installation, open RStudio (or base R) and run following commands in the R console

```
install.packages("devtools") # run if 'devtools' is not installed  
library(devtools)  
install_github("sysbiosig/SLEMI")
```

Are required packages not found, they will be installed automatically.

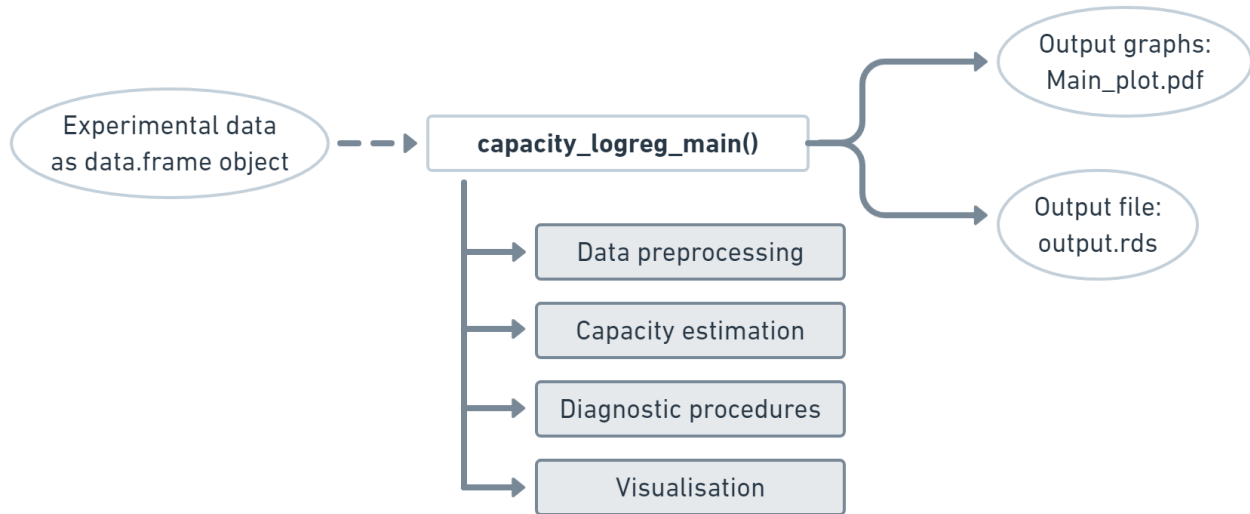


Figure 1: Main function to estimate channel capacity

## 2 Structure of the package

The three functions listed below constitute the key wrapper (interface) functions of the package.

1. `mi_logreg_main()` enables calculation of the mutual information
2. `capacity_logreg_main()` enables calculation of the information capacity
3. `prob_discr_pairwise()` serves to calculate probabilities of correct discrimination between pairs of input values

Below, we outline the architectures of these functions.

**The function `capacity_logreg_main()` triggers**

- i) preprocessing of the data
- ii) estimation of channel capacity
- iii) running diagnostic procedures
- iv) visualisation.

Each of the above steps is implemented within auxiliary functions as presented by the graph below.

The algorithm to compute the information capacity is implemented within the function `capacity_logreg_algorithm()`, which uses logistic regression from the **nnet** package.

Diagnostic procedures (significance and uncertainties of estimates) are provided in an internal function `capacity_logreg_testing()`. These are based on data bootstrapping and overfitting test.

For visualization, a set of graphs is created by an internal function `capacity_output_graphs()` and saved in a specified directory. In addition, `capacity_logreg_main()` returns a list with capacity estimates, optimal input probability distribution, diagnostic measures and other summary information about the analysis.

**The function `mi_logreg_main()`** serves to calculate the mutual information. It initiates similar steps as the function `capacity_logreg_main()` but without performing the optimization of the distribution of the input. Instead, it requires the input distribution to be specified by the user as a function's argument.

Logistic regression and Monte Carlo methods, following an analogous algorithm as within the `capacity_logreg_algorithm()` function, are combined to estimate mutual information within a function

`mi_logreg_algorithm()`. Visualisation and diagnostics are carried out by the same set of auxillary functions as for channel capacity (internal functions `capacity_output_graphs()` and `capacity_logreg_testing()`).

**The `prob_discr_pairwise()`** allows to estimate probabilities of correct discrimination between two different values of the input. It implements estimation of probabilities of correct classification by logistic regression (from `nnet` package) for each pair of input values. The probabilities of correct discrimination are visualized with a graph composed of pie charts.

