SWI-Prolog markdown

Emmanuel BATT

11/06/2019

```
library(swiplr)
#options(swipl_bin_folder = "")
```

Simple usage

returns bool

```
bar.
foo(bar).

?- bar.
?- foo(bar).
?- foo(boom).
```

TRUE TRUE FALSE

With data.frame as result

```
maman(claire).
maman(marieOdile).
a_maman(emmanuel, marieOdile).
a_maman(matthew, claire).
a_maman(célestine, claire).

est_enfant(X, Y) :- a_maman(X, Y).
a_frère(X1, X2)  :- a_maman(X1, Y), a_maman(X2, Y), dif(X1,X2).

?- a_frère(SIBLING_1, SIBLING_2)
```

SIBLING_1	SIBLING_2
matthew	célestine
célestine	matthew

hide columns

You can add _ at the end of a variable to remove it from results

```
maman(claire).
maman(marieOdile).
a_maman(emmanuel, marieOdile).
a_maman(matthew, claire).
a_maman(célestine, claire).

est_enfant(X, Y) :- a_maman(X, Y).
a_frère(X1, X2)  :- a_maman(X1, Y), a_maman(X2, Y), dif(X1,X2).

?- a_frère(SIBLING, _)
?- a_frère(SIBLING_, WHO), a_maman(SIBLING_, MUM)
?- a_frère(SIBLING_, WHO)
```

SIBLING
matthew
célestine

WHO	MUM
célestine	claire
matthew	claire

WHO
célestine
matthew

Exchange data with R

```
some_name <- "john"

value_list <- paste0("bar_", 1:3)

named_list <- list(
  var1 = list(
    field1 = "sömé àçênts",
    field2 = 66),
  var2 = "bar0")

some_table <- r_to_pro(iris[sample(1:150, 4),])
```

One can use `whisker::whisker.render` syntax.

```
% Inject the content of R some_name variable
foo({{ some_name }}).

foo('{{ named_list.var1.field1 }}').
```

```
% Iterates through the list of values in R value_list vector
{{#value_list}}
  foo({{.}}).
{{/value_list}}

% Iterate through a table (list of rows)
% use r_to_pro() to convert data.frame to the right format
{{#some_table}}
  some_table({{Species}},{{Sepal_Length}},{{Sepal_Width}}).
{{/some_table}}

?- foo(FOO)
?- some_table(Species, Sepal_Length, Sepal_Width)
```

FOO
john
sömé àçênts
bar_1
bar_2
bar_3

Species	Sepal_Length	Sepal_Width
virginica	7.7	2.6
virginica	6.4	3.1
versicolor	5.9	3
virginica	6.7	2.5

And then get back values in R

```
prolog_output$result_1
```

```
##           FOO
## 1        john
## 2 sömé àçênts
## 3       bar_1
## 4       bar_2
## 5       bar_3
```

Degraded modes

Infinite loop

```
s1(A) :- s2(A).
s2(A) :- s1(A).

?- s1(X)
```

FALSE

Advances features

external library

change interpreter path

```
options(swipl_bin_folder = "/usr/local/bin/")
```

use local library

```
:- use_module(library(pita)).
:- pita.

:- begin_lpad.

visage_type(homme):0.5; visage_type(femme):0.5.
presence_ride(oui):0.4; presence_ride(non):0.6.

age(enfant):0/10 ; age(ado):0/10 ; age(adulte):10/10 :-
  visage_type(homme), presence_ride(non).
age(enfant):1/10 ; age(ado):3/10 ; age(adulte):6/10 :-
  visage_type(homme), presence_ride(oui).
age(enfant):3/10 ; age(ado):3/10 ; age(adulte):3/10  :-
  visage_type(femme).

:- end_lpad.

?- prob(visage_type(X), age(enfant), P)
```

