package: s3x 0.9.2

Enhanced S3 Programming (Rough Draft)

Charlotte Maia

November 22, 2012

An R package for enhanced S3 programming, including: Redefinitions of most base/graphics generics (without object/x as their first argument); Enhanced functions (for object-functional programming, mixing object oriented programming with functional programming); Other related utility functions.

Introduction

This is an intentionally minimalistic R package.

Roughly speaking, R has two major language systems for object oriented programming, generally referred to as “S3” and “S4”.

In principle, this package tries to enhance R’s S3 system, which, in my personal opinion, is incomplete. However, it’s fair to say, that this package has largely evolved, to meet the needs of the author’s other packages.

Roughly speaking, this package has six parts:

1. Redefinitions of most base/graphics generics.
   This is essential, for S3 to support object oriented models.
2. Utility function, for extending class attributes.
3. Enhanced functions.
4. Enhanced function plots.
5. Utility functions, for reading package/file data.
6. Utility function, for deterministic sampling.

Re-iterating, this is an intentionally minimalistic R package.
A previous mistake of the author, was over diversifying.

Arguably, S3 could be enhanced in many other ways.

There’s a need for enhanced vectors.
However, if there’s enhanced vectors, then there’s a need for enhanced matrices.

This results in a complexity explosion...

Getting Started

Firstly, we need to load the package.

```
R> library (s3x)
```

Generic Redefinitions

Most generics from base and graphics have been redefined.

Here is an excerpt from the corresponding source file:

```
"all.equal" = function (...) base::"all.equal" (...)
"anyDuplicated" = function (...) base::"anyDuplicated" (...)
"as.array" = function (...) base::"as.array" (...)
"as.data.frame" = function (...) base::"as.data.frame" (...)
```

By default, many S3 methods, must name their first argument “object” or “x”.
Often this is highly inappropriate.
At least in an object oriented context, anyway.

After loading the s3x package, any corresponding S3 methods, can have whatever first argument they like.

The “extend” Function

This function merely returns an object, with a concatenated class attribute.

See the Rd file for more information.

```
R> x = 0
R> x = extend (x, "my_super_class")
R> x = extend (x, "my_sub_class")
R> x
```

```
[1] 0
attr(,"class")
[1] "my_sub_class"   "my_super_class"
```

Enhanced Function Objects

Enhanced functions (or “FUNCTION” objects, capitalised), wrap standard function objects. The major difference, is that they provide an operator “$”, to access their attributes. Note that the operator can be used inside functions to read (however, not write) their attributes.

Enhanced functions are intended for what I refer to as “Object-Functional Programming”, mixing object oriented programming with functional programming.

The major application of this is modelling interpolated functions.

Here’s an example with a straight line.

```
R> straight_line = function (intercept, slope)
   {        f_seed = function (x) .$intercept + .$slope * x
           f = FUNCTION (f_seed)
           f$intercept = intercept
           f$slope = slope
           f
   }
R> f1 = straight_line (0, 1)
R> f1
```

```
FUNCTION (x)
{  .$intercept + .$slope * x
}
object attributes:
intercept, slope
```

```
R> f1$intercept
```

```
[1] 0
```

```
R> f1$slope
```

```
[1] 1
```

```
R> f2 = straight_line (1, 2)
R> f2$intercept
```

```
[1] 1
```

```
R> f2$slope
```

```
[1] 2
```

Note that at the time of writing, there's a error printing functions.

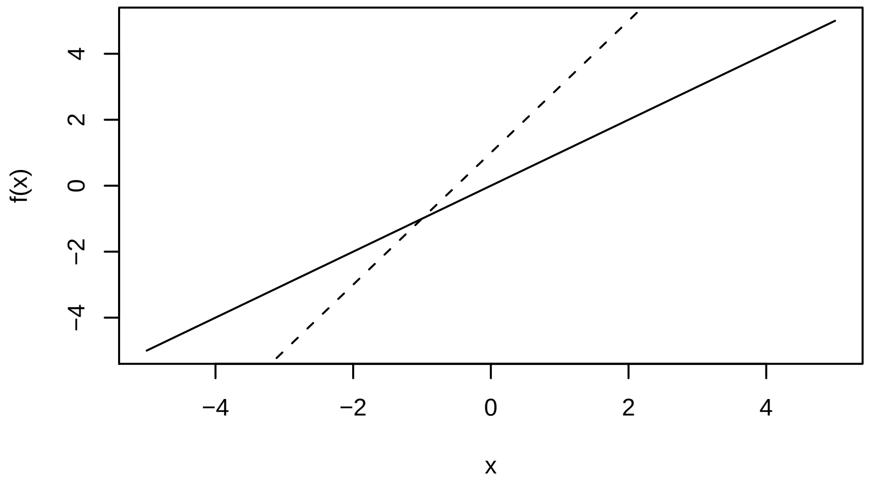
Plotting Simple Functions

We can plot simple functions.

We need to either specify an xlim argument in the constructor or the plotting function.

After we have plotted one function, we can add others..

```
R> plot (f1, xlim=c (-5, 5) )
R> lines (f2, lty=2, xlim=c (-5, 5) )
```



I/O Utility Functions

The functions "read_package_data" and "read_file_data" provide alternatives to existing I/O functions. Note that the defaults assume a CSV (comma separated) file with a header row and don't enforce column names with valid syntax.

See the Rd file for more information.

```
R> args (read_package_data)
```

```
function (package, file, sep = ",", header = TRUE, check.names = FALSE,
    ...)
NULL
```

```
R> args (read_file_data)
```

```
function (path, sep = ",", header = TRUE, check.names = FALSE,
    ...)
NULL
```

Deterministic Sampling

The functions "deterministic_sample" and "detsam", primarily intended for using I/O functions within vignettes, simplify the process of printing the first n and last m rows of a "data.frame" or "matrix".

```
R> #default, first 3, last 3
R> detsam (cars)
```

```
   speed dist
1      4    2
2      4   10
3      7    4
48    24   93
49    24  120
50    25   85
```

```
R> #alternative, first two, last two
R> detsam (cars, 2)
```

```
   speed dist
1      4    2
2      4   10
49    24  120
50    25   85
```

```
R> #alternative, first two, last one
R> detsam (cars, 2, 1)
```

```
   speed dist
1      4    2
2      4   10
50    25   85
```