Package: float (via r-universe)

November 1, 2026

Type Package

Title 32-Bit Floats

Version 0.3-4

Description R comes with a suite of utilities for linear algebra with ``numeric'' (double precision) vectors/matrices. However, sometimes single precision (or less!) is more than enough for a particular task. This package extends R's linear algebra facilities to include 32-bit float (single precision) data. Float vectors/matrices have half the precision of their ``numeric''-type counterparts but are generally faster to numerically operate on, for a performance vs accuracy trade-off. The internal representation is an S4 class, which allows us to keep the syntax identical to that of base R's. Interaction between floats and base types for binary operators is generally possible; in these cases, type promotion always defaults to the higher precision. The package ships with copies of the single precision 'BLAS' and 'LAPACK', which are automatically built in the event they are not available on the system.

License BSD 2-clause License + file LICENSE

Copyright The copyright for the single precision BLAS/LAPACK distribution located in src/lapack is given in the file src/lapack/LICENSE.

Depends R (>= 3.6.0), methods

Imports utils, tools

ByteCompile yes

LazyData TRUE

StagedInstall TRUE

URL https://github.com/wrathematics/float

BugReports https://github.com/wrathematics/float/issues

Maintainer Stefano Cacciatore <tkcaccia@gmail.com>

NeedsCompilation yes

Config/roxygen2/version 8.1.0

Author Drew Schmidt [aut, cph], Stefano Cacciatore [aut, cre], Wei-Chen Chen [aut], Dmitriy Selivanov [ctb] (improvements in external package linking), ORNL [cph]

Repository https://cran.r-universe.dev

Date/Publication 2026-10-02 12:52:05 UTC

RemoteUrl https://github.com/cran/float

RemoteRef HEAD

RemoteSha db46f46caa1234e74e6460d681faf403f52d57af

Contents

float-package . . . 3
arithmetic . . . 3
backsolve . . . 6
bind . . . 7
bracket . . . 8
c . . . 9
chol . . . 9
chol2inv . . . 10
colsums . . . 11
comparison . . . 12
converters . . . 12
crossprod . . . 14
diag . . . 15
dims . . . 15
eigen . . . 16
extremes . . . 17
float . . . 18
float32 . . . 19
float32-class . . . 19
hyperbolic . . . 20
is.float . . . 21
isSymmetric . . . 21
log . . . 22
Machine_float . . . 23
mathis . . . 24
matmult . . . 24
miscmath . . . 25
na . . . 26
NA_float_ . . . 27
names . . . 27
NaNf . . . 28
norm . . . 28

print-float32 29
qr 30
rand 31
rcond 32
rep 33
round 33
scale 34
sign 35
solve 35
specialmath 36
sum 37
svd 38
sweep 38
trig 39
xpose 40

Index **42**

`float-package` *32-Bit Floats*

Description

R comes with a suite of utilities for linear algebra with "numeric" (double precision) vectors/matrices. However, sometimes single precision (or less!) is more than enough for a particular task. This package extends R's linear algebra facilities to include 32-bit float (single precision) data. Float vectors/matrices have half the precision of their "numeric"-type counterparts but are generally faster to numerically operate on, for a performance vs accuracy trade-off. The internal representation is an S4 class, which allows us to keep the syntax identical to that of base R's. Interaction between floats and base types for binary operators is generally possible; in these cases, type promotion always defaults to the higher precision. The package ships with copies of the single precision 'BLAS' and 'LAPACK', which are automatically built in the event they are not available on the system.

Author(s)

Drew Schmidt

`arithmetic` *arithmetic*

Description

Binary arithmetic numeric/float matrices.

Usage

```
## S4 method for signature 'float32,float32'
e1 + e2

## S4 method for signature 'float32,float32'
e1 * e2

## S4 method for signature 'float32,float32'
e1 - e2

## S4 method for signature 'float32,float32'
e1 / e2

## S4 method for signature 'float32,float32'
e1 ^ e2

## S4 method for signature 'float32,BaseLinAlg'
e1 + e2

## S4 method for signature 'float32,BaseLinAlg'
e1 * e2

## S4 method for signature 'float32,BaseLinAlg'
e1 - e2

## S4 method for signature 'float32,BaseLinAlg'
e1 / e2

## S4 method for signature 'float32,BaseLinAlg'
e1 ^ e2

## S4 method for signature 'BaseLinAlg,float32'
e1 + e2

## S4 method for signature 'BaseLinAlg,float32'
e1 * e2

## S4 method for signature 'BaseLinAlg,float32'
e1 - e2

## S4 method for signature 'BaseLinAlg,float32'
e1 / e2

## S4 method for signature 'BaseLinAlg,float32'
e1 ^ e2

## S4 method for signature 'float32,float32'
e1 < e2
```

```
## S4 method for signature 'float32,float32'
e1 <= e2

## S4 method for signature 'float32,float32'
e1 == e2

## S4 method for signature 'float32,float32'
e1 > e2

## S4 method for signature 'float32,float32'
e1 >= e2

## S4 method for signature 'float32,BaseLinAlg'
e1 < e2

## S4 method for signature 'float32,BaseLinAlg'
e1 <= e2

## S4 method for signature 'float32,BaseLinAlg'
e1 == e2

## S4 method for signature 'float32,BaseLinAlg'
e1 > e2

## S4 method for signature 'float32,BaseLinAlg'
e1 >= e2

## S4 method for signature 'BaseLinAlg,float32'
e1 < e2

## S4 method for signature 'BaseLinAlg,float32'
e1 <= e2

## S4 method for signature 'BaseLinAlg,float32'
e1 == e2

## S4 method for signature 'BaseLinAlg,float32'
e1 > e2

## S4 method for signature 'BaseLinAlg,float32'
e1 >= e2
```

