Documentation of Statistical Learning based Estimation of Mutual Information (SLEMI) R package

T. Jetka, K. Nienaltowski, T. Winarski, M. Komorowski

July 22, 2018

Contents

The package SLEMI is designed to estimate channel capacity, mutual information, and probabilities of correct discrimination for signaling systems with a discrete input and multidimensional output. The package is based on the estimation of conditional input distributions using logistic regression.

SLEMI is released under the GNU license and is freely available from GitHub. Comprehensive documentation is available also on github.io. In case of any bugs, problems, and questions regarding the package or inquiries about information theory, contact t.jetka at sysbiosig.org The package is the component of the paper

Jetka et al., Information-theoretic analysis of multivariate signaling responses using SLEMI, 2018.

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: >= 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 running in the R console

```
install_packages("name_of_a_package")
```

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

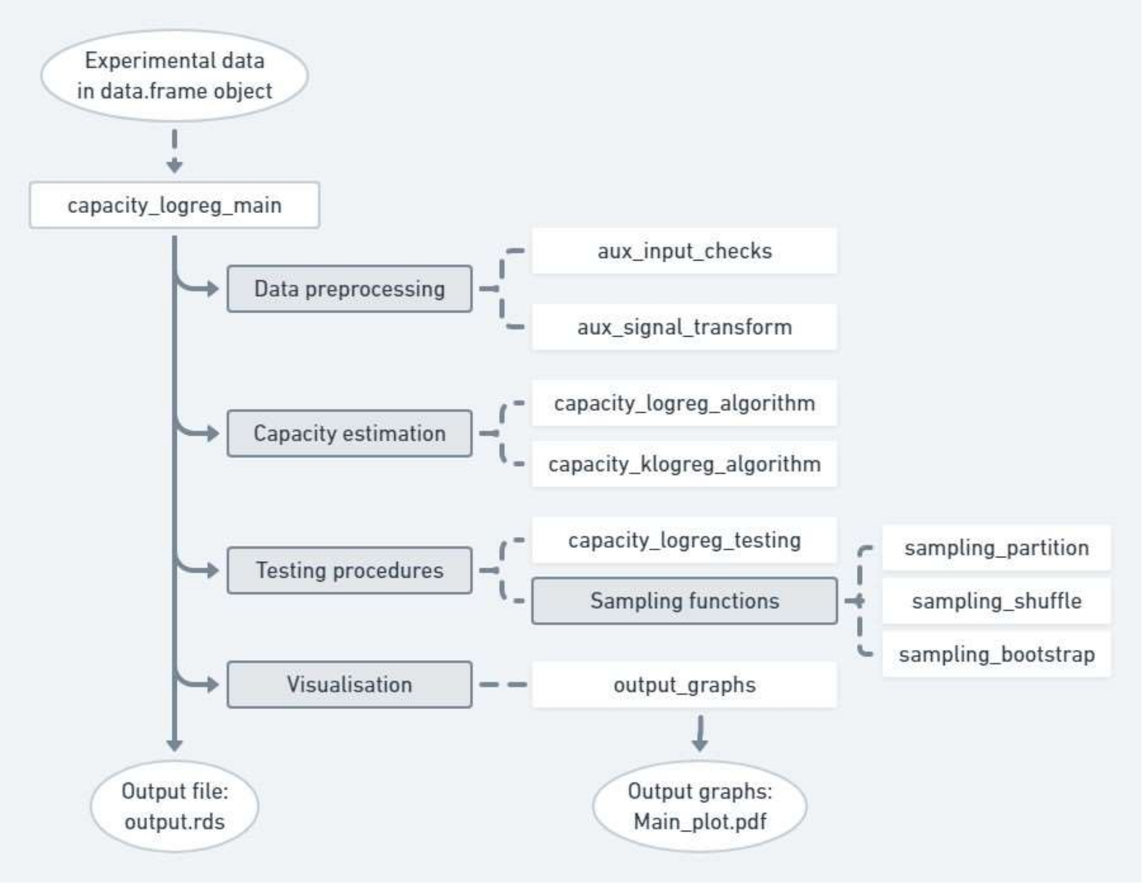
```
install_packages("devtools") # run if not installed
library(devtools)
install_github("TJetka/SLEMI")
```

All required packages are not found, they will be installed automatically.

2 Structure of the package

Estimation of the channel capacity is the core functionality of the package and is performed by the function`capacity_logreg_main()`. Estimation of the mutual information is performed by the function `mi_logreg_main()`, whereas probabilities of correct discrimination by the function `prob_discr_pairwise()`. Below, we outline the architectures of these functions.

The function `capacity_logreg_main()` triggers 1) preprocessing of the data; 2) estimation of channel capacity; 3) running diagnostic procedures; 4) visualisation. Dependencies between functions implemented in the package are presented in the graph below.



The algorithm to compute the information capacity is implemented within the function `capacity_logreg_algorithm()`, which uses logistic regression from `nnet package`. Diagnostic procedures (significance and uncertainties of estimates) are given in `capacity_logreg_testing()` function, which includes a) bootstrapping data and b) overfitting test. For visualization, a set of graphs is created by a 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, which leads to computation of the mutual information rather than of the information capacity.

An analogous approach based on logistic regression is assumed in a function `prob_discr_pairwise()`, which allows to estimate probabilities of discrimination between two different values of input. For each pair of input values it fits a model to find the accuracy of predicting the input from output measurements. Its result is given as a matrix of pie charts.

3 Basic workflow of the analysis

The package enables estimation of the information capacity, mutual information and probabilities of correct discrimination based on data that contain the output of the signaling system Y measured for a range of input values, x. Precisely, as described in detail in Section 1 of the Supplementary information, single cell responses y^i_j are assumed to be measured for a finite set of stimuli levels $x_1, x_2, \ldots, x_m$ and assumed to follow unknown distributions, $y^i_j \sim P(\cdot|X = x_i)$. Response y^i_j can be multidimensional.

3.1 Input data

Conceptually, a full experimental dataset in a described setting can be represented as a table.

	input	output 1	output 2	output 3	...
	x_1	$y^1_{1,1}$	$y^1_{1,2}$	$y^1_{1,3}$	
n_1	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	x_1	$y^1_{n_1,1}$	$y^1_{n_1,2}$	$y^1_{n_1,3}$	
	x_2	$y^2_{1,1}$	$y^2_{1,2}$	$y^2_{1,3}$	
n_2	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	x_2	$y^2_{n_2,1}$	$y^2_{n_2,2}$	$y^2_{n_2,3}$	
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...
	x_m	$y^m_{1,1}$	$y^m_{1,2}$	$y^m_{1,3}$	
n_m	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
	x_m	$y^m_{n_m,1}$	$y^m_{n_m,2}$	$y^m_{n_m,3}$	

In consequence, to use our package, it is required to represent experimental data as a `data.frame` R object with rows and columns organized as represented above. Specifically, we expect that for this data frame:

