Package: dist.structure (via r-universe)

October 28, 2026

Title Structured Random Variables for Reliability System Distributions

Version 0.5.1

Description Extends the 'algebraic.dist' distribution algebra to random variables with internal structure: coherent reliability systems decomposed into components arranged by a structure function (series, parallel, k-out-of-n, bridge, and arbitrary topologies via minimal path sets). Every 'dist_structure' object is a 'dist', so the full distribution algebra (mean, vcov, sampler, surv, cdf) works automatically via default methods that compose component-level distributions through the topology. Adds structural queries: structure function evaluation, minimal path and cut sets, system signature, critical states, dual, Birnbaum structural importance, and system reliability. Topology shortcut constructors (series_dist, parallel_dist, kofn_dist, bridge_dist) produce ready-to-use dists from component dists and a chosen structure.

License MIT + file LICENSE

URL https://github.com/queelius/dist.structure

BugReports https://github.com/queelius/dist.structure/issues

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports algebraic.dist, stats, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown, spelling, withr

VignetteBuilder knitr

Config/testthat/edition 3

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

Date/Publication 2026-10-28 05:52:57 UTC

RemoteUrl https://github.com/queelius/dist.structure

RemoteRef HEAD

RemoteSha fea129e92424cc337ccd8b7eb2475c9b24f3c1bf

Contents

as_dist_structure.dist_structure . . . 3
birnbaum_importance . . . 3
bridge_dist . . . 4
coherent_dist . . . 5
cold_standby_dist . . . 5
component . . . 7
compose_systems.dist_structure . . . 7
consecutive_k_dist . . . 8
critical_states.dist_structure . . . 9
criticality_importance . . . 9
dual.coherent_dist . . . 10
exp_kofn . . . 11
exp_parallel . . . 12
exp_series . . . 13
format.dist_structure . . . 14
gamma_series . . . 14
is_coherent.dist_structure . . . 15
is_dist_structure . . . 16
kofn_dist . . . 16
lognormal_series . . . 17
max_iid . . . 18
mean.cold_standby_dist . . . 19
min_cuts.dist_structure . . . 19
min_iid . . . 20
min_paths.dist_structure . . . 20
ncomponents . . . 21
order_statistic . . . 21
parallel_dist . . . 22
phi.dist_structure . . . 23
print.dist_structure . . . 24
reliability.dist_structure . . . 24
series_dist . . . 25
structural_importance.dist_structure . . . 25
substitute_component.dist_structure . . . 26
surv.dist_structure . . . 26
system_censoring.dist_structure . . . 28
system_lifetime.dist_structure . . . 28
system_signature.dist_structure . . . 29
validate_dist_structure . . . 29
vesely_fussell_importance . . . 30
wei_homogeneous_series . . . 31

wei_kofn 32
wei_series 33

Index **35**

`as_dist_structure.dist_structure` *Coerce to dist_structure*

Description

Wraps a plain `algebraic.dist` distribution in a 1-component series_dist so that it satisfies the dist_structure protocol. Useful for polymorphic code that operates on dist_structure inputs but is also handed unwrapped distributions. A `dist_structure` is returned unchanged.

Usage

```
## S3 method for class 'dist_structure'
as_dist_structure(x, ...)

## S3 method for class 'dist'
as_dist_structure(x, ...)

as_dist_structure(x, ...)
```

Arguments

`x`	A dist or dist_structure.
`...`	Reserved for methods.

Value

A dist_structure object.

`birnbaum_importance` *Birnbaum reliability importance*

Description

The Birnbaum importance of component `j` at component reliabilities `p` is `dR/dp_j = R(p with p_j = 1) - R(p with p_j = 0`
Measures how much the system reliability changes if `j` moves between certain failure and certain success (and, by monotonicity, the rate of change at any intermediate `p_j`).

Usage

```
birnbaum_importance(x, j, p)

## S3 method for class 'dist_structure'
birnbaum_importance(x, j, p)
```

Arguments

x	A dist_structure object.
j	Component index.
p	Numeric vector of length ncomponents(x) or a scalar in [0, 1].

Value

Numeric scalar in [0, 1].