Arguments

e1, e2	Numeric/float vectors/matrices.

Value

A matrix of the same type as the highest precision input.

Examples

```
library(float)

s1 = flrunif(5, 5)
s2 = flrunif(5, 5)
x = matrix(1:25, 5)

s1 + s2 # float

typeof(x) # integer
x + s2 # float

storage.mode(x) = "double"
x + s2 # double
```

---

backsolve *backsolve*

Description

Solve a triangular system.

Usage

```
## S4 method for signature 'float32,float32'
backsolve(r, x, k = ncol(r), upper.tri = TRUE, transpose = FALSE)

## S4 method for signature 'float32,BaseLinAlg'
backsolve(r, x, k = ncol(r), upper.tri = TRUE, transpose = FALSE)

## S4 method for signature 'BaseLinAlg,float32'
backsolve(r, x, k = ncol(r), upper.tri = TRUE, transpose = FALSE)

## S4 method for signature 'float32,float32'
forwardsolve(l, x, k = ncol(l), upper.tri = FALSE, transpose = FALSE)

## S4 method for signature 'float32,BaseLinAlg'
forwardsolve(l, x, k = ncol(l), upper.tri = FALSE, transpose = FALSE)

## S4 method for signature 'BaseLinAlg,float32'
forwardsolve(l, x, k = ncol(l), upper.tri = FALSE, transpose = FALSE)
```

Arguments

r, l	A triangular coefficients matrix.
x	The right hand sides.
k	The number of equations (columns of r + rows of x) to use.
upper.tri	Should the upper triangle be used? (if not the lower is)
transpose	Should the transposed coefficients matrix be used? More efficient than manually transposing with t().

Examples

```
library(float)

s = flrunif(10, 3)
cp = crossprod(s)
y = fl(1:3)
backsolve(cp, y)
```

---

bind *rbind*

Description

rbind() and cbind() for floats.

Usage

```
## S3 method for class 'float32'
rbind(..., deparse.level = 1)

## S3 method for class 'float32'
cbind(..., deparse.level = 1)
```

Arguments

...	vectors or matrices (numeric or float)
deparse.level	ignored

Value

A matrix of the same type as the highest precision input.

Examples

```
library(float)
x = fl(matrix(1:10, 5))

rbind(x, x)
cbind(x, x)
```

---

bracket *Extract*

Description

Extract subsets of a float vector/matrix.

Usage

```
## S4 method for signature 'float32'
x[i, j, drop = TRUE]

## S4 replacement method for signature 'float32'
x[i, j, ...] <- value
```

Arguments

x	A float vector/matrix.
i, j, ...	The indices. Most combinations of integer/double/logical values will be treated the same as R does. One major difference is that NA values will not be tolerated.
drop	Logical. If TRUE, single column matrices will be treated as one-dimensional vectors.
value	The replacement value.

Value

A float vector/matrix.

Examples

```
## Not run:
library(float)

s = flrunif(10, 3)
s[, -1]
s[c(1, 3, 5, 7), 1:2]

## End(Not run)
```

c *c*

Description

Combine float/numeric vector(s)/matri[x|ces].

Usage

```
## S4 method for signature 'float32'
c(x, ...)
```

Arguments

x	A float matrix.
...	Additional elements (numeric/float vectors/matrices) to sum.

Value

A matrix of the same type as the highest precision input.

Examples

```
library(float)
x = flrunif(10, 3)

c(x, NA, 1L)
```

chol *chol*

Description

Cholesky factorization for a float vector/matrix.

Usage

```
## S4 method for signature 'float32'
chol(x)
```

Arguments

x	A float vector/matrix.

Value

A float vector/matrix.

Examples

```
library(float)

s = flrunif(10, 3)
cp = crossprod(s)
chol(cp)
```

---

chol2inv *chol2inv*

Description

Return the inverse of the original matrix using the Cholesky factorization of a float vector/matrix.

Usage

```
## S4 method for signature 'float32'
chol2inv(x, size = NCOL(x), LINPACK = FALSE)
```

Arguments

x	A float vector/matrix.
size	The number of columns to use.
LINPACK	Ignored.

