Graph_Sampler:

A Network Inference and Simulation Program

by Frédéric Y. Bois

User's Manual, software version 3.0.0

contact:
Frederic Bois
fbois@member.fsf.org

Table of Contents

1 Software and Documentation Licenses

1.1 Software license

Graph_Sampler is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

1.2 Documentation license

The GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in

another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

2 Overview

Graph_Sampler is an inference and simulation tool for networks (understood as graphs). It can *simulate* random graphs for general directed graphs (eventually cyclic) (see [Bibliographic References], page 27: Bois & Gayraud 2015; Datta *et al.* 2017) or for directed acyclic graphs (Bayesian networks). The graphs are generated by Markov chain Monte Carlo (MCMC) simulations and their structure can be specified to follow probabilistic properties through the use of prior distributions. Graph_sampler lets you also *infer* about graph structure through the joint use of priors and data about node values (via a likelihood function).

2.1 General procedure

You write an input files and run compiled `graph_sampler` program. The input files specifies the kind of graph to simulate, some simulation parameters and output options, the priors you want, the eventual data and their likelihood (see Chapter 4 [Running Graph_Sampler], page 13). The simulation output is written to standard ASCII files. After simulations, you can also check the convergence of the MCMC sampling.

No knowledge of computer programming is required, unless you want to tailor the program to special needs (in which case you may want to contact us).

2.2 New features

- Version 1.0.0 provides basic sampling of general graphs and structural inference for Bayesian networks, see [Bibliographic References], page 27: Bois & Gayraud 2015; Datta *et al.* 2017.
- Version 2.0.0 implements missing data imputation, dynamic Bayesian networks, tempered MCMC sampling, the edge count prior, and various routines using the GNU Scientific Library (gsl).
- Version 3.0.0 implements mainly Bayesian inference for general (cyclic or acyclic) graphs, the associated constant-gamma data model, priors on strongly connected components (cycles), posterior convergence analyses, and tests for checking the software during or after installation.

3 Installation

3.1 System requirements

Graph_Sampler is written in ANSI-standard C language. We are distributing the source code and you should be able to compile it for any system, provided you have an ANSI C compliant compiler.

On any system we recommend the GNU `gcc` compiler (freeware). An automated compilation script (called `Makefile`) is provided and can be used if the standard command `make` is available to you. We also recommend that you install the GNU Scientific Library prior to installing *Graph_Sampler*, so that it can make used of the Mersenne twister and Tausworthe pseudo-random number generators it provides.

If you want to modify the input file parser you will need `lex` and `yacc` (that is for experienced C programmers).

3.2 Distribution

Graph_Sampler source code is available on Internet through:

`https://sites.google.com/site/utcchairmmbsptp/software`.

3.3 Machine-specific installation

3.3.1 Unix and GNU/Linux operating systems

To install on a Unix or GNU/Linux machine, download (in binary mode) the distributed archive file to your machine. Place it in a directory where there is no existing `graph_sampler` subdirectory that could be erased (make sure you check that). Decompress the archive with GNU gunzip (`gunzip <archive-name>.tar.gz`). Untar the decompressed archive with tar (`tar xf <archive-name>.tar`) (do `man tar` for further help). Many other archiving tools can be used in place of gunzip and tar. Move to the `graph_sampler` directory just created and issue the following commands:

```
make
```

This command compiles the `graph_sampler` program.

If you do not have the GNU Scientific Library installed, or do not want to use it, you should compile with the command `make -f Makefile_no_gsl`.

You can also compile this manual as an info file with the command `make info` or as an html file with `make html`.

To check the software installation, use the command `make test`. It runs a series of tests, which should all pass. The test files are included in the sub-folder `tests/` and can also be used as examples or templates for your own analyses.

3.3.2 Other operating systems

Under other operating systems (Windows, etc.) or if everything else fails you should be able to both uncompress and untar the archive with widely distributed archiving tools. Refer to the documentation of your C compiler to create an executable file from the source code files provided.

You are now ready to use *Graph_Sampler*.

4 Running Graph_Sampler

After having compiled `graph_sampler`, you are ready to run it. For this you need to write an input file. This chapter explains how to write such files with the proper syntax.

In Unix the command-line syntax to run that executable is simply:

```
graph_sampler [input-file [output-prefix]]
```

where the brackets indicate optional arguments. If no input file or/and output prefix are specified, the program will use defaults. The default input file name is `script.txt`, the output files created depend on flag you set in the input file (see below) and their name is printed on exit. The default output file names are `best_graph.out`, `graph_samples.out`, `degree_count.out`, `motifs_count.out`, `edge_p.out`, `results_mcmc.bin` and `missing_data.out` (the latter is created only if "`NaN`" data are specified). If you only specify an input file name, the output file names will still be the default ones. If you specify both an input file name and an output prefix, the default output file names will be prefixed by it (*i.e.*, with the prefix `my` the edge probabilities output file will be named `my_edge_p.out`).

