# SWI-Prolog interface to R

Nicos Angelopoulos

March 17, 2014

**Abstract**

This article documents the package R, a library to talk to R system for Statistical Computing.

## 1 R.pl: R session

**author** Nicos Angelopoulos
**version** 0:0:7
**See also**
- `ensure_loaded(library('../doc/packages/examples/R/r_demo.pl'))`
- `http://www.r-project.org/`


**license** GPL+SWI-exception or Artistic 2.0

This library facilitates interaction with the R system for statistical computing. It assumes an R executable in $PATH or can be given a location to a functioning R executable (see `r_bin/1` and `r_open/1` for details on how R is located). R is ran as a slave with Prolog writing on and reading from the associated streams. Multiple sessions can be managed simultaneously. Each has 3 main components: a name or alias, a term structure holding the communicating streams and a number of associated data items.

The library attempts to ease the translation between prolog terms and R inputs. Thus, Prolog term `x <- c(1,2,3)` is translated to atomic `'x <- c(1,2,3)'` which is then passed on to R. That is, <- is a defined/recognised operator. `X <- c(1,2,3)`, where X is a variable, instantiates X to the list `[1,2,3]`. Also 'Atom' <- [x1,...,xn] translates to R code: Atom <- `c(x1,...,xn)`. Currently vectors, matrices and (R)-lists are translated in this fashion. The goal "A <- B" translates to `r_in( A <- B )`.

Although the library is primarily meant to be used as a research tool, it still provides access to many functions of the R system that may render it useful to a wider audience. The library provides access to R's plethora of vector and scalar functions. We adicipate that of particular interest to Prolog programmers might be the fact that the library can be used to create plots from Prolog objects. Notably creating plots from lists of numbers.

There is a known issue with X11 when R is started without –interactive. `R.pl` runs by default the –interactive flag and try to surpress echo output. If you do get weird output, try giving to r_open, option `with(non_interactive)`. This is suboptimal for some tasks, but might resolve other issues. There is a issue with Macs, where –interactive doesnot work. On Macs, you should use `with(non_interactive)`. This can also be achieved using `settings/2`.

These capabilities are illustrated in the following example :

```
rtest :-
     r_open,
     y <- rnorm(50),
     r_print( y ),
     x <- rnorm(y),
     r_in( x11(width=5,height=3.5) ),
     r_in( plot(x,y) ),
     write( 'Press Return to continue...' ), nl,
     read_line_to_codes( user_input, _ ),
     r_print( 'dev.off()' ),
     Y <- y,
     write( y(Y) ), nl,
     findall( Zx, between(1,9,Zx), Z ),
     z <- Z,
     r_print( z ),
     cars <- c(1, 3, 6, 4, 9),
     r_in(pie(cars)),
     write( 'Press Return to continue...' ), nl,
     read_line_to_codes( user_input, _ ),
     r_close.
```

**r_bin**(*?Rbin*)

Register the default R location, *+Rbin*, or interrogate the current location: *-Rbin*. When interrogating *Rbin* is bound to the R binary that would be used by an `r_open/0`. The order of search is: registered location, environment variable 'R_BIN' and path defined. On unix systems path defined is the first R executable in $PATH. On MS wins it is the latest Rterm.exe found by `expand_file_name( 'C:/Program Files/R/R-*/bin/Rterm.exe', Candidates )`. The value *Rbin* == `retract` retracts the current registered location. *Rbin* == `test`, succeeds if an R location has been registered.

**r_open**

Open a new R session. Same as `r_open( [] )`.

**r_start**

Only start and session via `r_open/1`, if no open session existss.

**r_open**(*+Opts*)

Open a new R session with optional list of arguments. *Opts* should be a list of the following

**alias**(*Alias*)

Name for the session. If absent or a variable an opaque term is generated.

**assert**(*A*)

Assert token. By default session opened last is the default session (see `default_r_session/1`). Using *A* = `z` will push the session to the bottom of the pile.

**at_r_halt(***RHAction***)**

R slaves used to halt when they encounter an error. This is no longer the case but this option is still present in case it is useful in the future. This option provides a handle to changing the behaviour of the session when a halt of the R-slave occurs. *RHAction* should be one of `abort`, `fail`, `call/1`, `call_ground/1`, `reinstate` or `restart`. Default is `fail`. When *RHAction* is `reinstate`, the history of the session is used to roll-back all the commands sent so far. At 'restart' the session is restarted with same name and options, but history is not replayed.

**copy(***CopyTo, CopyWhat***)**

Records interaction with R to a file/stream. *CopyTo* should be one of `null`, `stream(Stream)`, OpenStream, AtomicFile, `once(File)` or `many(File)`. In the case of `many(File)`, file is opened and closed at each write operation. *CopyWhat* should be one of `both`, `in`, `out` or `none`. Default is no recording (*CopyTo* = `null`).

**ssh(***Host***)**

**ssh(***Host, Dir***)**

Run R on *Host* with start directory *Dir*. *Dir* defaults to /tmp. Not supported on MS Windows.

**rbin(***Rbin***)**

R executable location to use for this open operation. If the option is not present binary registered with `r_bin/1` and environment variable R_BIN are examined for the full location of the R binary. In MS windows *Rbin* should point to Rterm.exe. Also see `r_bin/1`.