Value

A float vector/matrix.

Examples

```
library(float)

s = flrunif(10, 3)
cp = crossprod(s)
cp %*% chol2inv(chol(cp))
```

colsums *colSums*

Description

Row and columns sums/means.

Usage

```
## S4 method for signature 'float32'
colSums(x, na.rm = FALSE, dims = 1)

## S4 method for signature 'float32'
rowSums(x, na.rm = FALSE, dims = 1)

## S4 method for signature 'float32'
colMeans(x, na.rm = FALSE, dims = 1)

## S4 method for signature 'float32'
rowMeans(x, na.rm = FALSE, dims = 1)
```

Arguments

x	A float vector/matrix.
na.rm	Should missing values be removed?
dims	Ignored. Be honest, you've never even used this argument before, have you?

Value

A matrix of the same type as the highest precision input.

Examples

```
library(float)

s = flrunif(5, 3)

rowSums(s)
colSums(s)
```

comparison *comparison*

Description

Binary comparison operators for numeric/float matrices.

Value

A vector/matrix of logicals.

Examples

```
## Not run:
library(float)
s = flrunif(5, 5)
x = matrix(1:25, 5)

s > x
s <= 0

## End(Not run)
```

converters *converters*

Description

Convert between a numeric vector/matrix and a float vector/matrix.

Usage

```
fl(x, strict = FALSE)

dbl(x, strict = FALSE)

int(x, strict = FALSE)

as.float(x, strict = FALSE)

## S3 method for class 'float32'
as.double(x, ...)

## S3 method for class 'float32'
as.integer(x, ...)
```

```
## S4 method for signature 'float32'
as.numeric(x, ...)

## S3 method for class 'float32'
as.vector(x, mode = "any")

## S3 method for class 'float32'
as.matrix(x, ...)

## S3 method for class 'float32'
as.data.frame(x, ...)

## S4 method for signature 'float32'
typeof(x)

## S4 method for signature 'float32'
storage.mode(x)
```

Arguments

x	A numeric or float vector/matrix.
strict	Should the function error if given the wrong kind of input? Otherwise it just silently returns the input.
mode, ...	Ignored.

Details

`fl()`, `int()`, and `dbl()` are shorthand for `as.float()`, `as.integer()`, and `as.double()`, respectively.

Value

The data stored in the type of whatever was asked for (the opposite of the input).

Examples

```
library(float)

x = matrix(1:30, 10, 3)
s = fl(x)

y = dbl(s)

all.equal(x, y)
```

crossprod *crossprod*

Description

Croddproducts.

Usage

```
## S4 method for signature 'Mat'
crossprod(x, y = NULL)

## S4 method for signature 'Mat'
tcrossprod(x, y = NULL)
```

Arguments

x	A float vector/matrix.
y	Either NULL, or a numeric/float matrix.

Details

If y is a numeric matrix, then x will be promoted to a numeric matrix, and the return will therefore be numeric (not float).

Value

A float matrix (unless y is numeric; see details section).

Examples

```
library(float)

s = flrunif(10, 3)
crossprod(s)
tcrossprod(s)
```

diag *diag*

Description

Methods for getting the diagonal of a float matrix, or constructing a float matrix given a float vector.

Usage

```
## S4 method for signature 'float32'
diag(x = 1, nrow, ncol)
```

Arguments

x	A float vector (create a diagonal matrix) or matrix (get its diagonal).
nrow, ncol	As in base R's diag().

Value

A float vector or matrix, depending on the input.

Examples

```
library(float)

s = flrunif(10, 3)
s
diag(s)
diag(diag(s))
```

dims *dim*

Description

Dimension information for a float vector/matrix.

Usage

```
## S4 method for signature 'float32'
nrow(x)

## S4 method for signature 'float32'
ncol(x)
```

```
## S4 method for signature 'float32'
NROW(x)

## S4 method for signature 'float32'
NCOL(x)

## S4 method for signature 'float32'
dim(x)

## S4 method for signature 'float32'
length(x)

## S4 replacement method for signature 'float32'
dim(x) <- value
```

Arguments

x	A float vector/matrix.
value	The right hand side for the "setter" (dim<-).

Value

The requested integer values.

Examples

```
library(float)

s = flrunif(10, 3)
dim(s)
nrow(s)
ncol(s)
```

---

eigen *eigen*

Description

Solve a system of equations or invert a float matrix.

Usage

```
## S4 method for signature 'float32'
eigen(x, symmetric, only.values = FALSE, EISPACK = FALSE)
```

Arguments

x	A float vector/matrix.
symmetric	Is the matrix symmetric? If not, it will be tested for symmetry with `isSymmetric()`. Note that only symmetric matrices are supported at this time.
only.values	Should only the values (and not the vectors) be returned?
EISPACK	Ignored.

Value

A list containing the values and optionally vectors, each stored as floats.

Examples

```
library(float)

s = flrunif(10, 3)
cp = crossprod(s)

eigen(cp)
```

---

extremes *extremes*

Description

Min/max values for any combination of float/numeric vector(s)/matri[x|ces].

