Calling XLisp from R

Duncan Temple Lang

September 1, 2002

This is a very incomplete document giving some examples/tests for running XLisp within R. They are here to give some idea of what is currently working and how one can call functions.

One starts the R-XLisp connection by loading the RXLisp package and call *.XLispInit()*.

1a ⟨ 1a⟩≡

```
library(RXLisp)
.XLispInit()
```

.XLispInit() accepts a character vector which is used as the command line arguments passed to the XLispStat engine. The defaults are given in *.XLispInitArgs* and provide the location of the XLispStat saved work area which was created when XLispStat was compiled and installed. You can append to this or override it. Simple calls to XLisp functions are done via the *.XLisp()* function in R. The first argument is the name of the XLisp function to be invoked. R values that are to be passed to the XLisp function/method call are given next. For example, we can use XLisp's random number generation facilities to generate a sample of 10 values from a Poisson distribution with mean 3.5.

1b ⟨ 1a⟩+≡

```
.XLisp("poisson-rand", as.integer(10), 3.5)
```

Note that the following command does not work since the 10 is represented in R as a real number rather than an integer. So the XLisp function poisson-rand fails.

1c ⟨*Example* 1c⟩≡

```
> .XLisp("poisson-rand", 10, 3.5)
Error: bad argument type - 10.0
Happened in: #<Subr-POISSON-RAND: #8a65f90>
Error in .XLisp("poisson-rand", as.numeric(10), 3.5) :
        Error in XLisp computation
```

It is also possible to supply named arguments to XLisp functions from R, in either of two ways. The primitive mechanism is to specify the name as a separate value in the R argument list and use the XLisp syntax by prefixing the name with a :. Alternatively, one can use the R notation and specify the name to be passed to XLisp simply as the name for the R argument. In this case, the *.XLisp()* function will prepend the name with the necessary : character. In both cases, the names will be converted to upper-case as is needed by XLisp.

1d ⟨ 1a⟩+≡

```
.XLisp("histogram", rnorm(100), ":title", "My Title")
.XLisp("histogram", rnorm(100), title="My Title")
```

Note that the first way is necessary if the named argument in XLisp is not followed by a value, e.g

2a ⟨ 1a⟩+≡

```
h <- .XLisp("histogram", rnorm(100))
.XLisp("SEND", h, ":close")
```

1 Graphics

These examples show how we can create At present, the return values are not protected and should not be used in subsequent calls. This is almost working.

2b ⟨ 1a⟩+≡

```
.XLisp("histogram", rnorm(100))
```

2c ⟨ 1a⟩+≡

```
myData <- list(rnorm(20), rnorm(20), rnorm(20))
.XLisp("scatterplot-matrix", myData)
```

These two examples illustrate the different approaches and create a scatterplot matrix whose variables are given the appropriate labels.

2d ⟨ 1a⟩+≡

```
.XLisp("scatterplot-matrix", myData, ":variable-labels", list("a", "b", "c"))
.XLisp("scatterplot-matrix", myData, "variable-labels"= list("a", "b", "c"))
```

This example merely illustrates that we can use XLispStat's facilities for interactive 3-D graphics directly from R.

2e ⟨ 1a⟩+≡

```
.XLisp("spin-plot", myData)
```

2 XLisp Reference Objects

2f ⟨ 1a⟩+≡

```
h <- .XLisp("histogram", rnorm(100))
.Xlisp("send", h, ":close")
```

A more succinct way to express such method calls to send in R is to use the overloaded *$()* notation. We have defined an R method for this for *XLispReference* objects which performs the dispatch directly. So the call to the close method could be invoked more readily in R as

2g ⟨ 1a⟩+≡

```
h$close()
```

Similarly, one can access slot values in XLisp references objects from R using the familiar *[[()* operator.

2h ⟨ 1a⟩+≡

```
h[["title"]]
```

3 Not yet working!

The following do not work yet as we need to export an R object as a reference and have callbacks to it.

See page 59 of [?].

⟨ 1a⟩+≡

```
.XLisp("spin-function", function(x, y) { x^2 + y^2})
```

Coming soon... Slider example from page 61 of [?].

Need the reference manager or a way to add nodes to the `forward_node()` list of gc-protected nodes. This is now available in the `ObjectManager.lsp` file.

⟨ 1a⟩
⟨*Example* 1c⟩