input	output 1	output 2	output 3	...
$n_1 \left\{ \begin{array}{l} x_1 \\ \vdots \\ x_1 \end{array} \right.$	$y_{1,1}^1$	$y_{1,2}^1$	$y_{1,m}^1$	
	$\vdots$	$\vdots$	$\vdots$	
	$y_{n_1,1}^1$	$y_{n_1,2}^1$	$y_{n_1,m}^1$	
$n_2 \left\{ \begin{array}{l} x_2 \\ \vdots \\ x_2 \end{array} \right.$	$y_{1,1}^2$	$y_{1,2}^2$	$y_{1,m}^2$	
	$\vdots$	$\vdots$	$\vdots$	
	$y_{n_2,1}^2$	$y_{n_2,2}^2$	$y_{n_2,m}^2$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$
$n_m \left\{ \begin{array}{l} x_m \\ \vdots \\ x_m \end{array} \right.$	$y_{1,1}^m$	$y_{1,2}^m$	$y_{1,m}^m$	
	$\vdots$	$\vdots$	$\vdots$	
	$y_{n_m,1}^m$	$y_{n_m,2}^m$	$y_{n_m,m}^m$	

Figure 2: Conceptual representation of a generic experimental dataset needed for quantifying information transmission of a channel

## 2.1 Input data

Functions `mi_logreg_main()`, `capacity_logreg_main()`, `prob_discr_pairwise()` require data in the form of the object `data.frame` with a specific structure of rows and columns. As described in detail in Section 1 of the **SI**, single cell responses  $y_j^i$  are assumed to be measured for a finite set of stimuli levels  $x_1, x_2, \dots, x_m$ . The responses  $y_j^i$  can be multidimensional. Usually, experimental dataset is represented as a table with rows and columns organized as below

Therefore, the input data frame is expected to have the form represented by the above table, which can be formally described by the following conditions

- each row represent a response of a single cell
- first column contains values of the input (X).
- second and subsequent columns contain values of the measured output(s); these columns should be of type `numeric`; order and number of outputs should be the same for all cells.
- the number of unique values of the input should be finite
- a large number of observations, possibly  $>100$ , per input value is required.

An example of the input `data.frame`, which contains the measurements of the NfkB system presented in the **MP** is available within the package under the variable `data_nfkb`. It has the following format

	signal	response_0	response_3	response_6
1	0ng	0.3840744	0.4252835	0.4271986
2	0ng	0.4709216	0.5777821	0.5361948
3	0ng	0.4274474	0.6696011	0.8544916
10001	8ng	0.3120216	0.3475484	1.0925967
10002	8ng	0.2544961	0.6611051	2.2894928
10003	8ng	0.1807391	0.4336810	1.9783171
11540	100ng	1.3534083	3.0158004	5.1592848
11541	100ng	1.7007936	2.2224497	3.5463418
11542	100ng	0.1997087	0.2886905	1.9324093

where each row represents measurements of a single-cell, the column named `signal` specifies the level of stimulation, while `response_T` is the response of the NfkB system in an individual cell at time point T. The above table can be shown in R by calling

```
library(SLEMI)
rbind(data_nfkb[1:3,1:4],data_nfkb[10001:10003,1:4],tail(data_nfkb[,1:4],3))
```

- `cc` - a numeric with channel capacity estimate (in bits)
- `p_opt` - a numeric vector with the optimal input distribution
- `model` - a `nnet` object describing fitted logistic regression model
- `data` - a `data.frame` with the raw experimental data (if `data_out=TRUE`)
- `time` - processing time of the algorithm
- `params` - a vector of parameters used in the algorithm
- `regression` - a confusion matrix of logistic regression predictions

By default, all returned elements are saved in `output_path` directory in a file `output.rds`. Along with the output data, results of the computations are visualised as the graphs listed below

- `MainPlot.pdf` - a simple summary plot with basic distribution visualization and capacity estimate
- `MainPlot_full.pdf` - a comprehensive summary plot with distribution visualization and capacity estimate
- `capacity.pdf` - a diagram presenting the capacity estimates
- `io_relation.pdf` - a graph with input-output relation
- `kdensities.pdf` - kernel density estimator of data distribution
- `histograms.pdf` - histograms of data
- `boxplots.pdf` - boxplots of data
- `violin.pdf` - violin plots of data

## 2.3 Calculation of the mutual information

The function `mi_logreg_main()` takes a similar list of arguments and generates analogous plots to the function `capacity_logreg_main()`. The differences are listed below.

Firstly, user must specify the distribution of input that should be used for calculation of the mutual information. It is done by passing a numeric vector via the argument `pinput` of `mi_logreg_main()` function. Secondly, the returned list stores the value of the computed mutual information (in bits) under the element `mi`.

## 2.4 Calculation of the probabilities of correct discrimination

Calculation of the probabilities of correct discrimination between pairs of input values is performed by running the following command

```
prob_discr_pairwise(dataRaw, signal, response, output_path)
```

where the required arguments are analogous to the arguments of the functions `capacity_logreg_main()` and `mi_logreg_main()`. The probabilities of correct discrimination are computed for each pair of unique input values and returned as a list with the following elements

- `prob_matr` - a symmetric numeric matrix with a probability of discriminating between  $i$ -th and  $j$ -th input values in cell (i,j)
- `diagnostics` - a list of summaries describing fitted logistic regression models of classification between each pair of input values.

In addition, a plot of corresponding pie charts is created in `output_path` in the pdf format.