Usage

```
## S4 method for signature 'float32'
min(x, ..., na.rm = FALSE)

## S4 method for signature 'float32'
max(x, ..., na.rm = FALSE)

## S4 method for signature 'float32'
which.min(x)

## S4 method for signature 'float32'
which.max(x)
```

Arguments

x	A float matrix.
...	Additional elements (numeric/float vectors/matrices) to sum.
na.rm	should NA's be removed?

Details

If there are any elements in ..., all elements in the list will first be summed in their native precision, then converted to double precision so they can be combined with `base::sum()`. The final result will be cast to single precision if ... contains only integer and/or float objects. Otherwise, the return will be double precision.

Value

A single value.

Examples

```
library(float)
x = flrunif(10, 3)

min(x)
min(x, 1)
```

`float` *float*

Description

An analogue to `integer()` and `double()` for preallocation.

Usage

```
float(length = 0, nrow, ncol)
```

Arguments

`length`	Input data of type integer.
`nrow`, `ncol`	Number of rows/columns if a matrix return is desired. See details section for more information.

Details

If both of `nrow` and `ncol` are specified, then `length` is ignored, and the retor is a matrix. If one (but not the other) of `nrow` or `ncol` is given, then the function errors. Otherwise, a vector of length `length` is returned.

Value

A float vector/matrix of 0's.

Examples

```
library(float)

float(10)
float(nrow=2, ncol=3)
```

`float32` *float32*

Description

A `float32` class constructor. For developers only.

Usage

```
float32(x)
```

Arguments

`x`	Input data of type integer.

Details

Wraps the integer-type data in the `float32` S4 class, so that the data will be interpreted as 32-bit floats.

If instead you merely want to convert numeric/double data to float type, instead you should call `fl(x)`.

Value

A `float32` class object.

`float32-class` *Class float32*

Description

An S4 container for 32-bit float vector/matrix objects.

Slots

`Data` A vector or matrix of integers.

hyperbolic *Hyperbolic functions*

Description

Hyperbolic functions.

Usage

```
## S4 method for signature 'float32'
sinh(x)

## S4 method for signature 'float32'
cosh(x)

## S4 method for signature 'float32'
tanh(x)

## S4 method for signature 'float32'
asinh(x)

## S4 method for signature 'float32'
acosh(x)

## S4 method for signature 'float32'
atanh(x)
```

Arguments

x	A float vector/matrix.

Value

A float vector/matrix of the same dimensions as the input.

Examples

```
## Not run:
library(float)

x = flrunif(10)
sinh(x)

## End(Not run)
```

`is.float` *is.float*

Description

Tests if argument is a float matrix.

Usage

```
is.float(x)
```

Arguments

`x`	An R object.

Details

`is.float()` and `is.float()` are different names for the same function.

Value

A logical value.

Examples

```
library(float)

x = matrix(0, 5, 5)
s = flrunif(10, 3)
is.float(x)
is.float(s)
```

`isSymmetric` *isSymmetric*

Description

Test if a float matrix is symmetric.

Usage

```
## S4 method for signature 'float32'
isSymmetric(object, ...)
```

Arguments

object	A float vector/matrix.
...	Ignored.

Value

A logical value.

Examples

```
library(float)

s = flrunif(10, 3)
isSymmetric(s)

cp = crossprod(s)
isSymmetric(s)
```

---

log *Logarithms and Exponentials*

Description

exp/log functions.

Usage

```
## S4 method for signature 'float32'
exp(x)

## S4 method for signature 'float32'
expm1(x)

## S4 method for signature 'float32'
log(x, base = exp(1))

## S4 method for signature 'float32'
log10(x)

## S4 method for signature 'float32'
log2(x)
```

Arguments

x	A float vector/matrix.
base	The logarithm base.

Value

A float vector/matrix of the same dimensions as the input.

Examples

```
## Not run:
library(float)

x = flrunif(10)
log(x)

## End(Not run)
```

`Machine_float` *Machine_float*

Description

Numerical characteristics of the machine for floats. Contains analogues of much of the double precision details of `.Machine`.

Usage

```
Machine_float
```

Format

A list containing:

- `float.eps` epsilon
- `float.xmin` smallest non-zero float
- `float.xmax` largest non-inf float
- `float.base` radix
- `float.digits` the number of bits for the sign+significand
- `float.exponent` number of bits for the exponent
- `float.min.exp` "largest negative" (smallest) integer for the exponent that generates a normalized floating-point number
- `float.max.exp` largest integer for the exponent that generates a normalized floating-point number

Details

Values are taken directly from `float.h`.

mathis *Finite, infinite, and NaNs*

Description

Finite, infinite, and NaNs.

Usage

```
## S4 method for signature 'float32'
is.finite(x)

## S4 method for signature 'float32'
is.infinite(x)

## S4 method for signature 'float32'
is.nan(x)
```

Arguments

x	A float vector/matrix.

