Package ‘vfunc’

July 28, 2025

Type Package

Title Manipulate Virtual Functions

Version 1.0

Depends R (>= 4.2.0)

Suggests testthat, knitr, rmarkdown, onion

Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>

Description If f <- function(x){x^2} and g <- function(x){x+1} it is a constant source of annoyance that ``f+g'' is not defined. Package 'vfunc' allows you to do this, and we have (f+g)(2) returning 5. The other arithmetic operators are similarly implemented. A wide class of coding bugs is eliminated.

License GPL-2

Imports methods

VignetteBuilder knitr

NeedsCompilation no

Author Robin K. S. Hankin [aut, cre] (ORCID: <https://orcid.org/0000-0001-5982-0415>)

Repository CRAN

Date/Publication 2025-07-28 18:50:02 UTC

Contents

vfunc-package *Manipulate Virtual Functions*

Description

If `f <- function(x){x^2}` and `g <- function(x){x+1}` it is a constant source of annoyance to me that "f+g" is not defined. Package vfunc allows you to do this.

Details

The package defines a single S4 class, `vf`. This has a single slot, `.Data`, of type `function` which means that `vf` objects inherit much of the behaviour of functions, but for which new methods (such as the `Arith` group of `S4` generics) may be defined.

Documentation Index

Index of help topics:

```
Compare-methods         'Compare' methods for 'vf' objects
Math                    Math group generic functions in the 'vfunc'
                        package: trig, exponential, log, etc.
Math-methods            Methods for Function 'Math', 'Arith' in the
                        'vfunc' package
as.vf                   Coerce functions to a virtual function.
pow                     Iterated functions; functional powers
vf-class                Class '"vf"'
vfunc-package           Manipulate Virtual Functions
```

Author(s)

Robin K. S. Hankin [aut, cre] (ORCID: <https://orcid.org/0000-0001-5982-0415>)

Maintainer: Robin K. S. Hankin <hankin.robin@gmail.com>

Examples

```
f <- as.vf(function(x){x^2})

f + Sin

as.function(f*Sin + Exp)(1:4)
```

`as.vf` *Coerce functions to a virtual function.*

Description

Coerce objects to a virtual function. Numeric or complex arguments are coerced to a constant function.

Usage

```
as.vf(x)
```

Arguments

`x`	Generally, a function or numeric

Value

Returns an object of class `vf`.

Note

It is rarely necessary to coerce objects such as vectors or matrices to class `vf` because the Arith methods operate on objects of class `ANY` directly.

Author(s)

Robin K. S. Hankin

Examples

```
as.vf(\(x)x^2)
Sin + as.vf(\(p){p^3})

as.vf(1:10)(1e99)
```

Compare-methods — Compare *methods for* vf *objects*

Description

Wouldn't it be nice to say `(f > g)(x)` rather than the terrible, tedious and error-prone `f(x) > g(x)`? Well, now you can!

Methods

`signature(e1 = "ANY", e2 = "vf")`

`signature(e1 = "function", e2 = "vf")`

`signature(e1 = "vf", e2 = "ANY")`

`signature(e1 = "vf", e2 = "function")`

`signature(e1 = "vf", e2 = "vf")`

Examples

```
x <- seq(from=0, to=2*pi, len=100)
(Sin > Cos*Tan)(x)
```

Math — *Math group generic functions in the* **vfunc** *package: trig, exponential, log, etc.*

Description

The S4 `Math` group contains 35 functions including `sin()`, `log()`, etc. The `vfunc` equivalents are capitalized, as in `Sin()`, `Log()`, etc.

Usage

```
Abs(x)
Sign(x)
Sqrt(x)
Ceiling(x)
Floor(x)
Trunc(x)
Cummax(x)
Cummin(x)
Cumprod(x)
Cumsum(x)
Log(x)
Log10(x)
```

```
Log2(x)
Log1p(x)
Acos(x)
Acosh(x)
Asin(x)
Asinh(x)
Atan(x)
Atanh(x)
Exp(x)
Expm1(x)
Cos(x)
Cosh(x)
Cospi(x)
Sin(x)
Sinh(x)
Sinpi(x)
Tan(x)
Tanh(x)
Tanpi(x)
Gamma(x)
Lgamma(x)
Digamma(x)
Trigamma(x)
```

Arguments

x	Generally take a single argument of class numeric, function, or vf

Details

The reason for this rather untransparent device is that primitive functions such as sin() behave somewhat differently from other functions. We have:

```
Sin <- as.vf(function(x){sin(x)})
setMethod("sin", "vf", function(x){as.vf(function(o){Sin(x(o))})})
```

We define Sin() to be an object of class vf; the call to setMethod() ensures that Sin(f) operates as intended.

