Cplint R Documentation

program version 1.0.0

Franco Masotti

Table of Contents

1 About

1.1 About

The purpose of this library is to provide an interface between the Cplint[1] suite for SWI Prolog[2] and R[3] to handle charts made with the ggplot2[4] package.

This means that most C3js[5] charting functions of Cplint are also available for R.

1.2 Terminology

In order to simplify the understanding of this document, a minimal set of terminology is necessary:

- Any code between `< >` represents pseudocode.
- The symbol `#` represents an unsigned integer or the `root` user, depending on the context.
- Optional code or parameters are encapsuled between `[ ]`.

[1] See item [**Cplint**] in Chapter 7 [References], page 12.
[2] See item [**SWI Prolog**] in Chapter 7 [References], page 12.
[3] See item [**R**] in Chapter 7 [References], page 12.
[4] See item [**ggplot2**] in Chapter 7 [References], page 12.
[5] See item [**C3js**] in Chapter 7 [References], page 12.

2 Installation

Cplint R is provided as part of the Cplint on SWISH[1] package. This is the recomended and simplest way to obtain a working installation without too much trouble.

You can also install it manually with `pack_install('cplint_r').` using the `swipl` command on UNIX-like systems. Unfortunately, doing this is not enough since you need to setup an R environment with the components described in the following section. For more information, have a look at the SWISH Installer Manual[2]

2.1 Dependencies

Cplint R has both Prolog and R dependencies:

2.1.1 Prolog

- Cplint
- Rserve Client[3]
- R SWISH library[4]

2.1.2 R

- ggplot2
- Rserve[5]
- gridExtra[6]

[1] See item [**Cplint on SWISH**] in Chapter 7 [References], page 12.
[2] See item [**SWISH Installer Manual**] in Chapter 7 [References], page 12.
[3] See item [**Rserve Client**] in Chapter 7 [References], page 12.
[4] See item [**R SWISH**] in Chapter 7 [References], page 12.
[5] See item [**Rserve**] in Chapter 7 [References], page 12.
[6] See item [**gridExtra**] in Chapter 7 [References], page 12.

3 Examples

3.1 Gaussian mean estimation

The following is the content of gauss_mean_est_R.pl.

```
/*
Posterior estimation in Bayesian models.
We are trying to estimate the true value of a Gaussian distributed random
variable, given some observed data.  The variance is known (2) and we
suppose that the mean has a Gaussian distribution with mean 1 and variance
5.  We take different measurement (e.g.  at different times), indexed
with an integer.
Given that we observe 9 and 8 at indexes 1 and 2, how does the distribution
of the random variable (value at index 0) changes with respect to the case of
no observations?
From
http://www.robots.ox.ac.uk/~fwood/anglican/examples/viewer/?worksheet=gaussian-posteriors
*/
:- use_module(library(mcintyre)).
:- use_module(library(cplint_r)).

:- mc.
:- begin_lpad.

value(I,X) :-
  mean(M),
  value(I,M,X).
% at time I we see X sampled from a Gaussian with mean M and variamce 2.0

mean(M): gaussian(M,1.0, 5.0).
% Gaussian distribution of the mean of the Gaussian of the variable

value(_,M,X): gaussian(X,M, 2.0).
% Gaussian distribution of the variable


:- end_lpad.

hist_uncond(Samples,NBins):-
  mc_sample_arg(value(0,X),Samples,X,L0),
  histogram_r(L0,NBins).
% plot an histogram of the density of the random variable before any
% observations by taking Samples samples and by dividing the domain
% in NBins bins

dens_lw(Samples):-
```