Value

An integer vector/matrix of the same dimensions as the input.

Examples

```
## Not run:
library(float)

x = flrnorm(10)
is.nan(sqrt(x))

## End(Not run)
```

matmult *matmult*

Description

Matrix multiplication for numeric/float matrices.

Usage

```
## S4 method for signature 'float32,float32'
x %*% y

## S4 method for signature 'float32,matrix'
x %*% y

## S4 method for signature 'matrix,float32'
x %*% y
```

Arguments

x, y	Numeric/float matrices.

Details

If a numeric matrix is multiplied against a float matrix, then if the "numeric" matrix is integers, the integers are promoted to floats. Otherwise, the float matrix is promoted to doubles.

Value

A matrix of the same type as the highest precision input.

Examples

```
library(float)

s1 = flrunif(5, 5)
s2 = flrunif(5, 2)
x = matrix(1:25, 5)

s1 %*% s2 # float

storage.mode(x) # integer
x %*% s2 # float

storage.mode(x) = "double"
x %*% s2 # double
```

---

miscmath *Miscellaneous mathematical functions*

Description

Miscellaneous mathematical functions.

Usage

```
## S4 method for signature 'float32'
abs(x)

## S4 method for signature 'float32'
sqrt(x)
```

Arguments

x	A float vector/matrix.

Value

A float vector/matrix of the same dimensions as the input.

Examples

```
## Not run:
library(float)

x = flrunif(10)
sqrt(x)

## End(Not run)
```

---

na *NA*

Description

NA utilities.

Usage

```
## S4 method for signature 'float32'
is.na(x)

## S4 method for signature 'float32'
na.omit(object, ...)

## S4 method for signature 'float32'
na.exclude(object, ...)
```

Arguments

x, object	A float vector/matrix.
...	Ignored.

Examples

```
library(float)

s = flrunif(10, 3)
is.na(s)
```

NA_float_ *NA_float_*

Description

32-bit float NA

Usage

```
NA_float_
```

Format

A 32-bit float.

---

names *names*

Description

"name" setter/getters.

Usage

```
## S4 method for signature 'float32'
names(x)

## S4 replacement method for signature 'float32'
names(x) <- value

## S4 method for signature 'float32'
rownames(x)

## S4 replacement method for signature 'float32'
rownames(x) <- value

## S4 method for signature 'float32'
colnames(x)
```

```
## S4 replacement method for signature 'float32'
colnames(x) <- value

## S4 method for signature 'float32'
dimnames(x)

## S4 replacement method for signature 'float32'
dimnames(x) <- value
```

Arguments

x	A float vector/matrix.
value	Replacement value.

NaNf *NaNf*

Description

32-bit float NaN

Usage

```
NaNf
```

Format

A 32-bit float.

---

norm *norm*

Description

Compute matrix norm.

Usage

```
## S4 method for signature 'float32,ANY'
norm(x, type = c("O", "I", "F", "M", "2"))
```

Arguments

x	A float vector/matrix.
type	"O"-ne, "I"-nfinity, "F"-robenius, "M"-ax modulus, and "2" norms.

Value

A single float.

Examples

```
library(float)

s = flrunif(10, 3)
norm(s, type="O")
```

`print-float32` *print-float32*

Description

Print methods for float vector/matrices.

Usage

```
## S4 method for signature 'float32'
print(x, ...)

## S4 method for signature 'float32'
show(object)
```

Arguments

`x, object`	A float vector/matrix.
`...`	Additional arguments to `print()` (see details section for more information).

Details

The printer works by

Examples

```
library(float)

s = flrunif(10, 3)
print(s)
s
```

qr *QR*

Description

QR factorization and related functions.

Usage

```
## S4 method for signature 'float32'
qr(x, tol = 1e-07, ...)

## S4 method for signature 'ANY'
qr.Q(qr, complete = FALSE, Dvec)

## S4 method for signature 'ANY'
qr.R(qr, complete = FALSE)

## S4 method for signature 'ANY'
qr.qy(qr, y)

## S4 method for signature 'ANY'
qr.qty(qr, y)
```

Arguments

x	A float matrix.
tol	The tolerance for determining numerical column rank.
...	Ignored.
qr	Output of qr().
complete	Should the complete or truncated factor be returned?
Dvec	Vector of diagonals to use when re-constructing Q (default is 1's).
y	A vector/matrix or right hand sides (int, float, or double).

Details

The factorization is performed by the LAPACK routine sgeqp3(). This should be similar to calling qr() on an ordinary R matrix with the argument LAPACK=TRUE. Calling qr(x, LAPACK=FALSE) on a double precision matrix 'x' (the default) will not be comparable in performance (it is much slower) or numerics to calling qr(s) where 's' is single a float matrix.

Examples

```
library(float)

x = flrunif(10, 3)
qr(x)
```

---

rand *Generators*

Description

Random float vector/matrix generators. `flrunif()` produces uniform random values. `flrnorm()` produces random normal values. `flrand()` will accept an arbitrary generator. See the details section for more information.