- 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). Those 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 paper 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_nkfb[1:3,1:4],data_nkfb[10001:10003,1:4],tail(data_nkfb[,1:4],3))
```

3.2 Calculation of information capacity and mutual information

To calculate channel capacity the user must call

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

where the required arguments are

- `dataRaw` - data frame with column of type `factor` containing values of input (X) and columns of type `numeric` containing values of output (Y), where each row represents a single observation
- `signal` - a character which indicates the name of the column in `dataRaw` with values of input (X)
- `response` - a character vector which indicates the names of columns in `dataRaw` with values of output (Y)
- `output_path` - a character with the directory, to which output should be saved

The function returns a list with elements

- 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 `dataout=TRUE`)
- time - processing time of the algorithm
- params - a vector of parameters used in the algorithm
- regression - a confusion matrix of logistic regression predictions
- logGraphs - a list of gg or ggtables objects with a standard set of exploratory graphs

By default, all returned elements are also saved in `output_path` directory in a file `output.rds` together with additional exploratory graphs which include:

- 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

Exactly the same basic list of arguments and set of plots are assumed for the function `mi_logreg_main()`, which estimates the mutual information (without optimising the optimal input distribution). The only exception is the resulting object of the function, which stores the value of the computed mutual information under the name

- mi - a numeric with mutual information estimate (in bits)

3.3 Calculation of probabilities of discrimination

To obtain only the probabilities of discriminating between input values, one can run

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

where the required arguments are analogous to arguments of functions `capacity_logreg_main()` and `mi_logreg_main()`. The accuracy of determining the most likely input is computed for each pair of unique input values and returns a list with elements

- prob_matr - a symmetric numeric matrix with a probability of discriminating between i-th and j-th input values in cell (i,j)
- model - a list of `nnet` objects describing fitted logistic regression models of classification two chosen input values.

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

3.4 Minimal working example

Below, we present a minimal working example that may serve as a quick introduction to the package. Firstly, we create a simple synthetic dataset. We assume a system with four different levels of stimulations: 0, 0.1, 1 and 10, a Log-normal output and Michaelis-Menten kinetics (compare with Test example 2 from Supplementary Information of the manuscript).

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

The generated data.frame has the following structure

	input	output
1	0	0.9160099
2	0	-1.5514693
2001	1	2.9531914
2002	1	5.0153107
3999	10	11.6404746
4000	10	7.8090393

The generated data frame contains sufficient data to calculate information-capacity using the `capacity_logreg_main()` function. The data frame “tempdata”" serves as `dataRaw` argument of the function, whereas 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. However, results are also 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.

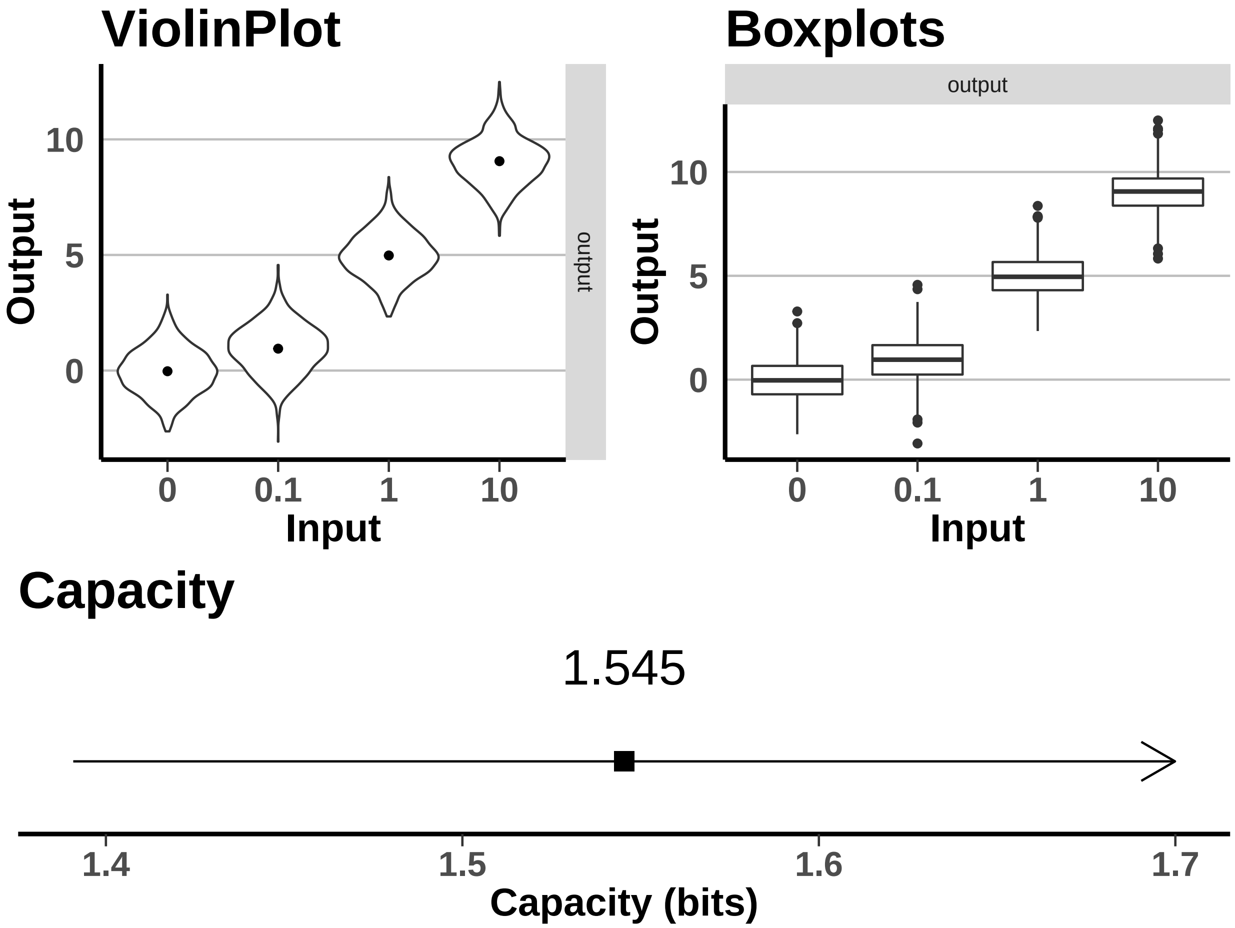


Figure 1: Standard output graph of minimal working example

To compare mutual information of experimental data with its channel capacity, we can run:

```
tempoutput_mi  <- mi_logreg_main(dataRaw=tempdata,
                                  signal="input", response="output",
                                  output_path="minimal_exampleMI/")
```

and show in the console their resepctive values

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