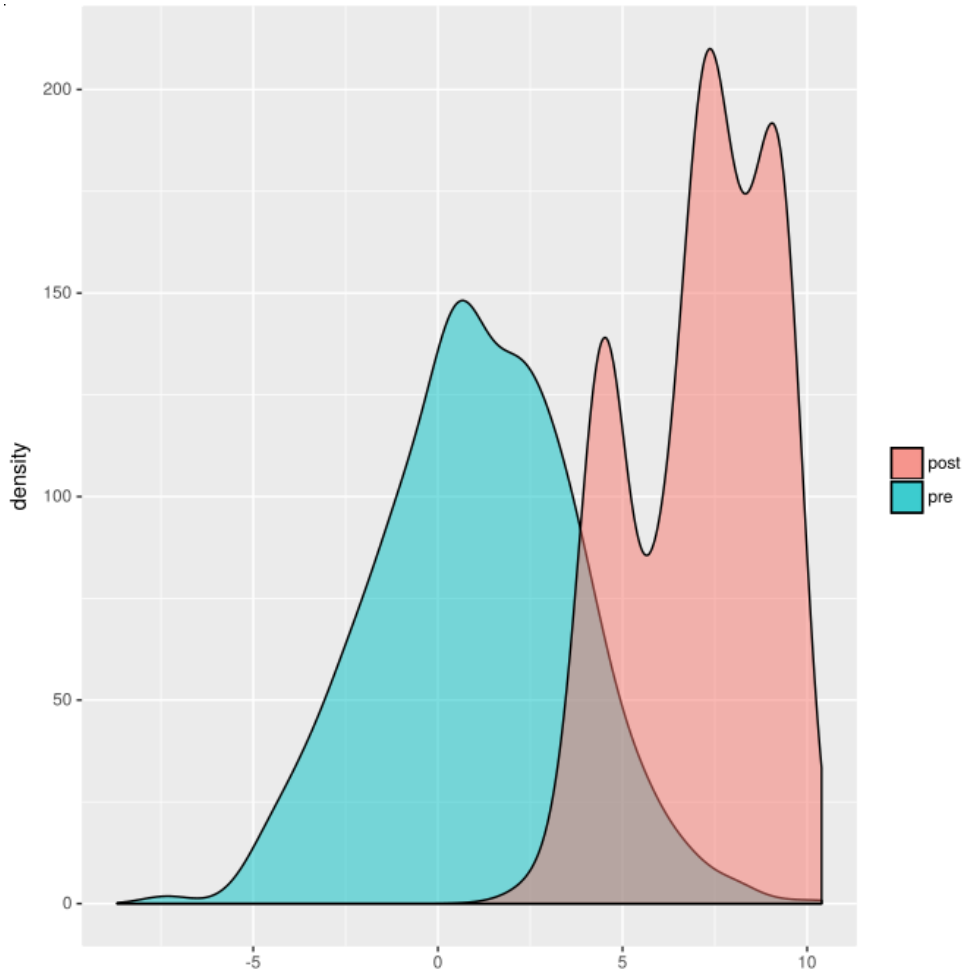
```
  mc_sample_arg(value(0,Y),Samples,Y,L0),
  mc_lw_sample_arg(value(0,X),(value(1,9),value(2,8)),Samples,X,L),
  densities_r(L0,L).
% plot the densities of the random variable before and after
% observing 9 and 8 by taking Samples samples.



/** <examples>
?- dens_lw(1000).
% plot the densities of the random variable before and after
% observing 9 and 8
?- hist_uncond(10000,40).
% plot an histogram of the density of the random variable before any
% observations
?-  mc_lw_expectation(value(0,X),(value(1,9),value(2,8)),1000,X,E).
% E = 7.166960047178755
?- mc_expectation(value(0,X),10000,X,E).
% E = 0.9698875384639362.

*/
```

Using `?- dens_lw(1000)` as query the following chart is produced:



3.2 Other examples

For other examples see all the files ending in "_R.pl" on `https://github.com/friguzzi/swish/tree/master/examples/inference`

4 Protocol

4.1 Interface

The following is a list of exported predicates available to the library users'.

Each predicate corresponds to one of the following categories

- Helper: specific of this library.
- Pita: part of Cplint.
- Mcintyre: part of Cplint.
- Auc: part of Cplint.

Each argument of a predicate correponds to a data type. See the SWI Prolog data types manual[1] and the Learn Prolog Now manual[2]. Have a look at the Cplint help manual[3] to learn in details about the functionality of each predicate.

`build_xy_list` (*+X:list*, *+Y:list*, *-Out:list*) *is det* [Helper]
: Given to lists X and Y build an output list *Out* in the form `[X1-Y1,X2-Y2,...,XN-YN]`.

`r_row` (*+X:atom*, *+Y:atom*, *-Out:atom*) *is det* [Helper]
: Given two atoms X and Y, build the term `r(X,Y)` in *Out*.

`get_set_from_xy_list` (*+L:list*, *-R:list*) *is det* [Helper]
: Given an input list L in the form `[X1-Y1,X2-Y2,...,XN-YN]`, transform it in an output list R in the form `[r(X1,Y1),r(X2,Y2),...,r(XN,YN)]`. This means that `R` will contain an X-Y relationship which can be then passed to an R data frame.

`prob_bar_r` (*:Query:atom*) *is nondet* [Pita]
: The predicate computes and plots the probability of *Query* as a bar chart with a bar for the probability of *Query* true and a bar for the probability of *Query* false. If *Query* is not ground, it returns in backtracking all ground instantiations of *Query* together with their probabilities.

`prob_bar_r` (*:Query:atom*, *:Evidence:atom*) *is nondet* [Pita]
: The predicate computes and plots the probability of *Query* given *Evidence* as a bar chart with a bar for the probability of *Query* true and a bar for the probability of *Query* false given *Evidence*. If *Query* / *Evidence* are not ground, it returns in backtracking all ground instantiations of *Query* / *Evidence* together with their probabilities.

`mc_prob_bar_r` (*:Query:atom*, *-Probability:dict*) *is det* [Mcintyre]
: See prob_bar_r/2.