When the program starts, it announces which model description file was used to create it. While the input file is read or while simulations are running, some informations will be printed on your computer screen. They can help you check that the input file is correctly interpreted and that the program runs as it should. *Graph_Sampler* can also post error messages, which should be self-explanatory. Where appropriate, they show the line number in the input file where the error occurred.

The program ends (if everything went fine) by giving you the name of the output file generated. If you want to run the program in batch mode (in the background), you may want to redirect the screen output and error messages; refer for this to the `man` pages for your command shell.

4.1 Input file syntax

An input files specifies the kind of graph to simulate, some simulation parameters and output options, the priors you want, the eventual data and their likelihood. All that is done through the specification of predefined variables, using some keywords, user defined variables, numbers and operators.

A *Graph_Sampler* input file is a text (ASCII) file that obeys a relatively simple syntax:

- An input file can contain statements, matrix definitions and comments.
- Statements and matrix definitions must end with ';' and can span several lines. They can be placed in any order, except that matrix sizes (defined by specific predefined variables) must be defined before the corresponding matrix definition.
- Comments start with the pound sign `#` and go up to the end of line. They are ignored. Example:

  ```
  # this is a comment, comments are useful
  ```

- Variables are user defined symbols whose name must start with a letter, followed eventually by other letters, numbers and `_`. Letters can be upper-case or lower-case. Variable names are case sensitive. Example:

  ```
  Xa_2
  ```

  Note that unassigned variables have a default value of zero.

- Predefined variables are reserved names that *Graph_Sampler* understands. An exhaustive list of the predefined variables is given in the next section. Predefined variables not explicitly assigned have (hopefully useful) default values which may differ from zero (refer to their description, below). Example:

  ```
  n_nodes
  ```

- Expressions are numbers (as in C) or formulae including numbers and/or user defined variables, operators ('+', '-', '*', '/') or parentheses. Formulae are computed at they appear, with usual precedences. The division is always a real division (not an integer division). Example:

  ```
  (5 + 6) * (3.4 / 1.1E-8) + Xa_2;
  ```

- Statements are in the format:

  `<variable> = <expression>;`.

  Example:

  ```
  X_a2 = 5000;
  n_nodes = 6 * Xa_2;
  ```

- White space consist of `space`, `tab` or `carriage return`. Several white space characers in a row are treated as just one white space. Example:

  ```
  Xa_2 =    (2   + 3)    /
            (25. - 5.76);
  ```

- Lists are comma separated lists of expressions. Example:

  ```
  1, 2, 2+1, 2*2, 5, Xa_2
  ```

- Array definitions can only be used with predefined variables at the lefthand side. They are in the forms:`<variable> = array{<list of expressions>};`.

  The term `array` is a reserved keyword (see the list of those keywords below). Example:

  ```
  n_data_levels = array{2, 2, 1+1};
  ```

- Matrix definitions can only be used with predefined variables at the lefthand side. They are in the forms:

  `<variable> = matrix{<list of expressions>};`.

  The term `matrix` is a reserved keyword (see the list of those keywords below). Example:

  ```
  data = matrix{
  1,   2,  2+1,
  2*2, 5,  Xa_2};
  ```

  That is the general form. Some matrices can accept keywords such as `empty`, `full`, `equanimous`, or `random` instead of a list of expression inside the curly braces (see the specification of each predefined matrix, below).

4.2 Predefined variables

Here are, grouped by topic, the predefined variables that *Graph_Sampler* understands (they may have different synomyms, for example a long and a short form, separed here by commas):

4.2.1 Global control variables

autocycle

The `autocycle` variable should be set to 1 (`true`) to allow edges from a node to itself, and to 0 (`false`) otherwise. Its default value is `false`. Setting it to `true` is incompatible with specifying `bayesian_network` to `true` (loops are not allowed in such networks).

bayesian_network

The predefined variable `bayesian_network` indicate whether the graphs to sample are Bayesian networks (in that case it should be set to 1 or `true`) or general directed graphs (in which case it should set to 0 or `false`). General directed graphs can only be simulated on the basis of priors. For Bayesian networks both simulation and structural inference can be performed. Note that `bayesian_network` is incompatible with `autocycle` set to `true`. The default value for `bayesian_network` is `false`.