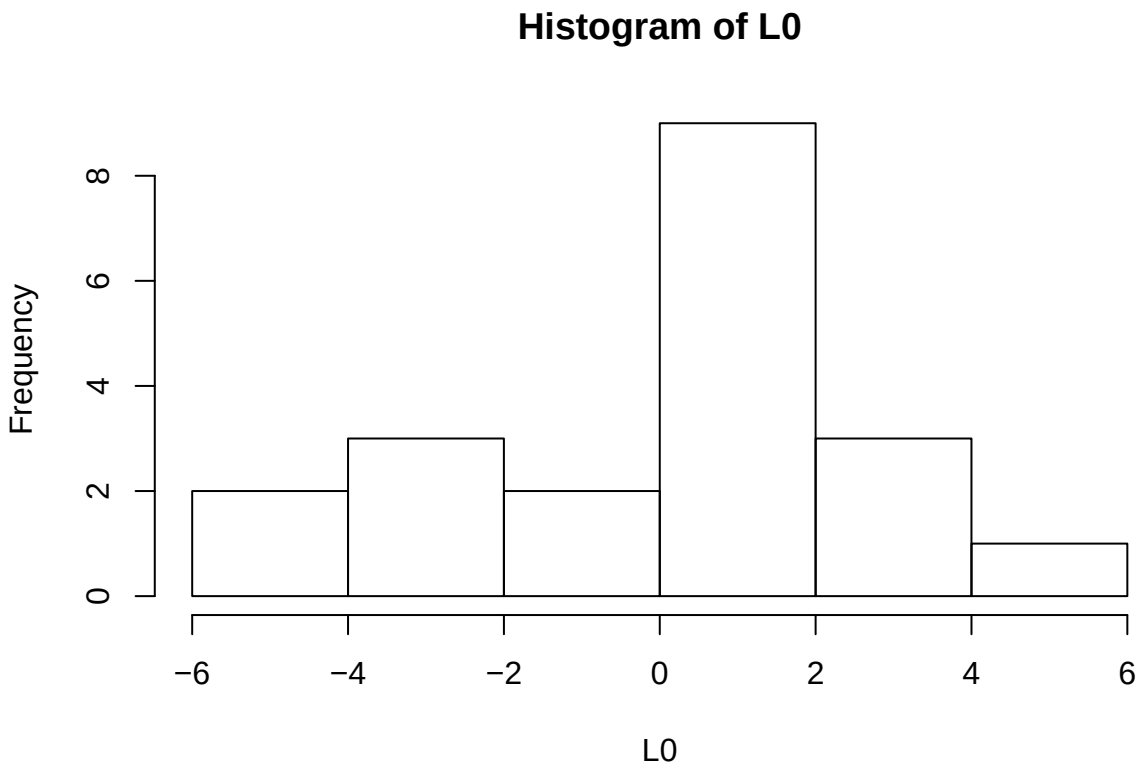
X	P
homme	0.117647058823529
femme	0.882352941176471

```
:- use_module(library(mcintyre)).
:- mc.
:- begin_lpad.

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

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

val(_,M,X): user(X,gauss(M, 2.0)).
% Gaussian distribution of the variable


:- end_lpad.

gauss(Mean,Variance,S):-
  number(Mean),!,
```

```
  random(U1),
  random(U2),
  R is sqrt(-2*log(U1)),
  Theta is 2*pi*U2,
  S0 is R*cos(Theta),
  StdDev is sqrt(Variance),
  S is StdDev * S0 + Mean.

?- mc_sample_arg(val(0,X_),20,X_,L0)
```

L0
-5.16932366449642,-5.11174940912137,-3.81971419525552,-2.32898315890946,-2.28405938722514,-1.32161066974553,-0.0488999302998557,0.0738720488200565,0.302949214838489,0.303427797204795,0.558281597672888,0.660236587533647,1.052

```
L0 <- as.numeric(unlist(strsplit(mc_var$L0, ",")))
hist(L0)
```



Other

variables with first upper case letter

```
are(B, B, []).
are(A, _, T)  :- not(memberchk(X, T)), are(A, X, [X|T]).
```

```
has(A, B) :- has(A, X), has(X, B).

has(human, age).
has(bob, children(john)).

age(mike, 33).
age(bob, 34).

is_a(mike, human).
is_a(bob, human).
is_a(john, children(bob)).

are(children(bob), human_child).
are(X, human_adult) :- is_a(X, human), age(X, AGE), AGE > 18.

?- is_a(Class, Super)
?- are(WHO, WHAT)
?- is_a(Class, Super), are(Super, WHAT)
```

Class	Super
mike	human
bob	human
john	children(bob)

WHO	WHAT
children(bob)	human_child
mike	human_adult
bob	human_adult

Class	Super	WHAT
john	children(bob)	human_child

strange cases

```
foo(no_space).
foo('with space').
foo([this, is, a, list]).
foo([this, is, a, 'list with space']).
foo([this, is, a, [list, [of, list]]]).
foo(this(has(pred))).
foo(this(has('pred with space'))).
foo('even , } { } ) or ( works in atom').

?- foo(X)
```

X
no_space
with space
c(this, is, a, list)
c(this, is, a, “list with space”)
c(this, is, a, c(list, c(of, list)))
this(has(pred))
this(has(“pred with space”))
even , } { }) or (works in atom

use profiling

```
bar(1).
bar(2).
bar(3).

foo(a).
foo(b).
foo(c).

% without termination, we search for the entire space
% foo([], []).

foo([H1|T1], [H2|T2]) :- foo(H1), bar(H2), foo(T1, T2).

?- foo(X, Y)
```

X	Y
NA	NA

known limitations

```
should_we_calculate(case1, 1+1).
should_we_calculate(case2, 2*5).
should_we_calculate(case3, 2/5).
should_we_calculate(case4, 2-5).
should_we_calculate(but5, 5**2).
should_we_calculate(but6, sqrt(9)).
should_we_calculate(bug, c(9, 3)).
% this is due to collision with R c() function

?- should_we_calculate(CASE, RESULT)
```

CASE	RESULT
case1	2
case2	10
case3	0.4
case4	-3

CASE	RESULT
but5	5^2
but6	sqrt(9)
bug	9,3