---

[1] See item [**SWI Prolog data types**] in Chapter 7 [References], page 12.

[2] See item [**LPN**] in Chapter 7 [References], page 12.

[3] See item [**Cplint**] in Chapter 7 [References], page 12.

`mc_sample_bar_r` (*:Query:atom*, *+Samples:int*) *is det* [Mcintyre]
The predicate samples *Query* a number of *Samples* times and plots a bar chart with a bar for the number of successes and a bar for the number of failures. If *Query* is not ground, it considers it as an existential query.

`mc_sample_arg_bar_r` (*:Query:atom*, *+Samples:int*, *?Arg:var*) *is det* [Mcintyre]
The predicate samples *Query Samples* times. *Arg* should be a variable in *Query*. The predicate plots a bar chart with a bar for each possible value of L, the list of values of *Arg* for which *Query* succeeds in a world sampled at random. The size of the bar is the number of samples returning that list of values.

`mc_sample_arg_first_bar_r` (*:Query:atom*, *+Samples:int*, *?Arg:var*) [Mcintyre]
is det
The predicate samples *Query Samples* times. *Arg* should be a variable in *Query*. The predicate plots a bar chart with a bar for each value of *Arg* returned as a first answer by *Query* in a world sampled at random. The size of the bar is the number of samples that returned that value. The value is failure if the query fails.

`mc_rejection_sample_arg_bar_r` (*:Query:atom*, *:Evidence:atom*, [Mcintyre]
+Samples:int, *?Arg:var*) *is det*
The predicate calls `mc_rejection_sample_arg/5` and builds an R graph of the results. It plots a bar chart with a bar for each possible value of L, the list of values of *Arg* for which *Query* succeeds given that *Evidence* is true The size of the bar is the number of samples returning that list of values.

`mc_mh_sample_arg_bar_r` (*:Query:atom*, *:Evidence:atom*, [Mcintyre]
+Samples:int, *+Lag:int*, *?Arg:var*) *is det*
The predicate calls `mc_mh_sample_arg/6` and builds an R graph of the results. The predicate plots a bar chart with a bar for each possible value of L, the list of values of *Arg* for which *Query* succeeds in a world sampled at random. The size of the bar is the number of samples returning that list of values.

`mc_mh_sample_arg_bar_r` (*:Query:atom*, *:Evidence:atom*, [Mcintyre]
+Samples:int, *+Mix:int*, *+Lag:int*, *?Arg:var*) *is det*
The predicate calls `mc_mh_sample_arg/7` and builds an R graph of the results. The predicate plots a bar chart with a bar for each possible value of L, the list of values of *Arg* for which *Query* succeeds in a world sampled at random. The size of the bar is the number of samples returning that list of values.

`histogram_r` (*+List:list*, *+NBins:int*) *is det* [Mcintyre]
Draws a histogram of the samples in List dividing the domain in *NBins* bins. *List* must be a list of couples of the form [V]-W or V-W where V is a sampled value and W is its weight.

`density_r` (*+List:list*) *is det* [Mcintyre]
Display a smooth density estimate of a sets of samples. The samples are in *List* as couples V-W where V is a value and W its weigth.

`densities_r` (*+PriorList:list*, *+PostList:list*) *is det* [Mcintyre]
Display a smooth density estimate of two sets of samples, usually prior and post observations. The samples from the prior are in *PriorList* while the samples from the posterior are in *PostList* as couples V-W where V is a value and W its weigth.

`compute_areas_diagrams_r` (*+LG:list*, *-AUCROC:float*, *-AUCPR:float*) *is det* [Auc]
The predicate takes as input a list *LG* of pairs probability-literal in asceding order on probability where the literal can be an Atom (incading a positive example) or `\+` Atom, indicating a negative example while the probability is the probability of Atom of being true. PR and ROC diagrams are plotted. The predicate returns:

- *AUCROC*: the size of the area under the ROC curve
- *AUCPR*: the size of the area under the PR curve

See `http://cplint.lamping.unife.it/example/exauc.pl` for an example.

4.2 Internals

Important predicates in this library follow a common structure to avoid confusion and promote standardization.

Interface predicates are involved in the interaction between input data from a program and the plot of that same data. These predicates are usable from the programs.

As the name suggests, plotting predicates are only involved in plotting the data.

Finally there are other functions which handle the lists and other types of data.

4.2.1 Interface predicates

All interface predicates have a similar structure. Their names end with `_r` (except the Helpers) in order to distinguish them from the original Cplint predicates.

First and last operations are always `load_r_libraries` and `finalize_r_graph` respectively.

Plotting is done right before the last operation with one of the `geom_` predicates.