### 3 Diagnostic procedures

In addition to the sole calculation of the information capacity, the function `capacity_logreg_main()` can also be used to assess accuracy of the channel capacity estimates resulting from potentially insufficient sample size and potential over-fitting of the regression model. Two tests are implemented. Precisely, the function can perform

1. Bootstrap test - capacity is re-calculated using  $\alpha\%$  of data, sampled from the original dataset without replacement. After repeating the procedure  $n$  times, standard deviation of the obtained sample can serve as an error of the capacity estimate.
2. Over-fitting test - the original data is divided into Training and Testing datasets. Then, logistic regression is estimated using  $\alpha\%$  of data (training dataset), and integrals of channel capacity are calculated via Monte Carlo using remaining  $(1 - \alpha)\%$  of data (testing dataset). It is repeated  $n$  times.

In order to perform diagnostic tests, that by default are turned off, user must set the value of the input argument

- `testing = TRUE` (default=FALSE)

In addition, settings of the diagnostic test can be altered by changing the following parameters

- `TestingSeed` (default= 1234) - the seed for the random number generator used to sample original dataset,
- `testing_cores` (default= 4) - a number of cores to use (via `doParallel` package) in parallel computing,
- `boot_num` (default= 40) - a number of repetitions of the bootstrap ,
- `boot_prob` (default= 0.8) - a fraction of initial observations to use in the bootstrap,
- `traintest_num` (default= 40) - a number of repetitions of the overfitting test,
- `partition_trainfrac` (default= 0.6) - a fraction of initial observations to use as a training dataset in the overfitting test

### 4 Additional functionalities of the function `capacity_logreg_main()`

In addition, to the basic functionalities described above, the function `capacity_logreg_main()` allows to control several other parameters of the algorithm that computes the information capacity. These parameters and their effects are listed below.

- `model_out` (default=TRUE) - logical, specify if `nnet` model object should be saved into output file
- `plot_width` (default = 6) - numeric, the basic width of created plots
- `plot_height` (default = 4) - numeric, the basic height of created plots
- `scale` (default = TRUE) - logical, value indicating if the columns of `dataRaw` are to be centered and scaled, what is usually recommended for the purpose of stability of numerical computations. From a purely theoretical perspective, such transformation does not influence the value of channel capacity.
- `lr_maxit` (default = 1000) - a maximum number of iterations of fitting step of logistic regression algorithm in `nnet` function. If a warning regarding lack of convergence of logistic model occurs, should be set to a larger value (possible if data is more complex or of a very high dimension).
- `MaxNWts` (default = 5000) - a maximum number of parameters in logistic regression model. A limit is set to prevent accidental over-loading the memory. It should be set to a larger value in case of exceptionally high dimension of the output data or very high number of input values. In principle, logistic model requires fitting  $(m - 1) \cdot (d + 1)$  parameters, where  $m$  is the number of unique input values and  $d$  is the dimension of the output.

The latter two parameters, i.e `lr_maxit` and `MaxNWts`, allow to change the parameters of the logistic regression model fitting within the dependent `nnet` package.

## 5 Examples

### 5.1 Minimal example

Below, we present a minimal model that may serve as a quick introduction to computations within the package. Precisely, we consider a system

- i) with four different input values  $X$ : 0, 0.1, 1 and 10
- ii) with the conditional output,  $Y|X = x$ , give by a one-dimensional log-normal distribution  $\exp\{\mathcal{N}(10 \cdot \frac{x}{1+x}, 1)\}$
- iii) and the sample consisting of 1000 observations for each input value.

The example is analogous to the Test example 2 of the **SI** (Section 3.2).

#### Input data

Firstly, we generate a synthetic dataset. The data corresponding to the model can be generated, and represented as the data frame `tempdata` with columns `input` and `output`, by running

```
xs=c(0,0.1,1,10) # concentration of input.
tempdata = data.frame(input = factor(c(t(replicate(1000,xs))),
                                   levels=xs),
                      output = c(matrix(rnorm(4000, mean=10*(xs/(1+xs)),sd=c(1,1,1,1)),
                                       ncol=4,byrow=TRUE) ))
```

The generated data.frame has the following structure

	input	output
1	0	-0.2447518
2	0	-0.5217063
2001	1	5.3107607
2002	1	5.2957607
3999	10	7.8274830
4000	10	9.5975094

#### Calculation of the information capacity

The Information capacity can be calculated using the `capacity_logreg_main()` function that takes the data frame “tempdata” as `dataRaw` argument. Column names “input” and “output” are used as arguments `signal` and `response`, respectively. The `output_path` is set as “minimal\_example/”. Therefore, the function is run as follows

```
tempoutput <- capacity_logreg_main(dataRaw=tempdata,
                                   signal="input", response="output",
                                   output_path="minimal_example/")
```

Results of the computations are returned as a data structure described before. In addition, results are presented in the form of the following graph (by default saved as `MainPlot.pdf` in `minimal_example/` directory). It represents the input-output data and gives the corresponding channel capacity.