Example:

```
bayesian_network = true; # bayesian_network = 1 would also work
```

dynamic_bayesian_network

If `dynamic_bayesian_network` is set to to 1 or `true` the graphs sampled are dynamic Bayesian networks (DBNs). Both simulation and structural inference can be performed. In DBNs, the data are supposed to be collected at different discrete times, and the node states (values) at a given time can influence the nodes values at subsequent times. That allows the modeling of loops (*e.g.*, a node at time t can be its own parent at time t+1) (see [Bibliographic References], page 27: Husmeier 2003). Currently, in *graph_sampler*, edges can only connect nodes from one time to the next (no connection to node values at the same time or times ulterior to the next). The pattern of edges from one time to the next is also constant and valid for all time pairs (the dependence structure is not allowed to change with time). The initial adjacency matrix specified, together with the priors on edges refer to edges between subsequent times and do not need to respect acyclicity (again, for example, a 1 on the diagonal of the adjacency matrix means that the corresponding node at time t is the parent of itself, but at time t+1). Despite the free structure of the adjacency matrix sampled, acyclicity is always maintained in such DBNs, and the adjacency matrices given just need to be "unrolled" in time. Note that `autocycle` can be set to either `true` or `false` if `dynamic_bayesian_network` is set to `true`.The default value for `dynamic_bayesian_network` is `false`.

hypergraph

If `hypergraph` is set to to 1 or `true` the graphs sampled are hypergraphs in which strongly connected components ("cycles" or "loops") are collapsed into multivariate nodes (see [Bibliographic References], page 27: Wiecek submitted). Only structural inference can be performed in this case. The data are specified as in a standard Bayesian network, but the global graph can contain cycles. Note that `autocycle` must be set to either `true` if `hypergraph` is set to `true`. The default value for `hypergraph` is `false`.

n_nodes

The number of nodes in the network considered is specified by setting `n_nodes` to an integer (not long integer) value. `n_nodes` must be set before the initial adjacency or prior on edges'

probability matrices are defined. The default value for `n_nodes` is 0, which raises an error message, because `n_nodes` should be set to a meaningful value.

initial_adjacency

The starting value of the graph adjacency matrix is defined by setting `initial_adjacency`, a square matrix of dimension `n_nodes`. Matrix elements should be either 0 or 1. Element [*i*,*j*] is set to 1 if an edge (link) goes from node *i* to node *j*. Setting it to 0 indicates no edge between the two nodes.

Example:

```
n_nodes = 3;
initial_adjacency = matrix {0, 0, 0,
                            1, 0, 0,
                            1, 0, 0};
```

`initial_adjacency` definition can also use an extended syntax:

`initial_adjacency = matrix{empty | full | random};`.

were "`|`" means "or".

- If the keyword `empty` is used, all elements will be set to zero.
- If `full` is used all elements will be set to 1 when `bayesian_network` is `false`. If `bayesian_network` is `true` the diagonal elements will be set to zero and the others to 1. If you want it to work with Bayesian networks, you should set `bayesian_network` to `true` before defining `initial_adjacency`, because its default value is `false`.
- If `random` is used all elements will be set randomly to 0 or 1 (with equal probability) when `bayesian_network` is `false`. If `bayesian_network` is `true` the diagonal elements will be set to zero and the others to 0 or 1. If you want it to work with Bayesian networks, you should set `bayesian_network` to `true` before defining `initial_adjacency`, because its default value is `false`.

n_runs

The total number of iterations to be performed by the MCMC sampler is specified by setting `n_runs` to an integer or a floating point number inferior to the maximum unsigned long integer value on your machine (typically 18.446.744.073.709.551.615 on a 64 bits machine). Its default value is 1000000000 (a billion).

n_burn_in

A certain number of "burn-in" iterations can be specified by setting `n_burn_in` to a long integer value. In that case the MCMC chain recording, and computation of summary outputs (such as the edge probabilities) starts only after `n_burn_in` iteration. Its default value is zero. This is typically used to discard the part of the MCMC chain that is not at equilibrium. However, checking that equilibrium is attained is best done, in our opinion, by running multiple independent chains and using Gelman and Rubin $\hat{R}$ diagnostic (see [Convergence analysis], page 24).

perk_scale

Tempered MCMC sampling is perfomed if a `perk_scale` is specified (Geyer and Thompson, Journal of the American Statistical Association, 1995, 90:909-920). The `perk_scale` defines a set of inverse temperatures (between 0 and 1) to be used in tempered MCMC.

```
perk_scale = array {0.1, 0.5, 0.9, 1};
```