```
## [1] "Mutual Information: 1.48834475926489 ;  Channel Capacity: 1.54537652807866"
```

For deeper understanding, probabilities of discriminating between input values can be explored by

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

which generates a following graph in output directory

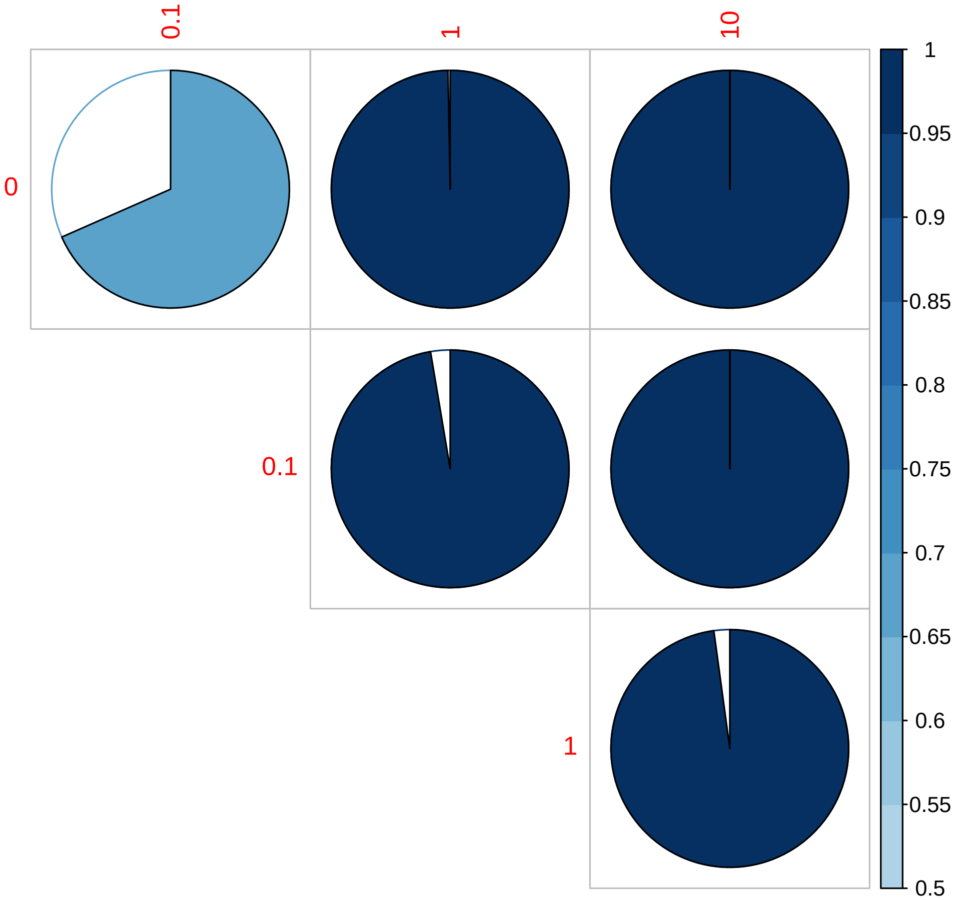


Figure 2: Standard output graph presenting probabilities of discriminating between each pair of unique input values.

4 Diagnostic procedures

The x-axis arrow used to plot the capacity serves to represent results of additional diagnostics computations described below. We implemented two diagnostic procedures to control the accuracy of the channel capacity estimates and to measure uncertainty due to finite sample size and model over-fitting. These include:

1. Bootstrap test - capacity is re-calculated using $\alpha\%$ of data, sampled from the original dataset without replacement. After repeating the procedure n times, its standard deviation can serve as an error of the obtained 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 use those procedures, user must provide additional arguments to function `logreg_capacity_main()`, i.e.

- testing (default=FALSE) - a logical value that turn on/off testing mode,
- 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

For example, a user can, by utilizing a synthetic dataset accompanying the package, run

```
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)
```

to run diagnostics with 40 re-sampling of data, where bootstrap is calculated using 80% of data, while over-fitting test include testing dataset of size 60% of original data.

It provides the following result

The top diagram shows the value of the original capacity estimate (in black) and the mean value of bootstrap repetitions with indicated +/- standard deviation (in red). Plots that follow show histograms of calculated capacities for different testing regimes. The black dot represents the basic channel capacity estimate. 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).

To ensure the correctness of the analysis

1. Bootstrap should yield distribution of capacity with small variance and basic capacity 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 channel capacity.
2. The over-fitting test should also provide a conclusion that the basic capacity estimate is non-significantly different from the distribution of capacities from 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.

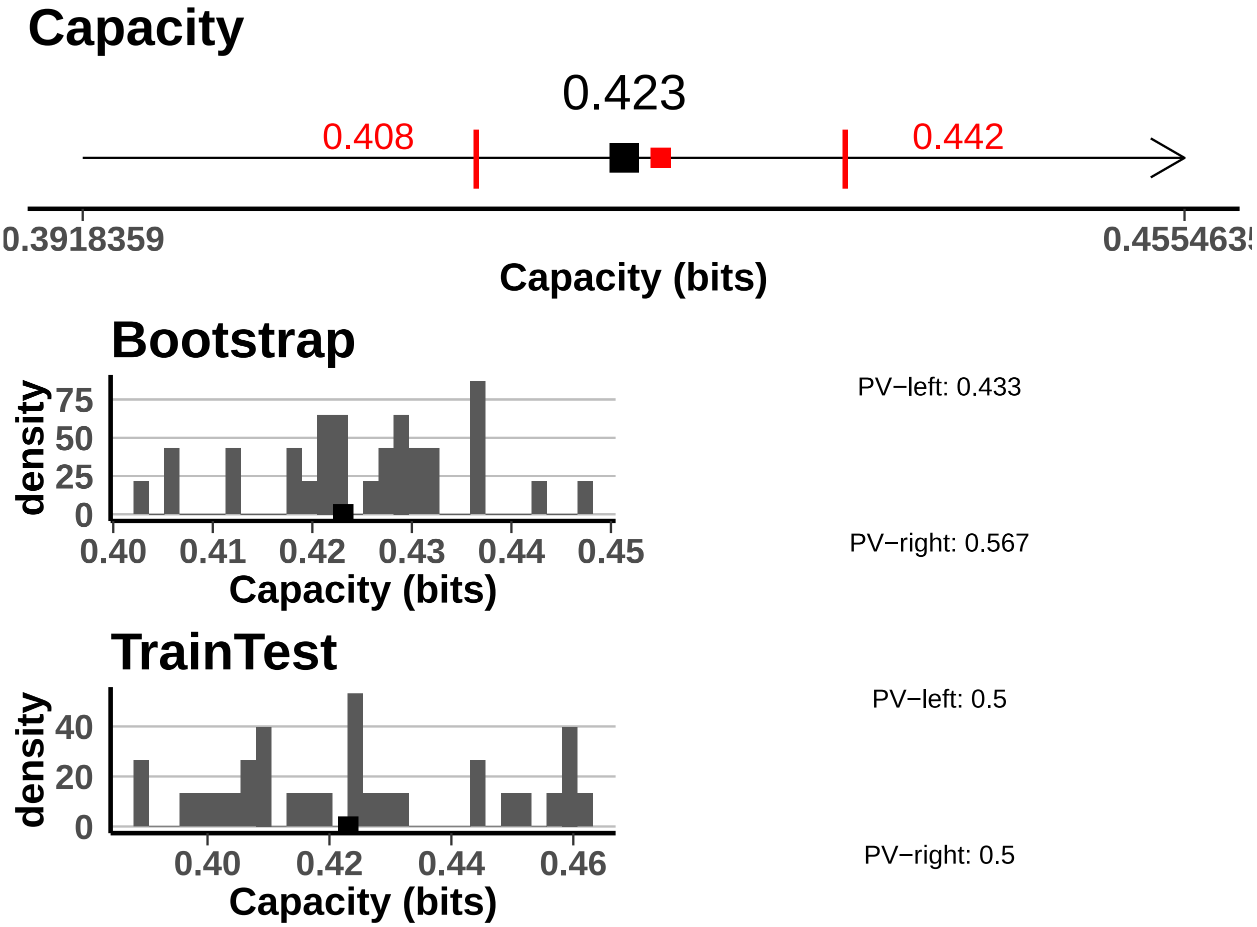


Figure 3: Standard output graph of testing procedures. P-values (PV) are based on empirical test either left- or right- sided. In the top vapacity axis, black dot represents original estimate of channel capacity (using full data set), red dot is the mean of bootstrap procedures, while the bars are mean +/- sd. The remaining panels are histograms of all repetitions of a specific testing procedure.

5 Other Functionalities

5.1 Additional parameters

Apart from required arguments, the function `capacity_logreg_main()` has also other parameters that can be used to tune the activity of the algorithm (see full specification at the end of the documentation). These are

- `model_out` (`default=TRUE`) - logical, specify if `nnet` model object should be saved into output file
- `graphs` (`default=TRUE`) - logical, controls creating diagnostic plots in the output directory.
- `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.

Calculation of the information capacity with `capacity_logreg_main()` is based on logistic regression implementation of `nnet` package. Some parameters of original `nnet::nnet` (used for fitting parameters) and `nnet::multinom` (used to implement logistic model) functions can be set via following arguments of `capacity_logreg_main()`.

Precisely,

- `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.

6 Examples

Below we present a step-by-step example that further illustrate the applicability of the SLEMI package. The R scripts with computations described below are available in the directory `testing_procedures` of the package's GitHub repository.

6.1 One-dimensional example

To demonstrate how to use SLEMI, we will generate a synthetic dataset of a channel, for which the conditional output distribution, $Y|X$, is Log-normal. This example is analogous to the Test example 2 from SI of the corresponding manuscript (Section 3.2) and is a simple case of an experimental setup commonly seen in systems biology problems. We assume that

i) input, X, has 6 different levels between 0 and 100;
ii) conditional output, $Y|X = x$, is a one-dimensional Log-normal distribution $\exp\{\mathcal{N}(10 \cdot \frac{x}{1+x}, 1)\}$;
iii) for each x, there are 1000 observations.

Our method expects as input a `data.frame` with a column indicating the value of the input (recommended type: `factor`) and other columns with corresponding output measurements (type: `numeric`). It will be created by the following steps

1. Set parameters of the example