#### Calculation of the mutual information

To compare mutual information of experimental data with its channel capacity, we can run (uniform distribution of input values is assumed)

```
tempoutput_mi <- mi_logreg_main(dataRaw=tempdata,
                                 signal="input", response="output",
```

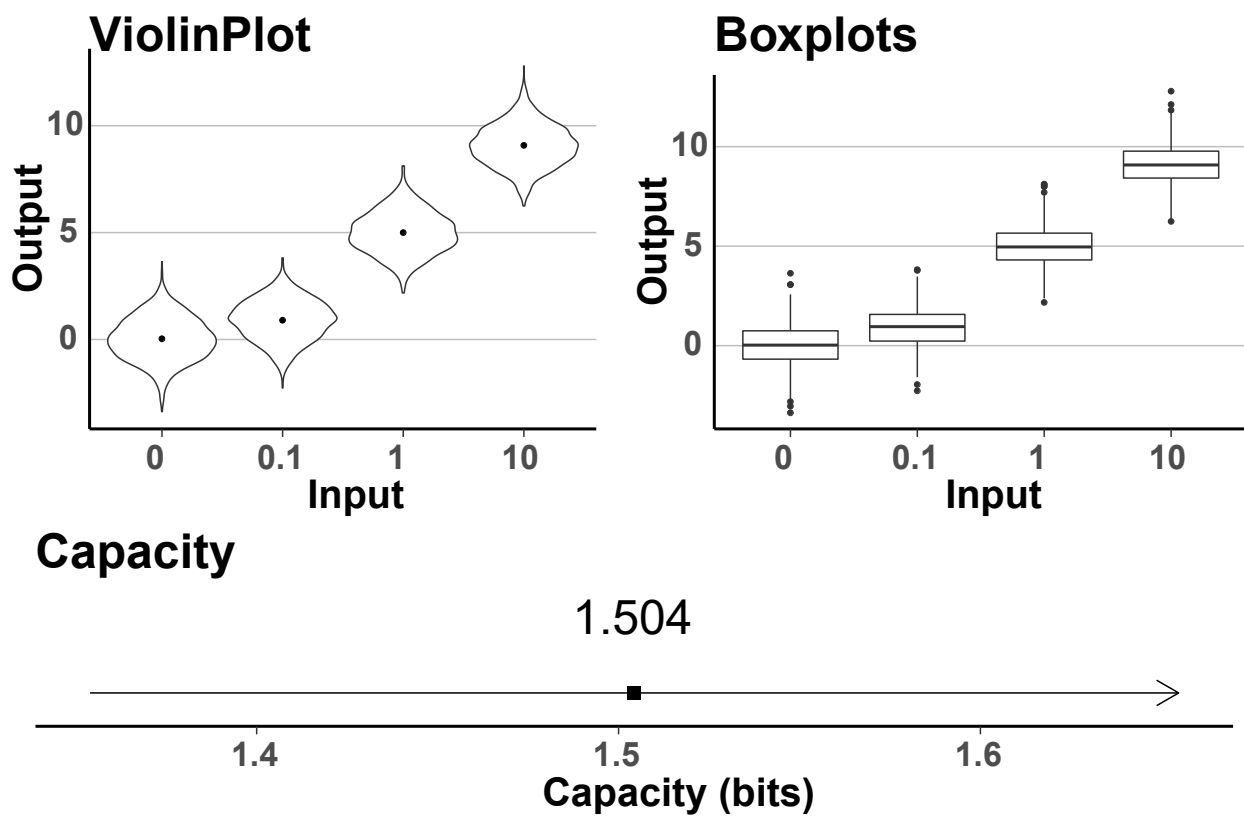


Figure 3: Standard output graph of the minimal working example

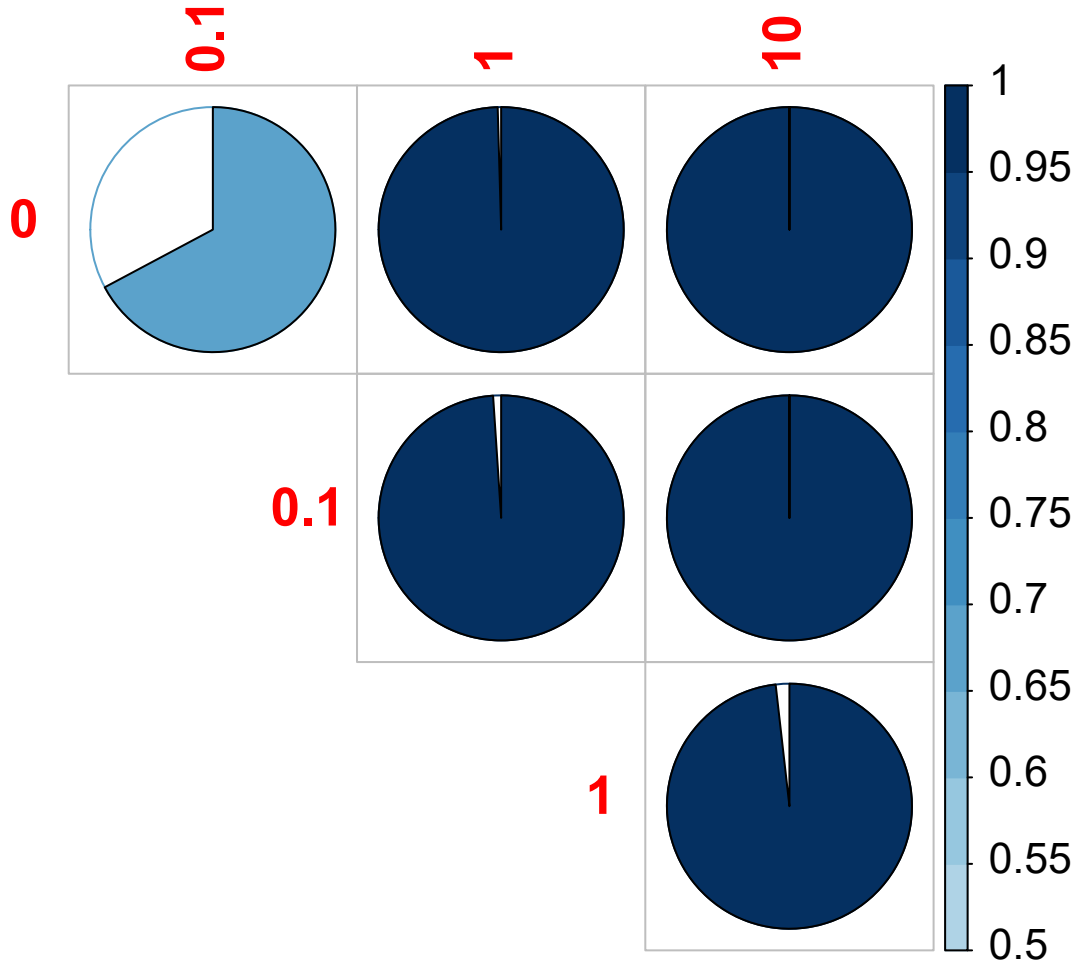


Figure 4: Standard output graph presenting probabilities of correct discrimination between each pair of input values.

```
output_path="minimal_exampleMI/",
pinput=rep(1/4,4))
```

and display results

```
print(paste("Mutual Information:", tempoutput_mi$mi, "; ",
           "Channel Capacity:", tempoutput_cc, sep=" "))
```

```
## [1] "Mutual Information: 1.48915671813912 ; Channel Capacity: 1.54047606466259"
```

#### Calculation of the probabilities of correct discrimination

Probabilities of correct discrimination between input values are calculated as follows

```
tempoutput_probs <- prob_discr_pairwise(dataRow=tempdata,
                                       signal="input", response="output",
                                       output_path="minimal_exampleProbs/")
```

The above command generates graph shown in Figure 4 in the output directory

## Diagnostics

The diagnostic test can be performed as follows

```
dir.create("example1_testing/")
outputCLR=capacity_logreg_main(dataRaw=data_example1,
                               signal="signal",response="response",
                               output_path="example1_testing/",
                               testing=TRUE, TestingSeed = 1234, testing_cores = 4,
