# Short user guide for SBM toolbox

## Introduction

The purpose of this project is to provide a platform for high-performance Bayesian block-modelling of large-scaled complex networks. *Implementation of the toolbox is still in progress*, but at the conclusion aims at supporting appropriate Bayesian parametric and non-parametric clustering methods for the common network topologies (weighted/unweighted, directed/undirected, bipartite/unipartite).

## Core functionality

The purpose of the application is to provide a platform for efficient, high-performance Bayesian stochastic blockmodelling, to infer parcellations from complex large-scaled networks. The application utilizes an MCMC sampling procedure for inferring the model parameters, where the clustering is inferred using a combination of full and restricted Gibbs sampling while the hyper-parameters are inferred using a Metropolis-Hastings random walking procedure.

## Compilation and installation

The SBMtool is written in high-performance C++, and must be compiled using a C++ compiler that supports C++11. The toolbox is provided with a precompiled version of the application, compiled using the **gcc 4.8.1** compiler, for **Linux x86/64** bit.

For guided compilation, the source code is provided together with a make file. To use this, the make tool must be available. This is build into most Linux distributions. Alternatively, all source files can be copied and compiled through an IDE. In all cases, the code utilizes certain Boost libraries that must be installed and available (see http://www.boost.org/ for details for installing Boost).

The optional graphical user interface is written in Java using the Swing toolkit. The GUI is both provided as a precompiled **.jar** file and as source code. Hence a Java runtime environment (JRE) must be installed and available to use the GUI, while a Java compiler must be available to modify and compile the source code (see https://www.java.com/ and https://www.java.com/en/download/ for details).

## Data format

As input the application expect two files, that must be formatted correctly: the *runscript* defines the model and sampling strategy, and the *network* file which is linked to within the runscript and contains the network from which the parcellation is inferred.

The format of the output files depends on the particular model and sampling procedure, but in generally the application writes a single file for each inferred parameter and a single file that describes the values of the likelihood, prior and posterior at different stages of the inference procedure.

Please note that percentage symbols (%) can be used in the runscript and output files to indicate that

the rest of a line is a comment and not part of the data.

### *Runscript file*

The filename for the runscript is given to the application as argument. This must be in a particular format, such that it can be correctly parsed by the application.

### *Network file*

The network file defines all links in the examined network, and must be provided as an ascii textfile in a particular format. Each line in the file represents a single link in the network, by three (two for binary networks) whitespace separated values: [ *source* *target* *weight* ].

Each line can hence be considered a three element tuple, that defines a single link in the network, linking from the node *source* to the node *target* with weight *weight*.

*source* and *target* must both be unique consecutive integer values for all nodes, such that the first node is assigned the value 1. *weight* depends on the type of link, such that integer values are used for counts and categories.

If the network is undirected, only lines where *source* < *target* are read (corresponding to the upper triangular part of the adjacency matrix) and considered to represent the link in both directions.

### *Monitor output*

The application outputs a single file to capture the value of the likelihood, prior and posterior at different iterations of the sampling procedure. Each line in the file represents a given iteration, such that the first (whitespace separated) value indicates the iteration, while the remaining values indicate the logarithm of the likelihood, prior and posterior.

### *Parameter output*

How often output are written can be defined in the runscript (determined by the scheduler), but the default is after each sweep of the entire sampling procedure.
The application writes in a single file for each parameter, such that each line describes the parameter value at a given iteration of the sampling procedure. The line can contain multiple values which are all whitespace separated. The first value always indicate the iteration of the sampling procedure for which the remaining values describe the parameter at.
A clustering will be described by multiple integer values, one for each node in the network, indicating the cluster that the node is assigned to, such that the first parameter value describes the cluster assignment of the first node and so forth.

## Graphical User Interface

The Graphical User Interface (GUI) has two main purposes:

1) In a clear way it allows the user to get an overview of the implemented models, samplers and inference procedures and explore the different possible user defined options for each element. This information can also be obtained through the command line interface, though the GUI presents it an easy, interactive format.

2) Interactively, it can aid the user in writing error-free runscripts.