```
# sample size for each input concentration;
# change to investigate its influence on estimation
n_sample <- c(1000)
# standard deviation of ln(Y)|X=x;
# change to investigate its influence on estimation
dist_sd  <- 1
# number of concentration of input X considered;
# change to investigate its influence on estimation
input_num <- 6
```

2. Create output directory

```
i_type <- "testing_basic"
path_output_main <-
  paste('output/',
    i_type,'/',
    sep="")
dir.create(
  path_output_main,
  recursive = TRUE)
```

3. Generating synthetic data

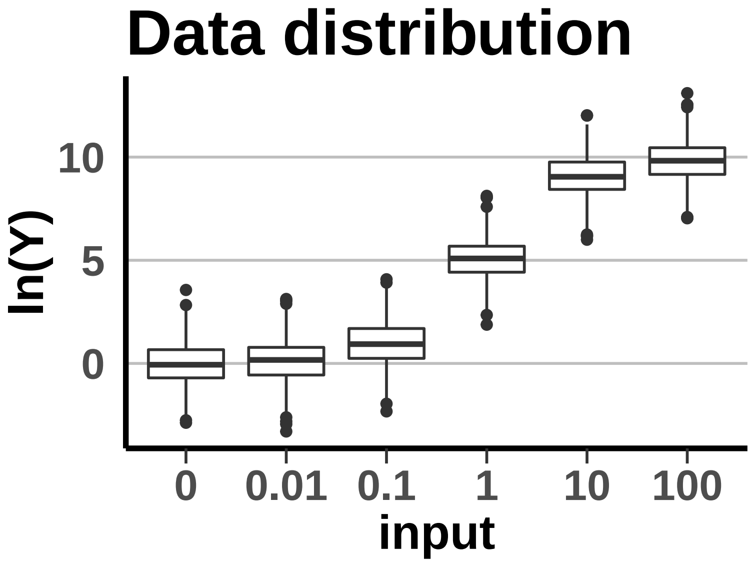
```
# concentration of input; spans from 0 to saturation.
xx <-
  signif(
c(0,
  exp(
    seq(
      from = log(0.01),
      to = log(100),
      length.out = input_num-1))),
digits = 2)
```

```
# mean of the dose-response relation; Michaelis-Menten assumed
example_means <- 10*(xx/(1+xx))
example_sds <- rep(dist_sd, input_num)
tempdata <-
  data.frame(
signal = c(t(replicate(n_sample, xx))),
output = c(matrix(
  rnorm(n = input_num*n_sample,
        mean = example_means,
        sd = example_sds),
  ncol = input_num,
  byrow = TRUE)))
tempdata$signal <-
  factor(
x = tempdata$signal,
levels = sort(unique(tempdata$signal)))
print(head(tempdata))
```

```
##   signal     output
## 1      0 -1.2070657
## 2      0 -0.5747400
## 3      0 -0.7762539
## 4      0 -0.8371717
## 5      0 -0.6937202
## 6      0  1.1022975
```

4. Preview the distribution of the data. It will create a graph as below

```
g_plot <- ggplot(
  data = tempdata,
  aes(x = factor(signal),
  group = signal,
  y = output)) +
  geom_boxplot() +
  theme_publ(version = 2)
if(display_plots){
  print(g_plot)
}
```



5. Running algorithm The estimation of channel capacity can be performed by using `capacity_logreg_main()` function. In most basic setting it expects four arguments: 1) data.frame with data, 2) name of input column, 3) names of output columns and 4) a path to the output directory. It can be obtained by

6. Set required parameters for the algorithm

```
signal_name <- "signal"
response_name <- "output"
```

7. Estimate channel capacity (takes several seconds)

```
tempoutput  <-
  capacity_logreg_main(
dataRaw = tempdata,
signal = signal_name,
response = response_name,
output_path = path_output_main
  )
```

8. Access and visualise results With the algorithm finished, results can be accessed by either exploring a list returned by the function or inspecting the visualisation implemented within the package (graphs created in output directory)

9. Print results of the estimation in the console

```
print(paste("Channel Capacity (bit):",
            tempoutput$cc,
            sep=" "))
```

```
## [1] "Channel Capacity (bit): 1.60456660237405"
```

```
print(paste("Optimal input probabilities,",
            "x_i = ",sort(unique(tempdata$signal)),": ",
            paste(tempoutput$p_opt,
               sep = "\n"),
            sep = " " ))
```

```
## [1] "Optimal input probabilities, x_i =  0 :  0.20276215140046"
## [2] "Optimal input probabilities, x_i =  0.01 :  0.00351480233210017"
## [3] "Optimal input probabilities, x_i =  0.1 :  0.145420938360615"
## [4] "Optimal input probabilities, x_i =  1 :  0.308751686587878"
## [5] "Optimal input probabilities, x_i =  10 :  0.126971845357898"
## [6] "Optimal input probabilities, x_i =  100 :  0.212578575961048"
```

```
print(paste("Accuracy of classification:",
            tempoutput$regression$overall[1],
            sep = " " ))
```

```
## [1] "Accuracy of classification: 0.610333333333333"
```

```
print(paste("Time of computations (sec.):",
            tempoutput$time[3],
            sep = " " ))
```

```
## [1] "Time of computations (sec.): 3.14"
```

10. Inspect object generated during computation. We verify if results saved in rds are the same as in the returned list. Full output of the estimation should be saved in "output_path/output.rds".