                               boot_num = 40, boot_prob = 0.8,
                               traintest_num = 40,partition_trainfrac = 0.6)
```

It will run diagnostics with 40 re-sampling of the data, where bootstrap is calculated using 80% of the data, while the over-fitting test uses 60% of the original dataset.

Its results are provided in graph presented in Figure 5.

The top diagram shows the value of the capacity estimate (in black) obtained from the complete dataset and the mean value of bootstrap repetitions with indicated  $\pm$  standard deviation (in red). Plots that follow show histograms of calculated capacities for different diagnostic regimes. The black dot represents the estimate of the channel capacity based on the complete dataset. In addition, corresponding empirical p-values of both tests (left- and right-sided) are calculated to assess the randomness of obtained results (PV in the plots).

A reliable estimation of the information capacity should yield the following results of the bootstrap and overfitting tests.

1. The bootstrap test should yield distribution of the capacity estimates with small variance. In addition, the capacity estimated based on the complete dataset should not be an outlier (p-value $>0.05$ ). Otherwise, it would indicate that the sample size is too low for an accurate estimation of the channel capacity.
2. The over-fitting test should provide similar results. The capacity estimate obtained based on the complete dataset should lie within the distribution of capacities generated in the test. In the opposite case, it could mean that the logistic regression model does not fully grasp the essential aspects of input-output dependencies in the data.