This option slows effective sampling (which occurs at perk equal to 1) by a factor equal to the number of perk factors provided, but it improves mixing and convergence dramatically. Perks (inverse temperatures) are sampled at the start of each adjacency matrix updating. Therefore if a perk of zero is specified, the Markov chain should regenerate when it hits this perk (which corresponds to an infinite temperature). Samples obtained at perk 1.0 between two random hits of perk zero should therefore be from the exact target distribution (*i.e.*, obtained at convergence). If recording the chain is requested (`save_chain` set to `true`) perks are output to the file `inverse_temperatures.out`. You can use it to trace back which graph samples were obtained at which temperature.

random_generator

If you have linked *Graph_Sampler* with GNU Scientific Library (gls) you can choose between two extremely long period random number generators provided by the library: either the "Mersenne twister" generator (gsl_rng_mt19937) or the "Tausworthe generator" (gsl_rng_taus2). To that effect you can set the variable `random_generator` to either the `mersenne_twister` keyword or the `tausworthe` keyword. By default the Mersenne twister is used.

Example:

```
random_generator = tausworthe;
```

If you have have compiled *Graph_Sampler* with the NO_LIBGSL option, the GNU Scientific Library is not available. In that case the Park and Miller's "minimal standard" MINSTD generator (a good one though) is used instead. Setting the `random_generator` variable is ignored in that case.

random_seed

The starting value of the pseudo-random generator `random_seed` can be explicitly set to any real or integer number superior to zero. That allows repeating exactly the same sequence random numbers. That is required to generate different chains for the same problem in order to check the convergence of the MCMC simulations. If it is not set by the user, `random_seed` has a default value of 314159265.3589793.

Example:

```
random_seed = 123.456;
```

4.2.2 Variables specifying priors

hyper_pB

The matrix `hyper_pB` is a square matrix of dimension `n_nodes` which specifies a prior distribution on edge probabilities. Each element [*i*,*j*] of `hyper_pB` is the parameter p (a real of double format) of a Bernoulli distribution for the presence of an edge from node i to node j. In the case of Bayesian networks, p values should be 0 on the first diagonal.

Example:

```
bayesian_network = true;
hyper_pB = matrix {0,   0.1, 0.1,
                   0.9, 0,   0.1,
                   0.9, 0.1, 0  };
```

`hyper_pB` definition can also use an extended syntax:

`hyper_pB = matrix{equanimous};`.

If the keyword `equanimous` is used, all elements of hyper_pB will be set to 0.5, except when `bayesian_network` has been defined as `true`, or `autocycle` is `true`, or `hypergraph` is `true`, in which cases the diagonal elements will be set to zero.

Finally, you can also read the `hyper_pB` matrix from a file, using:

```
hyper_pB = import_file {"filename"};
```

where `filename` is the valid name of an existing file containing the probability values, separated by white spaces or tabs (but not commas). That file cannot have extensions `.bin` or `.out`, as those are reserved.

Internally, `hyper_pB` is always used. If it is not defined by the user, p values will default to 0.5 (with zeroes on the diagonal if `bayesian_network` is `true`), so that the prior is neutral (equal probability for the absence or presence of any edge). The use of the keyword `equanimous` is equivalent, with the advantage of being explicit.

In the case of a Bayesian network, nodes which have been assigned a zero probability of having parents (a column of zero in the `hyper_pB` matrix) are understood to be special "control" nodes for which the likelihood will not be computed. Such nodes will typically correspond to experimental design variables. Their likelihood is not computed. They condition the likelihood of their eventual children node and then take the values assigned to them in the input file (in which case the "data" are rather forcing values than actual observations).

concordance_prior

The flag `concordance_prior` set to 1 or `true` indicates that a concordance prior should be used (in addition to the baseline Bernoulli prior on individual edges). By default `concordance_prior` is `false`. A concordance prior is an unnormalized score of the edge-wise difference between a reference adjacency matrix and the matrix being examined (see below `edge_requirements`). Beware that you should probably not used it in conjunction with an informative Bernoulli prior on edges, since both priors specify (explicitly in the case of Bernoulli) individual edge probabilies. Leaving the Bernoulli prior unspecified will be fine in that case as it will be assigned a non-informative default value.

edge_requirements

The matrix `edge_requirements` is a square matrix of dimension `n_nodes` which specifies the concordance between the edges of a reference adjacency matrix and the current one. Each element [i,j] of `edge_requirements` can take a value of 1, -1, or 0.

- The value 1 indicates that an edge from node i to node j is desired (rather than strictly required). The presence of that edge in the adjacency matrix evaluated raises its score by `lambda_concordance` (see below).

- The value -1 indicates that an edge from node i to node j is not desired. The presence of that edge in the adjacency matrix evaluated lowers its score. In the case of Bayesian networks, diagonal values should be -1, otherwise Hell might break loose.
- A value of 0 indicates no preference: the score is unaffected by the presence of absence of an edge from node i to node j. The presence of that edge in the adjacency matrix evaluated raises its score.

