Package: distplyr (via r-universe)

June 4, 2026

Type Package

Title Manipulate and Combine Probability Distributions

Version 0.2.0

Description Go beyond standard probability distributions such as the Normal or Exponential by combining, shifting, maximizing, and otherwise transforming distributions with simple, verb-based functions. Provides easy access to a broader space of distributions more representative of real-world systems such as river flows or insurance claims. Part of the probaverse framework of packages to support advanced statistical modeling and simulations with an intuitive workflow.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports ellipsis, rlang, vctrs, stats, checkmate, distionary

Suggests covr, knitr, rmarkdown, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://distplyr.probaverse.com/, https://github.com/probaverse/distplyr

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

Date/Publication 2025-12-01 17:58:55 UTC

RemoteUrl https://github.com/probaverse/distplyr

RemoteRef HEAD

RemoteSha ecaf7473b126be2fb759d55e7487708ce93d2a59

Contents

`flip` *Linear and Reciprocal Transformations*

Description

Transform distributions using location shifts, scaling, negation, and reciprocals. If `X` is a random variable following a distribution, these functions return the distribution of the transformed variable.

Usage

```
flip(distribution)

invert(distribution)

multiply(distribution, constant)

shift(distribution, constant)
```

Arguments

`distribution`	A probability distribution.
`constant`	A numeric value for shifting or scaling.

Value

A transformed distribution.

Functions vs Operators

These transformations can be applied using named functions or arithmetic operators:

- `shift(d, a)` or `d + a` - Returns distribution of `X + a`
- `multiply(d, a)` or `d * a` - Returns distribution of `X * a`
- `flip(d)` or `-d` - Returns distribution of `-X`
- `invert(d)` or `1 / d` - Returns distribution of `1 / X`

For complete documentation of operator usage, see `Ops.dst()`.

Special Cases

Negation in multiplication: When `multiply()` receives a negative constant, it internally calls `flip()` on the result of multiplying by the absolute value.

Inversion constraint: `invert()` requires that the distribution has no mass at zero (i.e., `P(X = 0) = 0`). An error is returned if this condition is violated.

Simplifications

These functions apply automatic simplifications when possible. For example:

- Shifting a Normal distribution returns another Normal distribution
- Multiplying a Uniform distribution returns another Uniform distribution
- Flipping a symmetric distribution may preserve its form

More simplifications will be added in future versions of distplyr.

See Also

`Ops.dst()` for arithmetic operators including `+`, `-`, `*`, `/`.

Examples

```
d_pois <- distionary::dst_pois(1.1)
d_norm <- distionary::dst_norm(4, 1)
d_unif <- distionary::dst_unif(0, 1)

# Shifting
shift(d_pois, 1)
d_pois + 1           # Equivalent using operator

# Scaling
multiply(d_unif, 2)
d_unif * 2           # Equivalent using operator

# Negation
flip(d_norm)
-d_norm              # Equivalent using operator

# Inversion
d_positive <- distionary::dst_unif(1, 2)
invert(d_positive)
1 / d_positive       # Equivalent using operator

# Combine multiple operations
4 - 2 * d_pois
multiply(flip(multiply(d_pois, 2)), -1) + 4  # Equivalent
```

Math.dst *Mathematical Transformations for Distributions*

Description

Apply mathematical functions like `log()` and `exp()` to probability distributions. If `X` is a random variable following a distribution, these functions return the distribution of the transformed variable.

Usage

```
## S3 method for class 'dst'
Math(x, ...)
```

Arguments

`x`	A probability distribution object.
`...`	Additional arguments passed to specific methods. For `log()`, you can specify `base` (defaults to `exp(1)` for natural log).

Details

These S3 methods extend base R functions to work with distributions.

Value

A transformed distribution object.

Supported Functions

`log(x, base = exp(1))` Returns the distribution of `log(X)`. The base can be specified (defaults to natural log). An error is returned if the distribution has non-positive values as possible outcomes.

`log10(x)` Returns the distribution of `log10(X)`. Equivalent to `log(x, base = 10)`.

`exp(x)` Returns the distribution of `exp(X)`.

`sqrt(x)` Returns the distribution of `sqrt(X)`. Equivalent to `x^0.5`. Requires all values to be non-negative.

Power Operator

The power operator `^` also works with distributions (see `Ops.dst()`). When raising a distribution to a numeric power (e.g., `dst^2`), it uses the relationship `X^a = exp(a * log(X))`, combining both exponential and logarithmic transformations.