```
tempoutput_rds = readRDS(
  paste0(
    path_output_main,
    "/output.rds"))
print(paste("Channel Capacity assertion:",
            tempoutput_rds$cc == tempoutput$cc))
```

```
## [1] "Channel Capacity assertion: TRUE"
```

```
print(paste("Optimal input probabilities assertion:",
            sum(tempoutput_rds$p_opt != tempoutput$p_opt) == 0))
```

```
## [1] "Optimal input probabilities assertion: TRUE"
```

```
print(paste("Accuracy of classification assertion:",
            tempoutput_rds$regression$overall[1] ==
              tempoutput$regression$overall[1]))
```

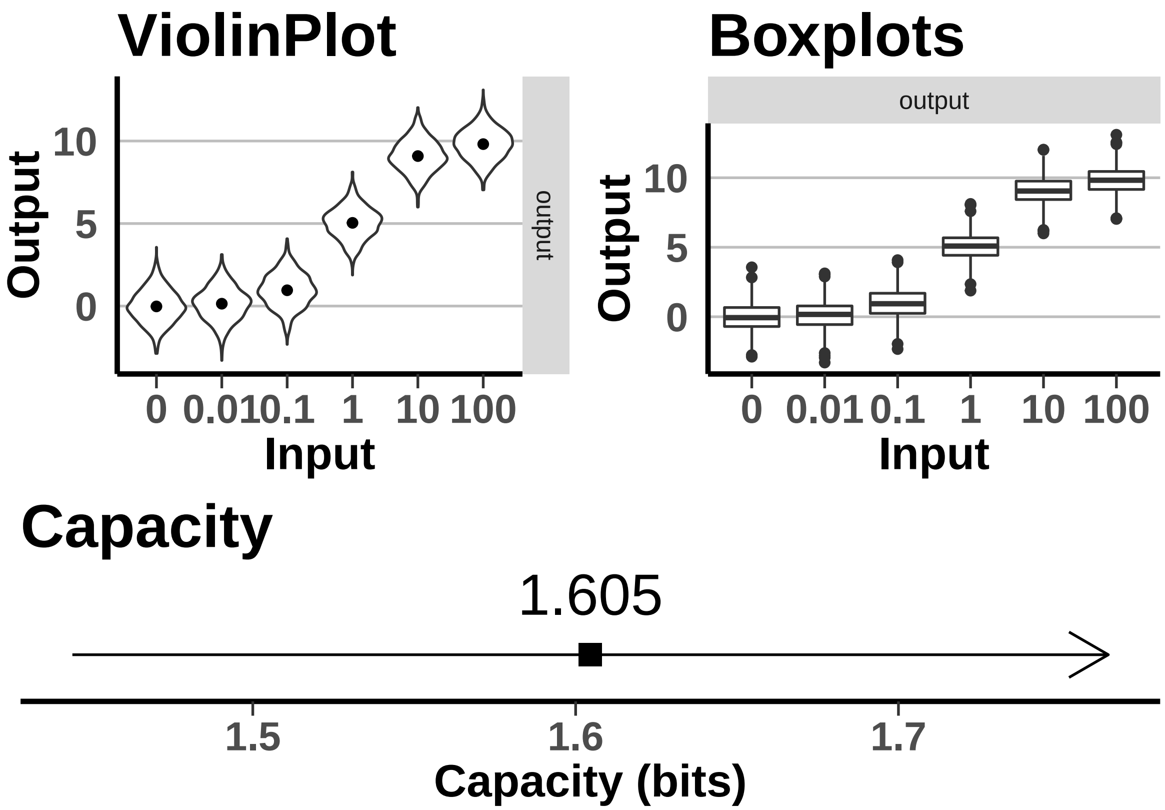
```
## [1] "Accuracy of classification assertion: TRUE"
```

```
print(paste("Time of computations assertion:",
            tempoutput_rds$time[3] ==
              tempoutput$time[3]))
```

```
## [1] "Time of computations assertion: TRUE"
```

11. Explore visualisation of the results. See pdf files in output_path/ for the visualisation of the data and capacity estimation. In the "MainPlot.pdf" the most important information are presented: mean input-output relation; distributions of output and channel capacity (see below). Graphs, as gg or gtable objects, are saved in `logGraphs` element of function output.

    ```
    if(display_plots){
      grid.arrange(tempoutput$logGraphs[[9]])
    }
    ```



12. Generate graphs of different size by specifying parameters `plot_width` and `plot_height`

```
# specify paramters `plot_width` and `plot_height`
plot_width_new  <- 10
plot_height_new <- 8
i_type <- "testing_basic_graphs_size"
path_output_main <- paste('output/', i_type,'/', sep="")
dir.create(path_output_main,
       recursive = TRUE)
tempoutput  <- capacity_logreg_main(
  dataRaw = tempdata,
  signal = signal_name,
  response = response_name,
  output_path = path_output_main,
  plot_width = plot_width_new,
  plot_height = plot_height_new
)
```

13. Run analysis without generating graphs by setting argument `graphs` to FALSE

```
# set argument `graphs` to FALSE
graphs_generate <- FALSE
i_type <- "testing_basic_nographs"
path_output_main <- paste('output/',i_type,'/',sep="")
dir.create(path_output_main,
       recursive = TRUE)
tempoutput  <- capacity_logreg_main(
  dataRaw = tempdata,
  signal = signal_name,
  response = response_name,
  output_path = path_output_main,
  graphs = graphs_generate
)
```

14. Run analysis with the minimal output by setting `graphs`, `scale`, `dataout`, `model_out` to FALSE. Respectively, it prevents creating graphs, scaling of the data, including data and regression model in the returned list. Such setting is useful mainly for batch processing.

