Subject: Re: LIST extensions
Posted by penteado on Sun, 02 Jan 2011 06:37:49 GMT
View Forum Message <> Reply to Message

On Dec 22 2010, 9:47 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> Other changes I am considering to put in my derived classes:
>
> 1) Make lists do nothing (as hashes already do) if !null is used as
> index on _overloadBracketsLeftSide.
>
> 2) Make lists and hashes return !null when !null is used as an index
> (now they throw an error).
>
> 3) Make lists and hashes accept !null on the _overloadPlus method and
> do nothing, instead of throwing an error.
>
> (3) is to work in conjunction with (2), so that lists/hashes can be
> added to indexed lists/hashes, without having to verify if the index
> is not !null.
>
> Any thoughts?

I have really been finding inconvenient the lack of these, and noticed
another shortcoming: _overloadPlus should add to a list something that
is not a list. So that

```
l1=list()
l2=list(1,2,3)
w=where(l2.toarray() eq 2)
l1+=l2[w]
```

Does not throw an error. As it is now, it takes a lot of work to
select elements from a list with where(): not only it is necessary to
test for no results (because !null is not accepted as index for
lists), but it is also necessary to test for a single match, as a list
indexed by a scalar (or 1-element array) returns the list element,
which cannot be concatenated to a list (unless the element happens to
be a list, which would not throw an error, but would concatenate in
the wrong way).

An alternative is not change _overloadPlus, but change
_overloadBracketsRightSide to return a 1-element list when given a 1-
element array as index. It should still return the element when
indexed by a scalar.

And doing these things also makes me think that, for syntatic sugar,
there should be a list::where() method that would simply call where()
on the list's toarray() result. Or where() should automatically call

toarray() if given a list.