Usage

```
flrunif(m, n, min = 0, max = 1)

flrnorm(m, n, mean = 0, sd = 1)

flrand(generator, m, n, ...)
```

Arguments

`m`, `n`	The dimensions of the matrix/vector. `m` must be specified. If `n` is not, then the return is a vector.
`min`, `max`	Minimum and maximum values for the uniform generator.
`mean`, `sd`	Mean and standard deviation values for the normal generator.
`generator`	A generating function, such as `rnorm`, or even something custom defined.
`...`	Additional arguments passed to the generator. For example, if `runif` is passed as `generator`, then you might additionally pass `max=10`.

Details

For `flrunif()` and `flrnorm()`, the data is produced without a double precision copy. That is, it is not (computationally) equivalent to `fl(matrix(runif(...)))`, though the operations are conceptually the same. For these, To produce a vector instead of a matrix, leave argument `n` blank. Setting `n=1` will produce an mx1 matrix.

For `flrand()`, the data is generated in double precision in 4KiB batches and copied over to a preallocated vector. This will be slower than generating all of the data up front and copying it, although it uses far less memory most of the time. So you can think of `flrunif()` and `flrnorm()` as highly optimized versions of `flrand()` for uniform and normal generators specifically.

Examples

```
library(float)

flrunif(10) # length 10 vector
flrunif(10, 1) # 10x1 matrix
flrunif(10, min=10, max=20)

flrand(runif, 10) # conceptually the same as flrunif(10)

mygen = function(n) sample(1:5, n, replace=TRUE)
flrand(mygen, 30)
```

---

rcond *rcond*

Description

Compute matrix norm.

Usage

```
## S4 method for signature 'float32'
rcond(x, norm = c("O", "I", "1"), triangular = FALSE, ...)
```

Arguments

x	A float vector/matrix.
norm	"O"-ne or "I"-nfinity norm.
triangular	Should only the lower triangle be used?
...	Additional arguments.

Value

A single float.

Examples

```
library(float)

s = flrunif(10, 3)
rcond(s)
```

rep *rep*

Description

Replicate elements of a float vector/matrix.

Usage

```
## S3 method for class 'float32'
rep(x, ...)
```

Arguments

x	A float matrix.
...	Additional arguments (passed to base::rep).

Value

A float vector.

Examples

```
library(float)
x = fl(matrix(1:6, 3, 2))

rep(x, 5)
```

round *Round*

Description

Rounding functions.

Usage

```
## S4 method for signature 'float32'
ceiling(x)

## S4 method for signature 'float32'
floor(x)

## S4 method for signature 'float32'
trunc(x, ...)

## S4 method for signature 'float32'
round(x, digits = 0)
```

Arguments

x	A float vector/matrix.
...	ignored
digits	The number of digits to use in rounding.

Value

A float vector/matrix of the same dimensions as the input.

Examples

```
library(float)

x = flrnorm(10)
floor(x)
```

---

scale *scale*

Description

Center/scale a float vector/matrix.

Usage

```
## S4 method for signature 'float32'
scale(x, center = TRUE, scale = TRUE)
```

Arguments

x	A float vector/matrix.
center, scale	Logical

Details

Only logical `center` and `scale` parameters are accepted at this time.

Value

A float matrix.

Examples

```
library(float)

s = flrunif(10, 3)
scale(s)
```

`sign` *sign*

Description

Sign.

Usage

```
## S4 method for signature 'float32'
sign(x)
```

Arguments

`x`	A float vector/matrix.

Value

A float vector/matrix.

`solve` *solve*

Description

Solve a system of equations or invert a float matrix.

Usage

```
## S4 method for signature 'float32'
solve(a, b, ...)
```

Arguments

`a, b`	A float vector/matrix.
`...`	Ignored.

Value

A float matrix if inverting. If solving a system, a float vector if given one "right hand side", and a float matrix otherwise (just like R).

Examples

```
library(float)

s = flrunif(10, 3)
cp = crossprod(s)
solve(cp)

y = fl(1:3)
solve(cp, y)
```

---

specialmath *Special mathematical functions*

Description

Special mathematical functions.

Usage

```
## S4 method for signature 'float32'
gamma(x)

## S4 method for signature 'float32'
lgamma(x)
```

Arguments

x	A float vector/matrix.

Value

A float vector/matrix of the same dimensions as the input.

Examples

```
## Not run:
library(float)

x = flrunif(10)
lgamma(x)

## End(Not run)
```

sum *sum*

Description

Sums any combination of float/numeric vector(s)/matri[x|ces].

Usage

```
## S4 method for signature 'float32'
sum(x, ..., na.rm = FALSE)
```

Arguments

x	A float matrix.
...	Additional elements (numeric/float vectors/matrices) to sum.
na.rm	should NA's be removed?