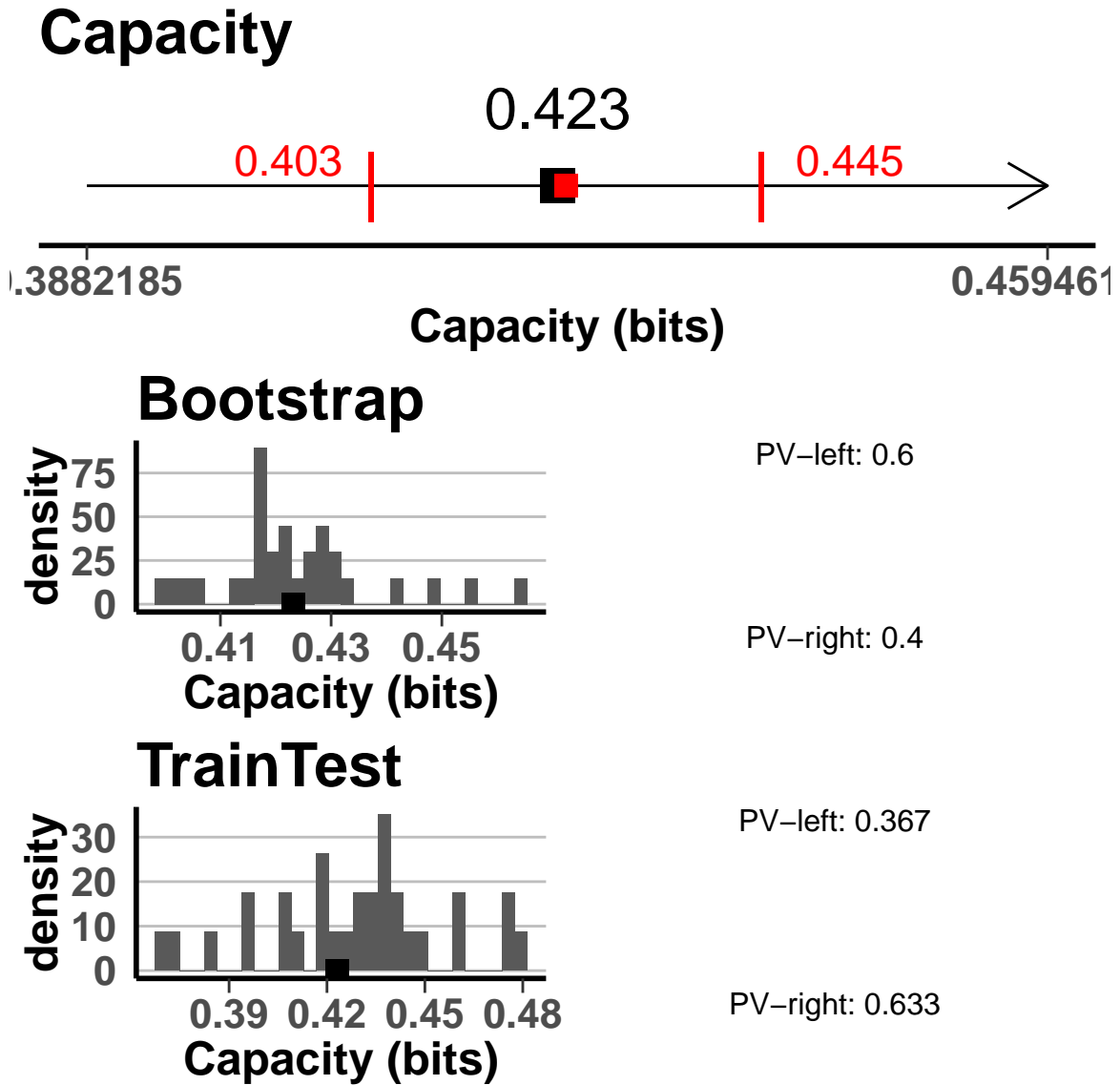


Figure 5: Standard output graph of the diagnostic procedures. P-values (PV) are based on empirical test either left- or right- sided. In the top axis, black dot represents the estimate of the channel capacity that involves the complete dataset, red dot is the mean of bootstrap procedures, while the bars are mean  $\pm$  sd. The remaining panels are histograms of all repetitions of a specific diagnostic procedure.

## 5.2 Further step-by-step introductory examples

Two step-by-step examples that further illustrate the applicability of the SLEMI package are provided in the Section 6 of the ‘testing procedures’ pdf file that is part of the package.

## 5.3 Examples in MP and SI

To reproduce results of the NFkB analysis presented in the MP as well as the results of the comparison with the KNN method presented in the Section 2 of the **SI** see Section 7 of the ‘testing procedures’ pdf file that is part of the package.

# 6 List of all package’s functions

The list below contains all functions available to the user:

- `capacity_logreg_main()` - is the main wrapper function that estimates channel capacity based on experimental data
- `capacity_logreg_algorithm()` - implements algorithm to estimate channel capacity using **nnet** package
- `mi_logreg_main()` - estimates mutual information
- `mi_logreg_algorithm()` - implements algorithm to estimate mutual information using **nnet** package
- `prob_discr_pairwise()` - estimates probabilities of discrimination between all pairs of input values

All other functionalities (graphs, testing procedures) are managed by internal functions. The tables below contain full specification of the package’s functions.