See Also

- `Ops.dst()` for the `^` operator and other arithmetic operations
- `shift()`, `multiply()`, `flip()`, `invert()` for linear transformations

Examples

```
# Logarithmic transformations
d <- distionary::dst_unif(1, 10)
log(d)              # Natural log
log(d, base = 10)   # Log base 10
log10(d)            # Also log base 10
sqrt(d)             # Square root of uniform

# Exponential transformation
d2 <- distionary::dst_norm(0, 1)
d3 <- distionary::dst_beta(5, 4)
exp(d2)             # Log-normal distribution
exp(d3)             # No simplification

# These can be combined
log(exp(d2))        # Returns back to normal distribution
log(exp(d3))        # Still returns d3.
5^(log(d3, base = 5)) # Still returns d3.
```

`maximize` *Extremum of Several Distributions*

Description

For a collection of distributions, obtain the distributions of the maximum (`maximize()`) and minimum (`minimize()`) from independent draws of each component distribution.

Aliases `maximise()` and `minimise()` are also provided.

Usage

```
maximize(
  ...,
  draws = 1,
  na_action_dst = c("null", "drop", "fail"),
  na_action_draws = c("null", "drop", "fail")
)

minimize(
  ...,
  draws = 1,
  na_action_dst = c("null", "drop", "fail"),
  na_action_draws = c("null", "drop", "fail")
)
```

Arguments

`...`	Distribution objects, or list of distributions.

draws
: Number of draws from each distribution considered in the maximum (possibly not integer, but never negative). Either a single numeric applying to all distributions in `...`, or a vector matching the number of distributions in `...`.

na_action_dst, na_action_draws
: What should be done with Null distributions in `...` and `NA` in `draws`? Character vector of length 1: one of "fail", "null" (default), or "drop". See details.

Details

To give an example of what distribution is returned, if X1 and X2 are two random variables with distributions D1 and D2 respectively, then `maximize(D1, D2, draws = c(2, 3))` returns the distribution of `max(X1, X1, X2, X2, X2)`.

Distributions in `...` and the `draws` vector are recycled to have the same length, but only if one of them has length 1 (via `vctrs::vec_recycle_common()`).

`na_action_dst` and `na_action_draws` specify the NA action for distributions and `draws`. "NA" here means either `NA` in the `draws` vector, or a Null distribution (`distionary::dst_null()`) in the distributions. Options are, in order of precedence:

- `"fail"`: Throw an error in the presence of NAs.
- `"null"`: Return a Null distribution in the presence of NAs.
- `"drop"`: Remove distribution-weight pairs having an NA value

Simplifications made in these functions include the following:

- If any distributions are entirely to the left (right) of others, then they are removed from consideration in `maximize()` (`minimize()`).
- If all Finite distributions are input, the result is also a Finite distribution.
- If the same distribution is input multiple times, their corresponding draws are summed.

Value

A distribution corresponding to the maximum or minimum.

Examples

```
library(distionary)
# One is always more extreme than the other in this case.
d1 <- dst_unif(-1, 2)
d2 <- dst_unif(5, 6)
maximize(d1, d2) # d2
minimize(d1, d2) # d1

# Visualizing the maximum and minimum
d3 <- dst_norm(4, 1)
d4 <- dst_exp(0.3)

dmax <- maximize(d3, d4, draws = 1:2)
dmin <- minimize(d3, d4, draws = 1:2)
```

```
# Maximum
plot(d3, col = "blue", lty = 2, from = 0, to = 14)
plot(d4, col = "red", lty = 2, add = TRUE)
plot(dmax, add = TRUE, n = 1000)
legend(
 "topright",
 legend = c("Maximum", "N(4,1)", "Exp(0.3)"),
 col = c("black", "blue", "red"),
 lty = c(1, 2, 2)
)

# Minimum
plot(d3, col = "blue", lty = 2, from = 0, to = 10)
plot(d4, col = "red", lty = 2, add = TRUE)
plot(dmin, add = TRUE, n = 1000)
legend(
  "topright",
  legend = c("Minimum", "N(4,1)", "Exp(0.3)"),
  col = c("black", "blue", "red"),
  lty = c(1, 2, 2)
)
```

`mix` *Mixture Distributions*

Description

Create a mixture distribution, which can be thought of as an average of multiple distributions (in terms of their CDF, density, PMF, or survival functions, for example). Data drawn from a mixture distribution involves two steps: first randomly selecting the distribution to draw from, followed by the random selection from that distribution.