Details

If there are any elements in ..., all elements in the list will first be summed in their native precision, then converted to double precision so they can be combined with `base::sum()`. The final result will be cast to single precision if ... contains only integer and/or float objects. Otherwise, the return will be double precision.

Value

A single value.

Examples

```
library(float)
x = flrunif(10, 3)

sum(x)
sum(x, 1)
```

svd *SVD*

Description

SVD factorization.

Usage

```
## S4 method for signature 'float32'
La.svd(x, nu = min(n, p), nv = min(n, p))

## S4 method for signature 'float32'
svd(x, nu = min(n, p), nv = min(n, p), LINPACK = FALSE)
```

Arguments

x	A float matrix.
nu, nv	The number of left/right singular vectors to return.
LINPACK	Ignored

Details

The factorization is performed by the LAPACK routine `sgesdd()`.

Examples

```
library(float)

x = flrunif(10, 3)
svd(x)
```

sweep *sweep*

Description

Sweep a vector through a float matrix.

Usage

```
## S4 method for signature 'float32'
sweep(x, MARGIN, STATS, FUN = "-", check.margin = TRUE, ...)
```

Arguments

x	A float vector/matrix.
MARGIN	1 (rows) or 2 (columns)
STATS	Vector to sweep out.
FUN	Sweeping function; must be one of "+", "-", "*", or "/".
check.margin	Should x/STATS margin lengths be checked?
...	Theoretically these are additional arguments passed to an arbitrary function. However, we only support basic arithmetic, so they are ignored.

Details

Note that if the length of STATS does not recycle exactly across MARGIN, the results here will differ slightly from the results of base R.

Value

A matrix of the same type as the highest precision input.

Examples

```
library(float)

s = flrunif(10, 3)
sweep(s, 2, fl(1))
```

---

trig *Trigonometric functions*

Description

Basic trig functions.

Usage

```
## S4 method for signature 'float32'
sin(x)

## S4 method for signature 'float32'
cos(x)

## S4 method for signature 'float32'
tan(x)

## S4 method for signature 'float32'
asin(x)
```

```
## S4 method for signature 'float32'
acos(x)

## S4 method for signature 'float32'
atan(x)
```

Arguments

x	A float vector/matrix.

Value

A float vector/matrix of the same dimensions as the input.

Examples

```
## Not run:
library(float)

x = flrunif(10)
sin(x)

## End(Not run)
```

---

xpose *xpose*

Description

Transpose a float vector/matrix.

Usage

```
## S4 method for signature 'float32'
t(x)
```

Arguments

x	A float vector/matrix.

Value

A float vector/matrix.

Examples

```
library(float)

s = flrunif(10, 3)
dim(s)
ts = t(s)
dim(ts)
```

Index

∗ **datasets**
Machine_float, 23
NA_float_, 27
NaNf, 28
∗ **package**
float-package, 3
*,BaseLinAlg,float32-method (arithmetic), 3
*,float32,BaseLinAlg-method (arithmetic), 3
*,float32,float32-method (arithmetic), 3
+,BaseLinAlg,float32-method (arithmetic), 3
+,float32,BaseLinAlg-method (arithmetic), 3
+,float32,float32-method (arithmetic), 3
-,BaseLinAlg,float32-method (arithmetic), 3
-,float32,BaseLinAlg-method (arithmetic), 3
-,float32,float32-method (arithmetic), 3
/,BaseLinAlg,float32-method (arithmetic), 3
/,float32,BaseLinAlg-method (arithmetic), 3
/,float32,float32-method (arithmetic), 3
<,BaseLinAlg,float32-method (arithmetic), 3
<,float32,BaseLinAlg-method (arithmetic), 3
<,float32,float32-method (arithmetic), 3
<=,BaseLinAlg,float32-method (arithmetic), 3
<=,float32,BaseLinAlg-method (arithmetic), 3
<=,float32,float32-method (arithmetic), 3
==,BaseLinAlg,float32-method (arithmetic), 3
==,float32,BaseLinAlg-method (arithmetic), 3
==,float32,float32-method (arithmetic), 3
>,BaseLinAlg,float32-method (arithmetic), 3
>,float32,BaseLinAlg-method (arithmetic), 3
>,float32,float32-method (arithmetic), 3
>=,BaseLinAlg,float32-method (arithmetic), 3
>=,float32,BaseLinAlg-method (arithmetic), 3
>=,float32,float32-method (arithmetic), 3
[,float32-method (bracket), 8
[<-,float32-method (bracket), 8
%*%,float32,float32-method (matmult), 24
%*%,float32,matrix-method (matmult), 24
%*%,matrix,float32-method (matmult), 24
^,BaseLinAlg,float32-method (arithmetic), 3
^,float32,BaseLinAlg-method (arithmetic), 3
^,float32,float32-method (arithmetic), 3