Function: `capacity_logreg_main()`

Main wrapper function to perform analysis of channel capacity from experimental data

Arguments		
name	description	default
<code>dataRaw</code>	data frame with input (X) and output (Y) values in separate columns	<b>(required)</b>
<code>signal</code>	character indicating a name of column of <code>dataRaw</code> with input (X)	<b>(required)</b>
<code>response</code>	character vector indicating names of columns of <code>dataRaw</code> with measurements of outputs (Y)	<b>(required)</b>
<code>output_path</code>	directory in which result and graphs will be saved	<b>(required)</b>
<code>scale</code>	logical indicating if preprocessing (centering and scaling) should be carried out before the analysis	TRUE
<code>model_out</code>	logical indicating if the model object should be returned	TRUE
<code>data_out</code>	logical indicating if the <code>dataRaw</code> should be returned with results	TRUE
<code>testing</code>	logical indicating if diagnostics should be performed	FALSE
<code>TestingSeed</code>	the seed of random number generator to be used in diagnostics	1234
<code>testing_cores</code>	number of cores to use in parallel computing in diagnostics	1
<code>boot_num</code>	the number of bootstrap tests to be performed (used if <code>testing=TRUE</code> )	10
<code>boot_prob</code>	the proportion of data to be used in bootstrap (used if <code>testing=TRUE</code> )	0.8
<code>traintest_num</code>	the number of over-fitting tests to be performed (used if <code>testing=TRUE</code> )	10
<code>partition_trainfrac</code>	the proportion of data to be used as a training dataset (used if <code>testing=TRUE</code> )	0.6
<code>side_variables</code>	an optional character vector indicating names of columns in <code>dataRaw</code> with side variables, if <code>NULL</code> no side variables are included in estimation	NULL
<code>sidevar_num</code>	is the number of resampling tests to be performed (used if <code>testing=TRUE</code> )	10
<code>plot_height</code>	the basic dimension of plots (height)	4
<code>plot_width</code>	the basic dimensions of plots (width)	6
<code>cc_maxit</code>	the maximum number of iteration to optimise channel capacity	100
<code>lr_maxit</code>	the maximum number of iteration to estimate logistic model	1000
<code>maxNWts</code>	the maximum number of parameters in logistic regression algorithm	5000
<code>formula_string</code>	character object that includes a formula syntax to use in logistic model	NULL

**Values** – a list with elements

name	description
<code>cc</code>	a numeric with the estimate of channel capacity (in bits)
<code>p_opt</code>	a numeric vector with estimated optimal input probability
<code>time</code>	processing time of the algorithm
<code>params</code>	a vector of parameters used in the algorithm
<code>data</code>	a <code>data.frame</code> with the raw experimental data (if <code>dataout=TRUE</code> )
<code>regression</code>	confusion matrix of logistic regression predictions
<code>model</code>	<code>nnet</code> object describing logistic regression model (if <code>model_out=TRUE</code> )
<code>testing</code>	a list of results of diagnostic procedures, e.g. <code>\$testing\$bootstrap</code> has <code>boot_num</code> elements, each with results of the algorithm of each diagnostic run
<code>testing_pv</code>	a list of left- and right-tailed p-values of diagnostic procedures



Function: `mi_logreg_main()`

Main wrapper function to mutual information from experimental data

Arguments		
name	description	default
<code>dataRaw</code>	data frame with input (X) and output (Y) values in separate columns	<b>(required)</b>
<code>signal</code>	character indicating a name of column of <code>dataRaw</code> with input (X)	<b>(required)</b>
<code>response</code>	character vector indicating names of columns of <code>dataRaw</code> with measurements of outputs (Y)	<b>(required)</b>
<code>output_path</code>	directory in which result and graphs will be saved	<b>(required)</b>
<code>scale</code>	logical indicating if preprocessing (centering and scaling) should be carried out before the analysis	TRUE
<code>model_out</code>	logical indicating if the model object should be returned	TRUE
<code>data_out</code>	logical indicating if the <code>dataRaw</code> should be returned with results	TRUE
<code>testing</code>	logical indicating if diagnostics should be performed	FALSE
<code>TestingSeed</code>	the seed of random number generator to be used in diagnostics	1234
<code>testing_cores</code>	number of cores to use in parallel computing in diagnostics	1
<code>boot_num</code>	the number of bootstrap tests to be performed (used if <code>testing=TRUE</code> )	10
<code>boot_prob</code>	the proportion of data to be used in bootstrap (used if <code>testing=TRUE</code> )	0.8
<code>traintest_num</code>	the number of over-fitting tests to be performed (used if <code>testing=TRUE</code> )	10
<code>partition_trainfrac</code>	the proportion of data to be used as a training dataset (used if <code>testing=TRUE</code> )	0.6
<code>side_variables</code>	an optional character vector indicating names of columns in <code>dataRaw</code> with side variables, if <code>NULL</code> no side variables are included in estimation	NULL
<code>sidevar_num</code>	is the number of resampling tests to be performed (used if <code>testing=TRUE</code> )	10
<code>plot_height</code>	the basic dimension of plots (height)	4
<code>plot_width</code>	the basic dimensions of plots (width)	6
<code>pinput</code>	an optional numeric vector with arbitrary probabilities of input. If <code>NULL</code> , fractions of observations in full dataset of each class are used.	NULL
<code>lr_maxit</code>	the maximum number of iteration to estimate logistic model	1000
<code>maxNWts</code>	the maximum number of parameters in logistic regression algorithm	5000
<code>formula_string</code>	character object that includes a formula syntax to use in logistic model	NULL