Usage

```
mix(
  ...,
  weights = 1,
  na_action_dst = c("null", "drop", "fail"),
  na_action_w = c("null", "drop", "fail")
)
```

Arguments

`...`	Distribution objects, or list of distributions.
`weights`	Vector of weights corresponding to the distributions; or, single numeric for equal weights. When normalized, they correspond to the probabilities of selecting each distribution.

na_action_dst, na_action_w
: What should be done with null distributions in ... and NA in weights? Character vector of length 1: one of "fail", "null" (default), or "drop". See details.

Details

Distributions in ... and the weights vector are recycled to have the same length, but only if one of them has length 1 (via vctrs::vec_recycle_common()).

na_action_dst and na_action_w specify the NA action for distributions and weights. "NA" here means either NA in the weights vector, or a Null distribution (distionary::dst_null()) in the distributions. Options are, in order of precedence:

- "fail": Throw an error in the presence of NAs.
- "null": Return a Null distribution in the presence of NAs.
- "drop": Remove distribution-weight pairs having an NA value

Value

A mixture distribution.

Examples

```
library(distionary)
a <- dst_norm(0, 1)
b <- dst_norm(5, 2)
m1 <- mix(a, b, weights = c(1, 4))
plot(a, col = "red", lty = 2, from = -3, to = 11)
plot(b, add = TRUE, col = "blue", lty = 2)
plot(m1, add = TRUE)
legend(
  "topright",
   legend = c("Mixture", "N(0,1)", "N(5,2)"),
   col = c("black", "red", "blue"),
   lty = c(1, 2, 2)
)
```

Ops.dst *Arithmetic Operations for Distributions*

Description

Apply arithmetic operators to probability distributions. These operations transform distributions in intuitive ways, treating them similarly to numeric values.

Usage

```
## S3 method for class 'dst'
Ops(e1, e2)
```

Arguments

`e1`	A probability distribution or numeric value.
`e2`	A probability distribution or numeric value.

Details

These S3 methods extend arithmetic operators to work with distributions.

Value

A transformed distribution object.

Supported Operators

`d + a` **or** `a + d` Shifts the distribution by adding constant `a`. Equivalent to `shift()`.

`d - a` Shifts the distribution by subtracting constant `a`. Equivalent to `shift(d, -a)`.

`-d` Flips the distribution (negation). Equivalent to `flip()`.

`d * a` **or** `a * d` Scales the distribution by multiplying by constant `a`. Equivalent to `multiply()`.

`d / a` Scales the distribution by dividing by constant `a`. Equivalent to `multiply(d, 1/a)`.

`a / d` Returns the distribution of `a / X` (reciprocal scaling). For `a = 1`, equivalent to `invert()`.

`d ^ a` Raises the distribution to power `a`. For positive distributions only, computed as `exp(a * log(d))`.

`a ^ d` Returns the distribution of `a^X`. Requires positive base `a`.

Power Operator Details

The power operator `^` deserves special attention:

- When the **base is a distribution** (e.g., `dst_beta(1, 1)^2`), it computes the distribution of `X^a` using the transformation `exp(a * log(X))`. This requires all values in the distribution to be positive.
- When the **exponent is a distribution** (e.g., `2^dst_norm(0, 1)`), it computes the distribution of `a^X` using `exp(X * log(a))`. The base `a` must be positive.

These implementations internally use both `Math.dst()` methods `log()` and `exp()`.

See Also

- `shift()`, `multiply()`, `flip()`, `invert()` for the underlying transformation functions
- `Math.dst()` for `log()`, `exp()`, and `sqrt()` functions

Examples

```
d <- distionary::dst_beta(3, 2)

# Shifting and scaling
d + 10          # Shift right by 10
d * 2           # Scale by 2
3 * d - 5       # Scale then shift

# Power operations
exp(d)          # e^X: exponential of Beta
2^d             # 2^X: base 2 raised to Beta

# With positive distributions
d_pos <- distionary::dst_unif(1, 2)
d_pos^2         # X^2: uniform squared
d_pos^0.5       # sqrt(X): square root
sqrt(d_pos)     # Equivalent to d_pos^0.5
```

Index