```
graphs_generate <- FALSE
data_rescale <- FALSE
data_save <- FALSE
model_save <- FALSE
i_type <- "testing_basic_minimalOutput"
path_output_main=paste('output/',i_type,'/',sep="")
dir.create(path_output_main,recursive = TRUE)
tempoutput  <- capacity_logreg_main(
  dataRaw = tempdata,
  signal = signal_name,
  response = response_name,
  output_path = path_output_main,
  graphs = graphs_generate,
  scale = data_rescale,
  dataout = data_save,
  model_out = model_save
)
```

15. Diagnostics We implemented two diagnostic procedures to test the correctness of channel capacity estimation and to compute uncertainties due to sample size and over-fitting. These include:

16. Set parameters of diagnostic tests

```
# Set a seed of a random number generator for reproducibility
seed_to_use <- 12345
# number of bootstrap repetitions
bootstrap_num <- 20
# fraction of data to sample
bootstrap_frac <- 0.8
# number of repetition of overfitting test
overfitting_num <- 20
# fraction of data to use as training sample
training_frac <- 0.6
i_type <- "testing_basic_diagnostic"
path_output_main <- paste('output/',i_type,'/',sep="")
dir.create(path_output_main,
           recursive = TRUE)
```

17. Run estimation with full diagnostics by using parameters and set `testing` argument to TRUE (takes up to 5 min)

```
tempoutput  <- capacity_logreg_main(
  dataRaw = tempdata,
  signal = signal_name,
  response = response_name,
  output_path = path_output_main,
  testing = TRUE,
  plot_width = 10 ,
  plot_height = 8,
  TestingSeed = seed_to_use,
  testing_cores = cores_num,
  boot_num = bootstrap_num,
  boot_prob = bootstrap_frac,
  traintest_num = overfitting_num,
  partition_trainfrac = training_frac
)
```

18. Inspect results of diagnostic tests. It is saved in the `testing` element of the returned list

```
print(paste("Channel Capacity, bootstrap mean (sd): ",
            round(mean(sapply(tempoutput$testing$bootstrap,
                        function(x) x$cc)),digits = 2),
            "(",round(sd(sapply(tempoutput$testing$bootstrap,
                          function(x) x$cc)),digits=2),")",
            sep = "" ))
```

```
## [1] "Channel Capacity, bootstrap mean (sd): 1.61(0)"
```

```
print(paste("Time of computations (sec.):",
            tempoutput$time[3],
            sep = " "))
```

```
## [1] "Time of computations (sec.): 39.64"
```

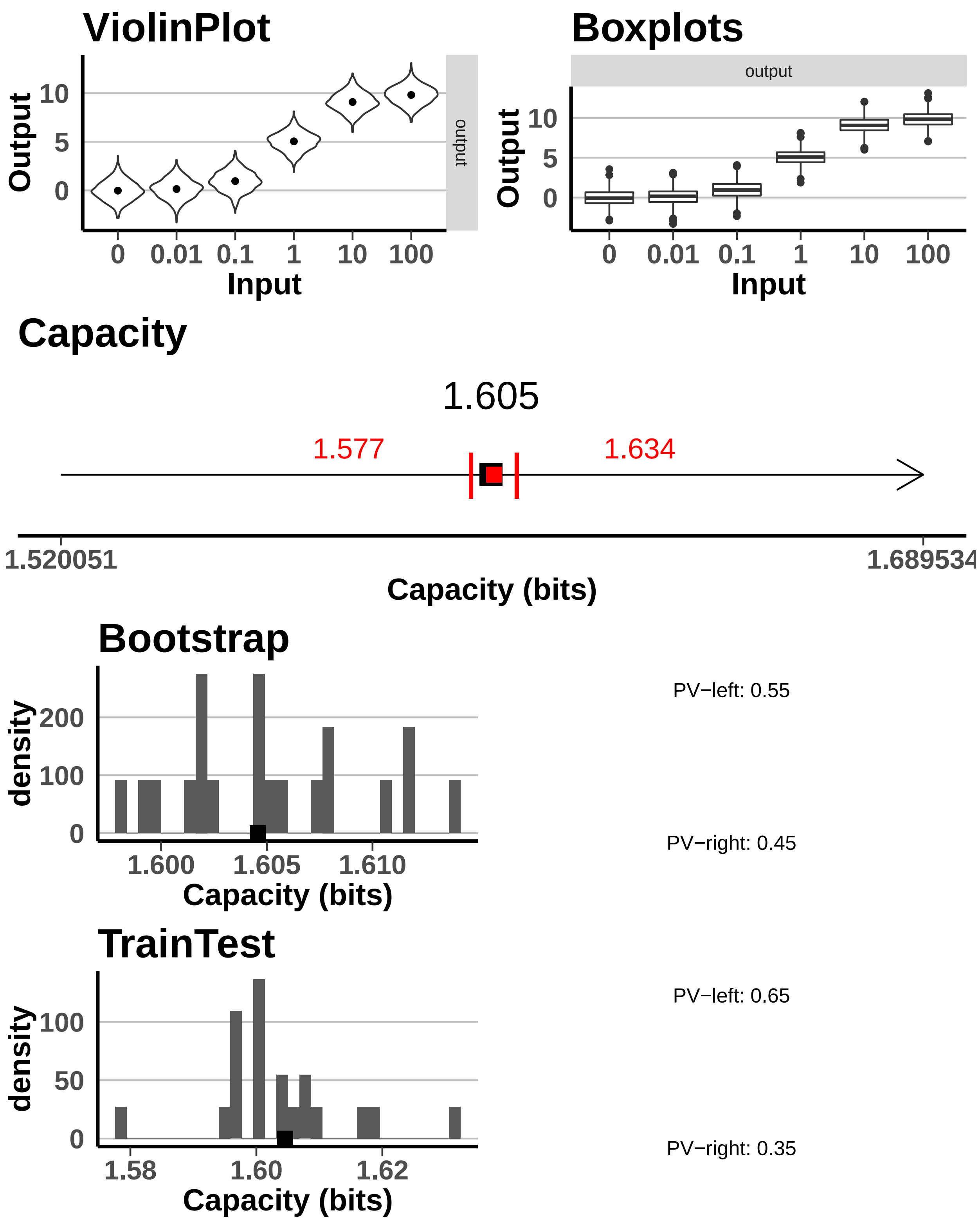
19. See the visualisation in the output directory - MainPlot.pdf. For each diagnostic test, there is a corresponding histogram of calculated capacities. This graph is also obtainable from the returned list by a command

    ```
    if(display_plots){
      grid.arrange(tempoutput$logGraphs[[9]])
    }
    ```

20. For each diagnostic test, we provide left- and right-tailed empirical p-values of the obtained channel capacity. They indicated if the regime of data bootstrapping or dividing data into training and testing sample influence the calculation of capacity in a significant way. A small p-value in any of these tests (e.g. <0.05) means a problem with the stability of channel capacity estimation and a possible bias due to too small sample size. P-values are printed on the MainPlot.pdf graph or can be obtained in

```
print(paste("P-values:",
            tempoutput$testing_pv$bootstrap[1],
            " (left-tailed); ",
            tempoutput$testing_pv$bootstrap[2],
            " (right-tailed) ",
            sep = ""))
```