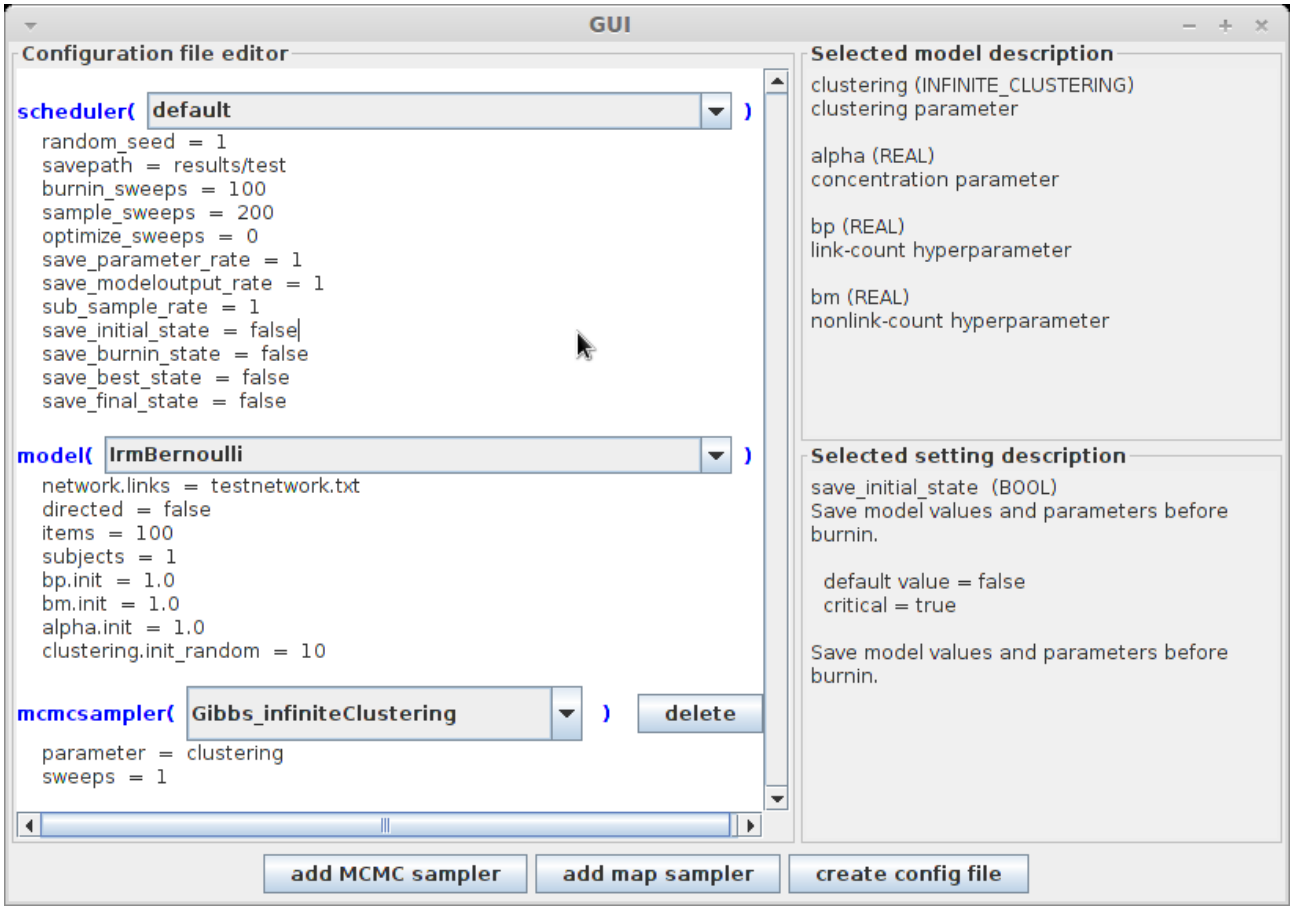


**Figure 1**, *The graphical user interface*

Figure 1 shows a screenshot of the light-weight GUI. The main window consists of three panels and some buttons.
The ***Configuration file editor*** panel is in principal an enhanced text editor for writing a runscript. Using the drop-down lists, the user can inspect, chose and change the scheduler, model and sampling procedure. When an entry in a drop-down list is chosen, all settings for the entry is added to the runscript, with default values that and can be manually changed by the user.
The ***delete*** button next to a map or mcmc procedure will remove the sampler and all associated settings from the runscript. The ***add MCMC sampler*** and ***add map sampler*** will respectively add a mcmc or map sampler ently to the end of the runscript (adding a drop-down list, from which a sampling procedure can be chosen). The ***create config file*** button will pares and present the runscript in an un-enhanced text editor, from which the runscript can be copied or saved to be used as input for the SBM tool.
The ***Selected model description*** panel shows names, types and descriptions of the parameters of the currently selected models. For each parameter, the type dictates what samplers that are appropriate. The name of a parameter is given as setting for the sampler, in order to sample that particular parameter. In the example in Figure 1, the Gibbs sampler for the parametertype 'infinite clustering' is chosen, with the parameter setting set to the name of the parameter (here, simply being 'clustering').
The ***Selected setting description*** panel shows information about the setting that is currently being edited (active line in the Confuguration panel).

## Comments

For comments or suggestions, you are welcome to email me at *kjal(-a-)dtu(dot)dk*.

This document was last updated 04/19/17