A skeleton of the structure follows.

```
<cplint_graphing_predicate>_r(<input>):-
    load_r_libraries,
    <operations on the input>,
    geom_<something>(<new input, possibly lists>),
    finalize_r_graph.
```

4.2.2 Plotting predicates

Predicates directly involved in plotting all start with `geom_` as prefix.

These predicates work with lists which are then transformed into R data frames, and, as a final instruction, a corresponding plot is generated.

You can visualize the structure with the following pseudocode:

```
geom_<something>(<Lists and/or other input>) :-
    <handle lists>,
    <create one or more R data frame with the lists data>,
```

```
    <rename data frame colnames to avoid using default ones>,
    <- ggplot <something>
```

4.2.3 List handling

List handling is useful to pass information between Prolog and R. This is done thanks to `build_xy_list/3`, `r_row/3` and `get_set_from_xy_list/2` predicates, described in the interface section.

In case there are multiple distributions we simply have to call `get_set_from_xy_list/2` the appropriate number of times, like: `get_set_from_xy_list(<something>,R#)`.

4.2.4 Main data frame creation

As descibed before, a data frame is useful to pass structured information between Prolog and R.

In Cplint R in particular, we use `r_data_frame_from_rows/2` provided by the Rserve Client library, in the following manner:

`r_data_frame_from_rows(df[#], R[#])`

For each distribution the optional number is incremented by one.

In case it is the last (or only) data frame then its name will simply be `df`.

4.2.5 Helpers

What follows are extracts of some trivial predicates indicated as internal helpers.

```
bin_width(Min,Max,NBins,Width) :-
  D is Max-Min,
  Width is D/NBins.

load_r_libraries :-
    <- library("ggplot2").

finalize_r_graph :-
    r_download.
```

5 Development

5.1 Changing the library

You are welcomed to change this library. In order to test it you simply have to place you modified version in the correct path and restart SWISH. If you have used the packages provided with SWISH Installer this can be achieved with the following:

```
# mv <your_modified_cplint_r_library> /home/swish/lib/swipl/pack/cplint_r/prolog/cplint_r.pl
# systemctl restart swish-cplint
```

5.2 Compiling this documentation

The source of this documentation is under the `doc` directory of the repository.

To be able to compile it, you have to install several tex packages (for example: `texlive-most` and `texi2html` if you are using Arch Linux) that contain the following binaries:

```
makeinfo
texi2dvi
docbook2html
docbook2pdf
docbook2txt
texi2html
perl
```

After running `make`, a directory named `manual` will be created and you can access all the files by opening `index.html` with a browser.

6 Thanks

Fabrizio Riguzzi for developement, testing and feedback.
Raivo Laanemets for a blog article[1] on how to write SWI Prolog packs.

[1] See item **[Prolog pack development experience]** in Chapter 7 [References], page 12.

7 References

Some quotations reported here are taken directly from the respective web sites.

- [**Cplint**] "A suite of programs for reasoning with probabilistic logic programs". See https://github.com/friguzzi/cplint and https://github.com/friguzzi/cplint/blob/master/doc/help-cplint.pdf for the Cplint help manual.

- [**SWI Prolog**] "SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications". See http://www.swi-prolog.org/

- [**R**] "R is an integrated suite of software facilities for data manipulation, calculation and graphical display". See https://cran.r-project.org/doc/manuals/r-release/R-intro.html

- [**ggplot2**] "ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics". See http://ggplot2.org/

- [**gridExtra**] "Miscellaneous Functions for "Grid" Graphics. Provides a number of user-level functions to work with "grid" graphics, notably to arrange multiple grid-based plots on a page, and draw tables". See https://github.com/baptiste/gridextra

- [**C3js**] "C3 enables deeper integration of charts into your application". See http://c3js.org/

- [**SWI Prolog data types**] See http://www.swi-prolog.org/datatypes.html.

- [**LPN**] "Learn Prolog Now". See http://www.learnprolognow.org/lpnpage.php?pagetype=html&pageid=lpn-htmlch1.

- [**Cplint on SWISH**] "A set of packages that are able to build and install SWISH, Cplint on SWISH and an R environment". See https://frnmst.github.io/swish-installer/

- [**Rserve Client**] "SWI-Prolog/SWISH client for Rserve". "The container exposes Rserve using a Unix-domain socket in /home/rserve/socket". See https://github.com/JanWielemaker/rserve_client

- [**R SWISH**] A library that talks between SWISH and Rserve. See `https://github.com/SWI-Prolog/swish/blob/master/lib/r_swish.pl`

- [**Rserve**] "Rserve is a TCP/IP server which allows other programs to use facilities of R (see www.r-project.org) from various languages without the need to initialize R or link against R library." See `https://www.rforge.net/Rserve/`

- [**SWISH Installer Manual**] See `https://frnmst.github.io/swish-installer/`

- [**Prolog pack development experience**] See `https://rlaanemets.com/post/show/prolog-pack-development-experience`