abs,float32-method (miscmath), 25
acos,float32-method (trig), 39
acosh,float32-method (hyperbolic), 20
arithmetic, 3
as.data.frame.float32 (converters), 12
as.double.float32 (converters), 12
as.float (converters), 12
as.integer.float32 (converters), 12
as.matrix.float32 (converters), 12
as.numeric,float32-method (converters), 12
as.vector.float32 (converters), 12
asin,float32-method (trig), 39
asinh,float32-method (hyperbolic), 20

atan,float32-method (trig), 39
atanh,float32-method (hyperbolic), 20

backsolve, 6
backsolve,BaseLinAlg,float32-method (backsolve), 6
backsolve,float32,BaseLinAlg-method (backsolve), 6
backsolve,float32,float32-method (backsolve), 6
bind, 7
bracket, 8

c, 9
c,float32-method (c), 9
cbind.float32 (bind), 7
ceiling,float32-method (round), 33
chol, 9
chol,float32-method (chol), 9
chol2inv, 10
chol2inv,float32-method (chol2inv), 10
colMeans,float32-method (colsums), 11
colnames,float32-method (names), 27
colnames<-,float32-method (names), 27
colsums, 11
colSums,float32-method (colsums), 11
comparison, 12
converters, 12
cos,float32-method (trig), 39
cosh,float32-method (hyperbolic), 20
crossprod, 14
crossprod,Mat-method (crossprod), 14

dbl (converters), 12
diag, 15
diag,float32-method (diag), 15
dim,float32-method (dims), 15
dim<-,float32-method (dims), 15
dimnames,float32-method (names), 27
dimnames<-,float32-method (names), 27
dims, 15

eigen, 16
eigen,float32-method (eigen), 16
exp,float32-method (log), 22
expm1,float32-method (log), 22
extremes, 17

fl (converters), 12
float, 18
float-package, 3
float32, 19
float32-class, 19
floor,float32-method (round), 33
flrand (rand), 31
flrnorm (rand), 31
flrunif (rand), 31
forwardsolve,BaseLinAlg,float32-method (backsolve), 6
forwardsolve,float32,BaseLinAlg-method (backsolve), 6
forwardsolve,float32,float32-method (backsolve), 6

gamma,float32-method (specialmath), 36

hyperbolic, 20

int (converters), 12
is.finite,float32-method (mathis), 24
is.float, 21
is.infinite,float32-method (mathis), 24
is.na,float32-method (na), 26
is.nan,float32-method (mathis), 24
isSymmetric, 21
isSymmetric,float32-method (isSymmetric), 21

La.svd,float32-method (svd), 38
length,float32-method (dims), 15
lgamma,float32-method (specialmath), 36
log, 22
log,float32-method (log), 22
log10,float32-method (log), 22
log2,float32-method (log), 22

Machine_float, 23
mathis, 24
matmult, 24
max,float32-method (extremes), 17
min,float32-method (extremes), 17
miscmath, 25

na, 26
NA_float_, 27
names, 27
names,float32-method (names), 27
names<-,float32-method (names), 27
NaNf, 28

NCOL,float32-method *(dims)*, 15
ncol,float32-method *(dims)*, 15
norm, 28
norm,float32,ANY-method *(norm)*, 28
NROW,float32-method *(dims)*, 15
nrow,float32-method *(dims)*, 15

print,float32-method *(print-float32)*, 29
print-float32, 29

qr, 30
qr,float32-method *(qr)*, 30
qr.Q,ANY-method *(qr)*, 30
qr.qty,ANY-method *(qr)*, 30
qr.qy,ANY-method *(qr)*, 30
qr.R,ANY-method *(qr)*, 30

rand, 31
rbind.float32 *(bind)*, 7
rcond, 32
rcond,float32,ANY-method *(rcond)*, 32
rcond,float32-method *(rcond)*, 32
rep, 33
round, 33
round,float32-method *(round)*, 33
rowMeans,float32-method *(colsums)*, 11
rownames,float32-method *(names)*, 27
rownames<-,float32-method *(names)*, 27
rowSums,float32-method *(colsums)*, 11

scale, 34
scale,float32-method *(scale)*, 34
show,float32-method *(print-float32)*, 29
sign, 35
sign,float32-method *(sign)*, 35
sin,float32-method *(trig)*, 39
sinh,float32-method *(hyperbolic)*, 20
solve, 35
solve,float32-method *(solve)*, 35
specialmath, 36
sqrt,float32-method *(miscmath)*, 25
storage.mode,float32-method
(converters), 12
sum, 37
sum,float32-method *(sum)*, 37
svd, 38
svd,float32-method *(svd)*, 38
sweep, 38
sweep,float32-method *(sweep)*, 38

t,float32-method *(xpose)*, 40
tan,float32-method *(trig)*, 39
tanh,float32-method *(hyperbolic)*, 20
tcrossprod,Mat-method *(crossprod)*, 14
trig, 39
trunc,float32-method *(round)*, 33
typeof,float32-method *(converters)*, 12

which.max,float32-method *(extremes)*, 17
which.min,float32-method *(extremes)*, 17

xpose, 40