Example:

```
concordance_prior = true;
edge_requirements = matrix {-1, -1,  0,
                            1, -1, -1,
                            1, -1, -1};
```

By default, all elements of `edge_requirements` will be set to 0 when `bayesian_network` is `false`. If `bayesian_network` is `true` the diagonal elements will be set to -1 and the others to 0. If you want it to work with Bayesian networks, you should set `bayesian_network` before defining `edge_requirements`, because its default value is `false`.

lambda_concordance

The parameter `lambda_concordance` is used to weight the differences between the reference adjacency matrix and the current adjacency matrix when `concordance_prior` is `true`. It should be set to a double (typically superior to zero). Its default value is 1.

degree_prior

The flag `degree_prior` set to 1 or `true` indicates that a power law prior is placed on the distribution of the nodes' degrees (the number of incoming and outgoing edges for a given node) (see [Bibliographic References], page 27: Bois & Gayraud 2015). The power law parameter`gamma_degree` (see next) has default value 1. This prior comes in addition to the baseline Bernoulli prior on individual edges. By default it is `false`.

gamma_degree

If `degree_prior` is `true`, `gamma_degree` specifies the parameter of the exponential prior on degree counts. It should be set to a double (typically superior to zero). Its default value is 1.

edge_count_prior

The variable `edge_count_prior` can be set to a number between 0 and the maximum number of edges possible to impose a binomial prior (with parameters `n_nodes` squared and edge_count_prior / (`n_nodes` squared)) on the total number of edges in the graph. The maximum number of edges is `n_nodes` squared, in a general graph, and `n_nodes` times (`n_nodes` - 1) in a Bayesian network. The number given should be the *a priori* expected value for the edge count. The variable `n_nodes` must be set before setting `edge_count_prior`.

Example:

```
n_nodes = 20;
edge_count_prior = 50;
```

motif_prior

The flag `motif_prior` set to 1 or `true` indicates that a beta-binomial prior is placed on the count of triangular feed-forward and feedback loops in the network (see [Bibliographic References], page 27: Bois & Gayraud 2015). It comes in addition to the baseline Bernoulli prior on individual edges and is incompatible with Bayesian networks or dynamic Bayesian networks (an error message will be issued). By default it is `false`.

alpha_motif

If `motif_prior` is `true`, `alpha_motif` specifies the first parameter of the beta-binomial prior on loops' counts. It should be set to an integer superior to zero. Its default value is 1.

beta_motif

If `motif_prior` is `true`, `beta_motif` specifies the second parameter of the beta-binomial prior on loops' counts. It should be set to an integer superior to zero. Its default value is 1.

scc_prior

The flag `scc_prior` set to 1 or `true` indicates that priors are placed on either or both the number and size of the graph's strongly connected components (SCCs) (see next). By default, only a prior on the number of SCCs is turned on. Those SCC priors come in addition to the baseline Bernoulli prior on individual edges and is incompatible with Bayesian networks (an error message will be issued). By default it is `false`.

lambda_scc

If `scc_prior` is `true`, `lambda_scc` specifies the rate parameter of Poisson prior on the number of non-trivial SCCs (SCCs of size larger than 1) in the graph. So it represents basically the expected a priori number of SCCs in the graph. It should be set to an integer superior to zero. Its default value is 1.

gamma_scc

If `scc_prior` is `true`, `gamma_scc` specifies the power parameter of a power law prior on the size of the graph's SCCs. It should be set to an integer equal or superior to zero. Larger values penalize large SCCs. Its default value is zero.

maximum_scc_size

The variable `maximum_scc_size` can be set to an integer to place an upper limit on the size of the SCCs present in the sampled graph. By default the limit is the number of nodes.

4.2.3 Variables specifying data and likelihood

n_data

If `bayesian_network` is `true`, data can be input to infer on the probabiliy of the presence of edges on the basis of priors and data likelihood, in a fully Bayesian framework. The predefined variable `n_data` should be set to an integer equal to the number of data points per node. Its default is zero. If no data are provided while `bayesian_network` is `true`, simulations will proceed simply on the basis of priors distributions.

data

After `n_nodes`, `n_data` and a proper model have been defined, a data matrix can be defined (actually if `n_data` is different from zero, a data matrix must be defined). `data` has no default value. The matrix of data should have n_nodes rows and each row should be a vector of `n_data` values (integers, doubles or "`NaN`" in the case of missing data). A warning is issued if a node has only missing data (that is likely to lead to identifiability problems).