```
## [1] "P-values:0.3 (left-tailed); 0.7 (right-tailed) "
```


ViolinPlot
Output
10
5
0
output
0
0.01
0.1
1
10
100
Input
Boxplots
output
Output
10
5
0
0
0.01
0.1
1
10
100
Input
Capacity
1.605
1.577
1.634
1.520051
1.689534
Capacity (bits)
Bootstrap
density
200
100
0
1.600
1.605
1.610
Capacity (bits)
PV−left: 0.55
PV−right: 0.45
TrainTest
density
100
50
0
1.58
1.60
1.62
Capacity (bits)
PV−left: 0.65
PV−right: 0.35

6.2 NfkB experimental data

The data set on NfkB pathway responses to TNF-α analysed in the corresponding manuscript is attached to the package as a data.frame object and can be accessed using `data_nfkb` variable. Each row of `data_nfkb` corresponds to an individual cell. Column 'signal' indicates the level of TNF-α stimulation, while columns 'response_T', gives the normalised ratio of nuclear and cytoplasmic fluorescence measured over time as described in Supplementary Information.

Below, we show how the analysis of the NfkB pathway presented in the main paper can be replicated. Specifically, we replicate Fig. 1 from MP.

Our investigation of the influence of the TNF-α on NfkB responses includes the analysis of information transfer by estimating channel capacity (Fig. 1A-B) and assessing the probability of discrimination between each pair of the TNF-α stimulation (Fig. 1C-D). Both of above approaches are applied to two variants of data: i) for time point (TP) measurements; ii) for time-series (TS) measurements.

The code is given in `paper_MP.R` file of the package and is divided into three sections that should be run consecutively:

1) Preliminary - setting up packages and working environment. (lines 60-77)
2) Capacity - replicates Fig.1 A-B (lines 79-180)
3) Probabilities of discrimination - replicates Fig.1 C-D (lines 184-299)

In default mode (5 repetition of bootstrap), running Capacity section takes approx. 2 hours with a single core. Set number of cores for parallel processing in line 83. For a graphs exactly like in MP, set line 84 to

```
analysis_type="long"
```

Running Probability of discrimination section takes about 3 minutes. Please follow instructions and run code in `paper_MP.R` to replicate the results from MP. Installation of additional packages is needed: `ggplot2`, `gridExtra`, `mvtnorm`, and `corrplot`.

6.3 Replication of the Results of the Section 2 of the Supplementary Information

In script `paper_SI.R` we present codes to replicate examples presented in the Section 3 of the Supplementary Methods. The code is divided into three sections that should be run in consecutively

1) Preliminary - setting up packages and working environment
2) Comparison - replicates Fig. S1 - shows the comparison of our method to the KNN approach.
3) Validation - replicates Fig. S3 - shows the performance of our method in four examples of simple channels

In default mode (10 repetition of data sampling), running Validation section takes 1 hour, similarly computations in Comparison section also take approximately 1 hour with a single core. Set number of cores for parallel processing in line 76. For a graphs exactly like in SI, set line 77 to