Value

Given a numeric, return a numeric; given a vf, return a vf

Note

Note that "sin <- as.vf(sin)" does not work as desired, giving a runtime error; trying to get round this with things like "sin <- as.vf(function(x)sin)" and similar means that "sin(3)" does not work.

There is no way to inform all vf objects that, if used as a function with an argument of a primitive such as sin, to return another vf object—and not to try and evaluate "f(sin)", which fails:

```
f <- as.vf(function(x){x^2 + 1})
f(Sin)
#> An object of class "vf"
#> function (...)
#> {
#>     e1(...) + e2
#> }
#> <bytecode: 0x6065e7c8a900>
#> <environment: 0x6065e7c8a548>
f(sin)
#> Error in x^2: non-numeric argument to binary operator
```

Above, we see `f(sin)` returning an error (it tries to evaluate "`sin^2 + 1`"). Observe that "`Sin^2 + 1`" is perfectly OK, for `Sin` is a virtual function.

Author(s)

Robin K. S. Hankin

Examples

```
Sin + Exp

c((Sin + Exp)(.02232) ,sin(0.02232) +  exp(0.02232))
```

`Math-methods` *Methods for Function* `Math`*,* `Arith` *in the* `vfunc` *package*

Description

Various S4 methods to work with `vf` objects. Comparison methods are documented at `Compare-methods`.

`pow` *Iterated functions; functional powers*

Description

Given a function $f\colon X \longrightarrow X$, we define

$$f^0 = \mathrm{id}_X$$

$$f^{n+1} = f \circ f^n = f^n \circ f, \qquad n \geqslant 0$$

This gives us $f^{n+m} = f^n \circ f^m$ and $(f^m)^n = f^{mn}$, which motivates the notation. For example, $\sin^3 = \sin \circ \sin \circ \sin$, so $\sin^3(x) = \sin(\sin(\sin x))$.

The operator is well-defined due to the power associativity of function composition.

Usage

```
pow(x, n)
```

Arguments

x	Object of class vf
n	Non-negative integer

Value

Returns an object of class vf

Note

There are possibly more efficient methods requiring fewer compositions, e.g. pow(f,9) (which would require 8 function compositions) could be evaluated by pow(pow(f,3),3) (which requires only four). But I am not sure that this would actually be any faster, and I have not got round to thinking about it yet.

Also, package idiom for the caret "^" is reserved for arithmetic exponentiation [so, for example, (f^3)(x) == f(x)*f(x)*f(x)]. I believe this is sub-optimal but was unable to overload the caret to implement functional iteration.

Author(s)

Robin K. S. Hankin

Examples

```
pow(Sin,5)
Sin^5

f <- as.vf(function(x){x^2+1})

pow(f + Sin,4)
pow(f + Sin,4)(2)
```

---

vf-class *Class "vf"*

Description

Class vf stands for "virtual function"

Objects from the Class

Objects can be created by calls of the form new("vf", ...).

Slots

`.Data`: Object of class `"function"`

Methods

Arith `signature(e1 = "function", e2 = "vf")`: ...

Arith `signature(e1 = "ANY", e2 = "vf")`: ...

Arith `signature(e1 = "vf", e2 = "function")`: ...

Arith `signature(e1 = "vf", e2 = "missing")`: ...

Arith `signature(e1 = "vf", e2 = "ANY")`: ...

Arith `signature(e1 = "vf", e2 = "vf")`: ...

as.function `signature(x = "vf")`: ...

as.vf `signature(x = "vf")`: ...

coerce `signature(from = "function", to = "vf")`: ...

coerce `signature(from = "ANY", to = "vf")`: ...

coerce `signature(from = "vf", to = "function")`: ...

Compare `signature(e1 = "function", e2 = "vf")`: ...

Compare `signature(e1 = "ANY", e2 = "vf")`: ...

Compare `signature(e1 = "vf", e2 = "function")`: ...

Compare `signature(e1 = "vf", e2 = "ANY")`: ...

Compare `signature(e1 = "vf", e2 = "vf")`: ...

Math `signature(x = "vf")`: ...

Author(s)

Robin K. S. Hankin

Examples

```
showClass("vf")
```

Index