Example:

```
n_nodes = 3;
bayesian_network = true;
n_data = 4;
data = matrix {1.1, 1.3, 1.4, 1.35,
               2.1, NaN, 2.5, 2.45,
               3.4, 3.6, 3.8, 3.85};
```

You can also read the data matrix from a file, using:

```
data = import_file {"filename"};
```

where `filename` is the valid name of an existing file containing the data values (or missing data coded as `NaN`), separated by white spaces or tabs (without commas). The data file cannot have extensions `.bin` or `.out`, those are reserved.

If missing data are entered (as "`NaN`"), they will be replaced in the computation by imputed values (imputation is performed by MCMC sampling). A sub-sample (at the moment a thousand) of the imputed data value is written to a text file (named `missing_data.out`)

likelihood

If data are supplied, their likelihood function must also be specified with the following syntax:

```
likelihood = dirichlet | normal_gamma | constant_gamma | zellner;
```

were "`|`" means "or".

- The keyword `dirichlet` should be used for discrete data only. It specifies a Dirichlet-multinomial model (See [Bibliographic References], page 27: Laskey & Myers 2003; Heckerman *et al.* 1994; Heckerman *et al.* 1995). In that case the data have to be coded by integers from zero to n. The number of levels for each node has to be specified using an [n_data_levels], page 22, declaration. The Dirichlet hyper-parameters are internally set to one, specify a uniform prior on configurations of parents for any node.
- The keyword `normal_gamma` specifies a vague normal-gamma prior for the regression parameters describing the dependance of children nodes with respect to their parents. Such a prior and model can be used for continous or discrete data. Its hyper-parameters [alpha_normal_gamma], page 22, and [beta_normal_gamma], page 22, can be set at user-defined values.
- The keyword `constant_gamma` specifies an improper constant-gamma prior for the regression parameters describing the dependance of children nodes with respect to their parents. Such a prior and model is used for hypergraphs and can be used in standard BNs for continous or discrete data. Its hyper-parameters [extra_df_wishart],

page 22, [scale_wishart_diagonal], page 22, [scale_wishart_offdiagonal], page 22, and [maximum_scc_size], page 22, can be set at user-defined values.

- The keyword `zellner` specifies a Zellner prior for the normal regression model (based on our experience, we do not recommend it though). One of its drawbacks is that any node cannot have more parents than there are data about it (arguably, an artificial constraint).Its hyper-parameter [gamma_zellner], page 23, can be set at a user-defined value.

n_data_levels

If a Dirichlet-multinomial model is used, discrete data have to be specified for each node. Such data have to be coded as integers from zero to n, n being the number of levels for a given node. Those levels are specified using the `n_data_levels` [array], page 25, declaration.

```
n_data_levels = array{2, 2, 3, 2, 4};
```

alpha_normal_gamma

The parameter `alpha_normal_gamma` corresponds to the prior shape of the Gamma distribution of the data precision in the Normal-Gamma regression model. It can be set to any positive floating point value. By default its value is 1.5, which works well in our hands. You may want to tailor it to your needs.

beta_normal_gamma

The parameter `beta_normal_gamma` corresponds to the prior rate of the Gamma distribution of the data precision in the Normal-Gamma regression model. It can be set to any positive floating point value. By default its value is 1000, which is rather vague and works well in our hands.

extra_df_wishart

The tuning parameter `extra_df_wishart` can be set to any positive integer. It is added to the number of nodes in a hypernode to define the degrees of freedom for the inverse-Wishart prior distribution of the regression parameter covariance matrix. By default its value is 1.

scale_wishart_diagonal

The tuning parameter `scale_wishart_diagonal` can be set to any positive floating point value. It defines the value of the diagonal elements of the scale matrix for the inverse-Wishart prior distribution of the regression parameter covariance matrix. By default its value is 1.0.

scale_wishart_offdiagonal

The tuning parameter `scale_wishart_offdiagonal` can be set to any positive floating point value. It defines the value of the off-diagonal elements of the scale matrix for the inverse-Wishart prior distribution of the regression parameter covariance matrix. By default its value is 0.

maximum_scc_size

The contraining parameter `maximum_scc_size` can be set to any positive floating point value. It defines the maximum allowed size of a strongly connected component in the

network (if hypergraphs are sampled). By default its value is n_nodes. It does not make sense to set it to a higher value...

`gamma_zellner`

The tuning parameter `gamma_zellner` can be set to any positive floating point value. When it is equal to the number of data points per node, the data and Zellner prior on the regression parameters have equal weight. By default its value is 1.0.