**Values** – a list with elements

name	description
<code>mi</code>	a numeric with the estimate of mutual information (in bits)
<code>pinput</code>	a numeric vector with prior probabilities of input
<code>time</code>	processing time of the algorithm
<code>params</code>	a vector of parameters used in the algorithm
<code>data</code>	a <code>data.frame</code> with the raw experimental data (if <code>dataout=TRUE</code> )
<code>regression</code>	confusion matrix of logistic regression predictions
<code>model</code>	<code>nnet</code> object describing logistic regression model (if <code>model_out=TRUE</code> )
<code>testing</code>	a list of results of diagnostic procedures, e.g. <code>\$testing\$bootstrap</code> has <code>boot_num</code> elements, each with results of the algorithm of each diagnostic run
<code>testing_pv</code>	a list of left- and right-tailed p-values of diagnostic procedures

Function: `prob_discr_pairwise()`

Computation of pairwise probabilities of discrimination

Arguments		
name	description	default
<code>dataRaw</code>	data frame with input (X) and output (Y) values in separate columns	<b>(required)</b>
<code>signal</code>	character indicating a name of column of dataRaw with input (X)	<b>(required)</b>
<code>response</code>	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	<b>(required)</b>
<code>output_path</code>	directory in which result and graphs will be saved	<b>(required)</b>
<code>scale</code>	logical indicating if preprocessing (centering and scaling) should be carried out before the analysis	TRUE
<code>diagnostics</code>	is a logical indicating if details of logistic regression fitting should be included in output list	TRUE
<code>side_variables</code>	an optional character vector indicating names of columns in dataRaw with side variables, if <code>NULL</code> no side variables are included in estimation	NULL
<code>lr_maxit</code>	the maximum number of iteration to estimate logisitic model	1000
<code>maxNWts</code>	the maximum number of parameters in logistic regression algorithm	5000
<code>formula_string</code>	character object that includes a formula syntax to use in logistic model	NULL

**Values** – a graph of pie charts is created in `output_path` directory.

In addition, function returns a list with elements

name	description
<code>prob_matr</code>	a symmetric numeric matrix of size $\text{length}(\text{unique}(\text{dataRaw}[[\text{signal}]]) \times \text{length}(\text{unique}(\text{dataRaw}[[\text{signal}]])$ with probability of discriminating between i-th and j-th input values in [i,j] cell
<code>diagnostics</code>	a list of diagnostics summaries that correspond to logistic regression models fitted for each pair of input values (if <code>diagnostics=TRUE</code> )

Function: `capacity_logreg_algorithm()`

Implements algorithm to estimate channel capacity using **nnet** package

Arguments		
name	description	default
data	data frame with input (X) and output (Y) values in separate columns	<b>(required)</b>
signal	character indicating a name of column of dataRaw with input (X)	<b>(required)</b>
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	<b>(required)</b>
model_out	logical indicating if the model object should be returned	TRUE
side_variables	an optional character vector indicating names of columns in dataRaw with side variables, if <b>NULL</b> no side variables are included in estimation	NULL
cc_maxit	the maximum number of iteration to optimise channel capacity	100
lr_maxit	the maximum number of iteration to estimate logisitic model	1000
maxNWts	the maximum number of parameters in logistic regression algorithm	5000
formula_string	character object that includes a formula syntax to use in logistic model	NULL

**Values** – a list with elements

name	description
cc	a numeric with the estimate of channel capacity (in bits)
p_opt	a numeric vector with estimated optimal input probability
regression	confusion matrix of logistic regression predictions
model	<b>nnet</b> object describing logistic regression model (if <b>model_out=TRUE</b> )

Function: `mi_logreg_algorithm()`

Implements algorithm to estimate mutual information using `nnet` package

Arguments		
name	description	default
data	data frame with input (X) and output (Y) values in separate columns	<b>(required)</b>
signal	character indicating a name of column of dataRaw with input (X)	<b>(required)</b>
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	<b>(required)</b>
model_out	logical indicating if the model object should be returned	TRUE
side_variables	an optional character vector indicating names of columns in dataRaw with side variables, if <code>NULL</code> no side variables are included in estimation	NULL
lr_maxit	the maximum number of iteration to estimate logisitic model	1000
maxNWts	the maximum number of parameters in logistic regression algorithm	5000
formula_string	character object that includes a formula syntax to use in logistic model	NULL

**Values** – a list with elements

name	description
mi	a numeric with the estimate of mutual information (in bits)
pinput	a numeric vector with prior probabilities of input
regression	confusion matrix of logistic regression predictions
model	<code>nnet</code> object describing logistic regression model (if <code>model_out=TRUE</code> )