**with(***With***)**

*With* is in [environ,non_interactive,restore,save]. The default behaviour is to start the R executable with flags `interactive --no-environ --no-restore --no-save`. For each *With* value found in *Opts* the corresponding `--no-` flag is removed. In the case of non_interactive, it removes the default –interactive. This makes the connection more robust, and allows proper x11 plots in linux. However you get alot all the echos of what you pipe in, back from R.

**r_close**

Close the default R session.

**r_close(+***R***)**

Close the named *R* session.

**r_in(+***Rcmd***)**

Push *Rcmd* to the default R session. Output and Errors will be printed to the terminal.

**r_in(+***R, +Rcmd***)**

As `r_in/1` but for session *R*.

**r_push(+***Rcmd***)**

As `r_in/1` but does not consume error or output streams.

**r_push**(+*R, +Rcmd*)
As `r_push/1` but for named session.

**r_out**(+*Rcmd, -Lines*)
Push *Rcmd* to default R session and grab output lines *Lines* as a list of code lists.

**r_out**(+*R, +Rcmd, -Lines*)
As `r_out/2` but for named session *R*.

**r_err**(+*Rcmd, -Lines, -ErrLines*)
Push *Rcmd* to default R session and grab output lines *Lines* as a list of code lists. Error lines are in *ErrLines*.

**r_err**(+*R, +Rcmd, -Lines, -ErrLines*)
As `r_err/3` but for named session *R*.

**r_print**(+*X*)
A shortcut for `r_in( print(X) ).`

**r_print**(+*R, +X*)
As `r_print/1` but for named session *R*.

**r_lines_print**(+*Lines*)
Print a list of code lists (*Lines*) to the user_output. *Lines* would normally be read of an R stream.

**r_lines_print**(+*Lines, +Type*)
As `r_lines_print/1` but *Type* declares whether to treat lines as output or error response. In the latter case they are written on user_error and prefixed with '!'.

**r_lines_print**(+*Lines, +Type, +Stream*)
As `r_lines_print/3` but *Lines* are written on *Stream*.

**r_lib**(+*L*)
A shortcut for `r_in( library(X) ).`

**r_lib**(+*R, +L*)
As `r_lib/1` but for named session *R*.

**r_flush**
Flush default R's output and error on to the terminal.

**r_flush**(+*R*)
As `r_flush/0` but for session *R*.

**r_flush_onto**(+*SAliases, -Onto*)
Flush stream aliases to code lists *Onto*. *SAliases* should be one of, or a list of, [output,error].

**r_flush_onto**(+*R, +SAliases, -Onto*)
As `r_flush_onto/2` for specified session *R*.

**current_r_session**(*?R*)
True if *R* is the name of current *R* session. Can be used to enumerate all open sessions.

**current_r_session**(*?R, ?S, ?D*)
: True if *R* is an open session with streams *S* and data *D* (see introduction to the library).

**default_r_session**(*?R*)
: True if *R* is the default session.

**r_streams_data**(+*SId, +Streams, -S*)
: True if *Streams* is an R session streams structure and *S* is its stream corresponding to identifier *SId*, which should be one of [input,output,error].

**r_session_data**(+*DId, +Data, -Datum*)
: True if *Data* is a structure representing R session associated data and *Datum* is its data item corresponding to data identifier *DId*. *DId* should be in [at_r_halt,copy_to,copy_this,interactive,version,opts].

**r_history**
: Print on user_output the history of the default session.

**r_history**(*-H*)
: *H* unifies to the history list of the Rcmds fed into the default session. Most recent command appears at the head of the list.

**r_history**(*?R, -H*)
: As `r_history/1` but for named session *R*. It can be used to enumerate all histories. It fails when no session is open.

**r_session_version**(*-Version*)
: Installed version. *Version* is of the form Major:Minor:Fix, where all three are integers.

**r_verbosity**(*?Level*)
: Set, *+Level*, or interrogate, *-Level*, the verbosity level. *+Level* could be `false` (=0), `true` (=3) or an integer in {0,1,2,3}. 3 being the most verbose. The default is 0. *-Level* will instantiate to the current verbosity level, an integer in {0,1,2,3}.

**r_bin_version**(*-Version*)
: Get the version of R binary identified by `r_bin/1`. *Version* will have the same structure as in `r_session_version/1` ie M:N:F.

**r_bin_version**(+*Rbin, -Version*)
: Get the version of R binary identified by *+Rbin*. *Version* will have the same structure as in `r_session_version/1` ie M:N:F.

**settings**(+*Setting, +Value*) *[multifile]*
: Multifile hook-predicate that allows for user settings to sip through. Currently the following are recognised:

  **r_open_opt**
  : These come after any options given explicitly to `r_open/1`. For example on a Mac to avoid issue with –interactive use the following before querring `r_open/0`,1.

```
:- multifile settings/2.
r_session:settings(r_open_opt,with(non_interactive)).
```

**atom_is_r_function**

expands atoms such as x11 to r function calls x11()

**r_function_def(**+*Function*)

where *Function* is an R function. This hook allows default argument values to R functions. Only Arg=*Value* pairs are allowed.

```
:- multifile settings/2.
r_session:settings(r_function_def(x11),width=5).
```

# Index