4.2.4 Variables specifying outputs

`save_chain`

The entire MCMC sampling chain can be saved (after the burnin period) in binary format to a file (named by default `results_mcmc.bin`) by setting `save_chain` to 1 or `true`. By default, the chain is not saved. Beware, MCMC chains can be very large, even though the recording format is very compact: `results_mcmc.bin` starts with the number of nodes in the graph (as a binary integer, *i.e.*, a byte), followed by the value of the adjacency matrix (n_nodes by n_nodes bits) at the end of burn-in period, followed by a two-bits encoding of the difference between successive adjacency matrices or end of chain. The chain saving algorithm assumes that the matrix is scanned systematically, and does not encode the location of changes. The `results_mcmc.bin` file can be used to recreate the successive adjacency matrices sampled or check convergence of the MCMC algorithm.

`n_saved_adjacency`

The user can request the output of a number `n_saved_adjacency` (integer) of randomly generated adjacency matrices. Those matrices are saved at regularly spaced iterations along the MCMC chain (after the burn-in period) in the file with defaut name `graph_samples.out` in text format, along with the logarithmes of the prior probability, data likelihood (if data were specified) and posterior probability. A value of 1 for element [*i*,*j*] (that is, on the ith row and jth column) indicates that node i is parent of node j. By default `n_saved_adjacency` is zero and no matrices are recorded.

`save_best_graph`

By setting `save_best_graph` to `true`, the user can request the output of the adjacency matrix of the graph having the highest posterior probablity among all random graphs generated after the burn-in period. That matrice is saved in the file with default name `best_graph.out` in text format, along with the logarithmes of its prior probability, data likelihood (if data were specified) and posterior probability. By default `save_best_graph` is `false`.

`save_edge_probabilities`

Setting `save_edge_probabilities` to `true`, triggers the calculation and output of a matrix of the individual edge probabilities in the file `edge_p.out`, in text format. If requested, the edge probabilities are computed at each full update of the adjacency matrix (*i.e.*, when each element of the adjacency matrix has been visited. By default `save_edge_probabilities` is `false`.

`save_degree_counts`

Setting `save_degree_counts` to `true`, forces the output of a count of the nodes' degrees in the graphs sampled after the burn-in period to the file with default name `degree_count.out`, in text format. By default `save_degree_counts` is `false`.

`save_motifs_probabilies`

Setting `save_motifs_probabilies` to `true`, forces the output (to the file with default name `motifs_count.out`, in text format) of a count of triangular feed-forward and feedback loops in the graphs sampled after the burn-in period. By default `save_motifs_probabilies` is `false`.

4.2.5 Variables specifying posterior analyses

`convergence_check`

The command `convergence_check` instructs *Graph_Sampler* to check that a set of MCMC chains have converged, after running them, according to the following syntax (were "`|`" means "or"):

```
convergence_check = standard_w_chain | standard_w_edgeP |
                    incremental_w_chain;
```

Note that, when checking convergence, no sampling is made and that priors, likelihoods, data, and save statements are ignored. The three values (keywords) that `convergence_check` can take specify slightly different kinds of convergence analyses:

- The keyword `standard_w_chain` instructs *Graph_Sampler* to read a set of chain recording files (of format `results_mcmc.bin`) and perform a Gelman and Rubin convergence diagnostic on them (see [Bibliographic References], page 27: Gelman & Rubin 1992, and other relevant statistical literature). By default, the whole recorded chains are read, but a range of iterations can also be specified. The number of nodes in the graph is read from the files and needs to match across files.
- The keyword `standard_w_edgeP` asks *Graph_Sampler* to read a set of edge probabilities recording files (of format `edge_p.out`) and perform a Gelman and Rubin convergence diagnostic.This is much faster than using chain recording files, because edge probabilities files are much smaller, while containing all the needed information for computing convergence. However, in that case, the range options are not available (the whole chains used for computing edge probabilities are implicitly used) and you do need to specify the number of nodes in the graph using the `n_nodes` variable.
- The keyword `incremental_w_chain` will cause *Graph_Sampler* to read a set of chain recording files (of format `results_mcmc.bin`) and perform a Gelman and Rubin convergence diagnostic at each iteration. By default, the whole recorded chains are read, but a range of iterations can also be specified. This is the slowest procedure. The number of nodes in the graph is read from the files and needs to match across files.

File names need to be specified as an array, using a `file_names` specification:

```
file_names = array {"xa_results_mcmc.bin", "xb_results_mcmc.bin",
                    "xc_results_mcmc.bin"};
```

The variables `start_at_iter` and `end_at_iter` can be used when reading the chains to specify the (inclusive) range of iterations to check convergence on. In each case the results

are written to a file named `convergence.out`. This file can be very large in the case of incremental convergence analysis (a pair of numbers is written for each iteration read). In pathological cases where the chains are stuck, the convergence criterion can be infinite and is coded in the output file with the value -1.