```
analysis_type="long"
```

Please follow instructions and run codes in `paper_SI.R` to replicate the results from SI. Installation of additional packages is needed: `ggplot2`, `gridExtra`, `mvtnorm`, `nloptr`, `FNN`, `DEoptimR`, `TDA` and `corrplot`.

7 List of all functions in the package

Following functions from SLEMI package are available to the user:

- `capacity_logreg_main()` - Main wrapper function to perform analysis of channel capacity of experimental data
- `capacity_logreg_algorithm()` - Implements algorithm to estimate channel capacity using `nnet` package
- `capacity_klogreg_algorithm()` - Implements algorithm to estimate channel capacity using `glmnet` package
- `capacity_logreg_testing()` - Performs diagnostic procedures
- `capacity_output_graphs()` - Generates exploratory graphs
- `mi_logreg_main()` - Estimates mutual information
- `prob_discr_pairwise()` - Calculates probabilities of discrimination between all pairs of input values
- `formula_generator()` - Generates a formula object based on input and output specification
- `sampling_bootstrap()`, `sampling_partition()`, `sampling_shuffle()` - Used to generate subsets of data to use in diagnostic procedures
- `theme_publ()` - Changes the visual elements of ggplot object

Function: `capacity_logreg_main()`
Main wrapper function to perform analysis of channel capacity from experimental data

Arguments

name	description	default
dataRaw	data frame with input (X) and output (Y) values in separate columns	**(required)**
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
output_path	directory in which result and graphs will be saved	**(required)**
scale	logical indicating if preprocessing (centering and scaling) should be carried out before the analysis	`TRUE`
graphs	logical indicating if standard graphs should be created	`TRUE`
model_out	logical indicating if the model object should be returned	`TRUE`
dataout	logical indicating if the dataRaw should be returned with results	`TRUE`
testing	logical indicating if diagnostics should be performed	`FALSE`
TestingSeed	the seed of random number generator to be used in diagnostics	1234
testing_cores	number of cores to use in parallel computing in diagnostics	1
boot_num	the number of bootstrap tests to be performed (used if `testing=TRUE`)	10
boot_prob	the proportion of data to be used in bootstrap (used if `testing=TRUE`)	0.8
traintest_num	the number of over-fitting tests to be performed (used if `testing=TRUE`)	10
partition_trainfrac	the proportion of data to be used as a training dataset (used if `testing=TRUE`)	0.6
side_variables	an optional character vector indicating names of columns in dataRaw with side variables, if `NULL` no side variables are included in estimation	`NULL`
resamp_num	is the number of resmapling tests to be performed (used if `testing=TRUE`)	10
plot_height	the basic dimnesion of plots (height)	4
plot_width	the basic dimnesions of plots (width)	6
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`
glmnet_algorithm	logical indicating if the `glmnet` package should be used	`FALSE`
dataMatrix	numeric matrix with columns treated as explanatory variables in logistic model (used if `glmnet_algorithm=TRUE`)	`NULL`
glmnet_cores	the number of cores to use in parallel computing of `glmnet package` (used if `glmnet_algorithm=TRUE`)	1
glmnet_lambdanum	is the lambda parameter as in `glmnet` package (used if `glmnet_algorithm=TRUE`)	10

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
time	processing time of the algorithm
params	a vector of parameters used in the algorithm
data	a data.frame with the raw experimental data (if `dataout=TRUE`)
regression	confusion matrix of logistic regression predictions
model	nnet object describing logistic regression model (if `model_out=TRUE`)
logGraphs	a list of gg or ggtables objects with a standard set of exploratory graphs (if `graphs=TRUE`)
testing	a list of results of diagnostic procedures, e.g. $testing$bootstrap has `boot_num` elements, each with results of the algorithm of each diagnostic run
testing_pv	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
dataRaw	data frame with input (X) and output (Y) values in separate columns	**(required)**
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
output_path	directory in which result and graphs will be saved	**(required)**
scale	logical indicating if preprocessing (centering and scaling) should be carried out before the analysis	`TRUE`
graphs	logical indicating if standard graphs should be created	`TRUE`
model_out	logical indicating if the model object should be returned	`TRUE`
dataout	logical indicating if the dataRaw should be returned with results	`TRUE`
testing	logical indicating if diagnostics should be performed	`FALSE`
TestingSeed	the seed of random number generator to be used in diagnostics	1234
testing_cores	number of cores to use in parallel computing in diagnostics	1
boot_num	the number of bootstrap tests to be performed (used if `testing=TRUE`)	10
boot_prob	the proportion of data to be used in bootstrap (used if `testing=TRUE`)	0.8
traintest_num	the number of over-fitting tests to be performed (used if `testing=TRUE`)	10
partition_trainfrac	the proportion of data to be used as a training dataset (used if `testing=TRUE`)	0.6
side_variables	an optional character vector indicating names of columns in dataRaw with side variables, if `NULL` no side variables are included in estimation	`NULL`
resamp_num	is the number of resmapling tests to be performed (used if `testing=TRUE`)	10
plot_height	the basic dimnesion of plots (height)	4
plot_width	the basic dimnesions of plots (width)	6
pinput	an optional numeric vector with arbitrary probabilities of input. If `NULL`, fractions of observations in full dataset of each class are used.	`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`
glmnet_algorithm	logical indicating if the `glmnet` package should be used	`FALSE`
dataMatrix	numeric matrix with columns treated as explanatory variables in logistic model (used if `glmnet_algorithm=TRUE`)	`NULL`
glmnet_cores	the number of cores to use in parallel computing of `glmnet package` (used if `glmnet_algorithm=TRUE`)	1
glmnet_lambdanum	is the lambda parameter as in `glmnet` package (used if `glmnet_algorithm=TRUE`)	10

Values – a list with elements

name	description
mi	a numeric with the estimate of mutual information (in bits)
p_opt	a numeric vector with estimated optimal input probability
time	processing time of the algorithm
params	a vector of parameters used in the algorithm
data	a data.frame with the raw experimental data (if `dataout=TRUE`)
regression	confusion matrix of logistic regression predictions
model	nnet object describing logistic regression model (if `model_out=TRUE`)
logGraphs	a list of gg or ggtables objects with a standard set of exploratory graphs (if `graphs=TRUE`)
testing	a list of results of diagnostic procedures, e.g. $testing$bootstrap has `boot_num` elements, each with results of the algorithm of each diagnostic run
testing_pv	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
dataRaw	data frame with input (X) and output (Y) values in separate columns	**(required)**
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
output_path	directory in which result and graphs will be saved	**(required)**
scale	logical indicating if preprocessing (centering and scaling) should be carried out before the analysis	`TRUE`
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 `NULL` 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 graph of pie charts is created in `output_path` directory.
In addition, function returns a list with elements

name	description
prob_matr	a symmetric numeric matrix of size `length(unique(dataRaw[[signal]]))`×`length(unique(dataRaw[[signal]]))` with probability of discriminating between i-th and j-th input values in [i,j] cell
model	a list of nnet objects describing logistic regression models (if `model_out=TRUE`)

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	**(required)**
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
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 `NULL` 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	`nnet` object describing logistic regression model (if `model_out=TRUE`)

Function: `capacity_klogreg_algorithm()`
Implements algorithm to estimate channel capacity using `glmnet` package

Arguments

name	description	default
dataMatrix	numeric matrix with columns treated as explanatory variables (output, Y, of the channel)	**(required)**
dataSignal	factor vector with inputs (X) of the channel length must be equal to the number of rows of dataMatrix	**(required)**
cv_core_num	the number of cores to use in parallel computing of `glmnet package`	1
lambda_num	is the lambda parameter as in `glmnet` package	10
model_out	logical indicating if the model object should be returned	`TRUE`
cc_maxit	the maximum number of iteration to optimise channel capacity	100

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	`glmnet` object describing logistic regression model (if `model_out=TRUE`)

Function: `capacity_logreg_testing()`
Performs diagnostic procedures

Arguments

name	description	default
data	data frame with input (X) and output (Y) values in separate columns	**(required)**
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
output_path	directory in which result and graphs will be saved	**(required)**
TestingSeed	the seed of random number generator to be used in diagnostics	1234
testing_cores	number of cores to use in parallel computing in diagnostics	1
boot_num	the number of bootstrap tests to be performed (used if `testing=TRUE`)	10
boot_prob	the proportion of data to be used in bootstrap (used if `testing=TRUE`)	0.8
traintest_num	the number of over-fitting tests to be performed (used if `testing=TRUE`)	10
partition_trainfrac	the proportion of data to be used as a training dataset (used if `testing=TRUE`)	0.6
side_variables	an optional character vector indicating names of columns in dataRaw with side variables, if `NULL` no side variables are included in estimation	`NULL`
resamp_num	is the number of resmapling tests to be performed (used if `testing=TRUE`)	10
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`
glmnet_algorithm	logical indicating if the `glmnet` package should be used	`FALSE`
dataMatrix	numeric matrix with columns treated as explanatory variables in logistic model (used if `glmnet_algorithm=TRUE`)	`NULL`
glmnet_cores	the number of cores to use in parallel computing of `glmnet package` (used if `glmnet_algorithm=TRUE`)	1
glmnet_lambdanum	is the lambda parameter as in `glmnet` package (used if `glmnet_algorithm=TRUE`)	10

Values – a list with elements

bootstrap	list of size `boot_num`, where each element is the returned value of `capacity_logreg_algorithm()` from a single run of bootstrap	
traintest	list of size `traintest_num`, where each element is the returned value of `capacity_logreg_algorithm()` from a single run of over-fitting test	
resamplingMorph	list of size `resamp_num`, where each element is the returned value of `capacity_logreg_algorithm()` from a single run of resampling test (used if `side_variables` is not `NULL`)	
bootResampMorph	list of size `resamp_num`, where each element is the returned value of `capacity_logreg_algorithm()` from a single run of resampling test II (used if `side_variables` is not `NULL`)	

Function: `capacity_output_graphs()`
Generates exploratory graphs

Arguments

name	description	default
data	data frame with input (X) and output (Y) values in separate columns	**(required)**
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
output_path	directory in which result and graphs will be saved	**(required)**
cc_output	logical indicating if preprocessing (centering and scaling)	`TRUE`
height	the basic dimnesion of plots (height)	4
width	the basic dimnesions of plots (width)	6

Values – a list with elements

name	description	
1	A comprehensive summary plot	
2	Input-Output relation	
3	Boxplots of data	
4	Violin plots of data	
5	Histograms of data	
6	Boxplot of side variables in data	
7	Capacity results	
8	Density plots	
9	A simple summary plot	

Function: `formula_generator()`

Generates a formula object based on input and output specification

Arguments		
name	description	default
signal	character indicating a name of column of dataRaw with input (X)	**(required)**
response	character vector indicating names of columns of dataRaw with measurements of outputs (Y)	**(required)**
side_variables	an optional character vector indicating names of columns in dataRaw with side variables, if `NULL` no side variables are included in estimation	`NULL`

Values – a list with elements		
name	description	
formula_string	character object that includes a formula syntax to use in logistic model	`NULL`

Function: `sampling_bootstrap()`, `sampling_partition()`, `sampling_shuffle()`

Used to generate subsets of data to use in diagnostic procedures

Arguments		
name	description	default
data	data.frame to be resampled	**(required)**
dataDiv	character indicating column of data, with respect to which split the data; only in `sampling_bootstrap()` and `sampling_partition()`	**(required)**
prob	the of data that should be sampled from the whole dataset; only in `sampling_bootstrap()`	**(required)**
partition_trainfrac	the proportion of data to be used as a training dataset; only in `sampling_partition()`	**(required)**
side_variables	vector of characters indicating columns of data the will be reshuffled; only in `sampling_shuffle()`	**(required)**
Values – a data.frame with the same structure as initial data object		

Function: `theme_publ()`

Changes the visual elements of ggplot object

Arguments		
name	description	default
version	possible values: 1,2,3. Selects different coloring and presentation options	1
base_size	the size of font to use in graph	12
base_family	the type of font to use in graph	sans
Values – a ggplot theme object		