4.3 Reserved keywords

The following keywords can be used in Graph_Sampler input files:

`array` keyword

This keyword is used for vector definition. Example:

```
n_data_levels = array {2, 2, 1+1};
```

`dirichlet` keyword

This keyword should be used in the case of discrete data, to specify a Dirichlet prior (syntax: `likelihood = dirichlet;`).

`empty` keyword

This keyword can be used to create an empty initial adjacency matrix (syntax: `matrix{empty};`).

`equanimous` keyword

This keyword can be used to create a prior matrix on edge probabilities (`hyper_pB`) with all elements set to 0.5 when `bayesian_network` is `false`. If `bayesian_network` is `true` the diagonal elements will be set to zero and the others to 0.5. (syntax: `matrix{equanimous};`).

`false` (or `False` or `FALSE`) keyword

This keyword is equivalent to zero and can be used when assigning variables.

`full` keyword

This keyword can be used to create a full initial adjacency matrix (all elements at 1, except a zeroed diagonal in Bayesian networks) (syntax: `matrix{full};`).

`import_file` keyword

This keyword should be used to indicate that hyper_pB or data should be imported from a text file (syntax: `hyper_pB = import_file{"filename"};` or `data = import_file{"filename"};`).

`incremental_w_chain` keyword

This keyword specifies an incremental Gelman and Rubin convergence analysis on a set of chain recording files.

`matrix` keyword

This keyword is used for matrix definition. Example:

```
data = matrix {1.1, 1.3, 1.4, 1.35,
               2.1, 2.4, 2.5, 2.45,
               3.4, 3.6, 3.8, 3.85};
```

`mersenne_twister` keyword

This keyword can be used to specify that the Mersenne twister pseudo-random numeber generator will be used (syntax: `random_generator = mersenne_twister;`).

`NaN` keyword

This keyword should be used instead of a number to indicate that a data value is missing. It can only be used within a data matrix.

`normal_gamma` keyword

This keyword can be used for discrete or (preferably) continuous data, to specify a Normal-Gamma prior (syntax: `likelihood = normal_gamma;`).

`constant_gamma` keyword

This keyword can be used for discrete or (preferable) continuous data, to specify a Constant-Gamma prior (syntax: `likelihood = constant_gamma;`).

`random` keyword

This keyword can be used to create a random initial adjacency matrix (all elements 0 or 1 at random, except a zeroed diagonal in Bayesian networks) (syntax: `matrix{random};`).

`standard_w_chain` keyword

This keyword specifies a simple Gelman and Rubin convergence analysis on a set of chain recording files.

`standard_w_edgeP` keyword

This keyword specifies a simple Gelman and Rubin convergence analysis on a set of edge probabilities recording files (of format `edge_p.out`).

`true` (or `True` or `TRUE`) keyword

This keyword is equivalent to 1 and can be used when assigning variables.

`tausworthe` keyword

This keyword can be used to specify that the Tausworthe pseudo-random numeber generator will be used (syntax: `random_generator = tausworthe;`).

`zellner` keyword

This keyword should be used in the case of continuous data, to specify a Zellner prior (syntax: `likelihood = zellner;`).

Bibliographic References

Barry T.M. (1996). Recommendations on the testing and use of pseudo-random number generators used in Monte Carlo analysis for risk assessment. Risk Analysis **16**:93-105.

Bernardo J.M. and Smith A.F.M. (1994). Bayesian Theory. Wiley, New York.

Bois F. and Gayraud G. (2015). Probabilistic generation of random networks taking into account information on motifs occurrence. Journal of Computational Biology, **22**:25-36, doi:10.1089/cmb.2014.0175, also at ArXiv:1311.6443 [q-bio.QM].

Datta S., Gayraud G., Leclerc E., Bois F. (2017). Graph_sampler: a simple tool for fully Bayesian analyses of DAG-models. Computational Statistics, **32**:691-716, doi: 10.1007/s00180-017-0719-1, also at ArXiv:1505.07228v2 [stat.CO].

Gelman A. and Rubin D.B. (1992). Inference from iterative simulation using multiple sequences (with discussion). Statistical Science **7**:457-511.

Heckerman et al. (1994). in Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, WA, p. 293-301. Morgan Kanfmann.

Heckerman et al. (1995). Machine Learning, 20, 197-243.

Husmeier D. (2003). Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic bayesian networks. Bioinformatics **19**:2271-82. doi:10.1093/bioinformatics/btg313.

Laskey and Myers (2003). Machine Learning, 50:175-196.

Wiecek W., Bois F., Datta S., Gayraud G. (submitted). Structure learning of causal graphs involving cycles.

Concept Index

S

T

V

Z