---

bridge_dist *Bridge system distribution*

Description

The classical 5-component bridge reliability network with minimal path sets {1,4}, {2,5}, {1,3,5}, {2,3,4}. Components 1 and 2 are the input side, 4 and 5 the output side, and 3 the cross-link. The bridge is a canonical non-series, non-parallel example used throughout the reliability literature; see Barlow and Proschan (1975, "Statistical Theory of Reliability and Life Testing") for the standard treatment.

Usage

```
bridge_dist(components)
```

Arguments

components	List of 5 dist objects.

Value

A bridge_dist inheriting from coherent_dist.

coherent_dist *Coherent system distribution from minimal path sets*

Description

General-purpose constructor. Users supply a list of minimal path sets (each a vector of component indices) and a list of component distributions (each an `algebraic.dist::dist` object with parameters baked in). The resulting object is a `dist_structure` and `dist`.

Usage

```
coherent_dist(min_paths, components, m = NULL)
```

Arguments

`min_paths`	List of integer vectors; each is a minimal path set.
`components`	List of `dist` objects, length `m`. Each is a fully-parameterized component lifetime distribution.
`m`	Optional integer. Inferred from `components` and `min_paths` if omitted.

Value

An object of class `c("coherent_dist", "dist_structure", "univariate_dist", "dist")`.

Examples

```
# A bridge network with exponential components
sys <- coherent_dist(
  min_paths = list(c(1, 4), c(2, 5), c(1, 3, 5), c(2, 3, 4)),
  components = replicate(5, algebraic.dist::exponential(1), simplify = FALSE)
)
reliability(sys, 0.9)
```

cold_standby_dist *Cold-standby system distribution*

Description

Constructs a distribution representing a cold-standby system: one component active at a time with perfect, instantaneous switching to the next spare upon failure. System lifetime equals the sum of independent component lifetimes.

Estimates `S(t) = P(T_1 + ... + T_m > t)` by simulating `mc` system lifetimes and computing the empirical fraction exceeding `t`. The returned closure caches its sample vector: the first call generates `mc` samples, subsequent calls reuse them as long as `mc` is unchanged (a different `mc` triggers a fresh draw). This makes repeated `S(t)` calls deterministic given the same `mc`.

Computes the CDF by deferring to a fresh `surv` closure. The CDF closure has its own sample cache (independent of any external `surv` closure), so `cdf(x)(t) + surv(x)(t)` is generally not exactly 1 unless callers reuse a single `surv` closure: prefer `S <- surv(x); F_t <- 1 - S(t)` over computing both independently.

Usage

```
cold_standby_dist(components)

## S3 method for class 'cold_standby_dist'
sampler(x, ...)

## S3 method for class 'cold_standby_dist'
surv(x, ...)

## S3 method for class 'cold_standby_dist'
cdf(x, ...)
```

Arguments

`components`	List of `dist` objects representing per-stage component lifetimes.
`x`	A `cold_standby_dist` object.
`...`	Ignored.

Details

Cold standby is not a coherent system in the structure-function sense (its topology is temporal succession, not an order statistic), so the returned object does not inherit `dist_structure`. It IS a `dist` (with `surv`, `cdf`, `sampler`, `mean` available) and exposes `ncomponents()` and `component()` for inspection.

Defaults: `sampler` is exact (sample each component independently and sum); `mean` is exact when every component implements `mean()` (otherwise falls back to Monte Carlo via the sampler); `surv` and `cdf` use Monte Carlo with a default of `1e5` simulated lifetimes (override via the `mc` argument). The returned `surv` / `cdf` closures cache their samples after the first call so subsequent evaluations at different `t` values are deterministic given the same `mc`.

Methods inherited from `algebraic.dist::univariate_dist` that require `density` or `sup` (notably `expectation`, `vcov`) are NOT supported on cold-standby objects out of the box; specialized subclasses with closed-form aggregate distributions (e.g., iid exponential collapses to `Gamma(m, rate)`) should provide their own methods.

For reproducibility across calls to `surv()` itself (i.e., between separately constructed closures), set the RNG seed externally via `set.seed()` before invoking `surv(x)`. Override the method on a subclass with an exact aggregate distribution (e.g., iid exponential collapses to `Gamma(m, rate)`) when an analytic form is available.

Value

An object of class `c("cold_standby_dist", "univariate_dist", "continuous_dist", "dist")`.

Examples

```
# Cold standby of 3 iid Exp(1) components: aggregate is Gamma(3, 1)
sys <- cold_standby_dist(replicate(3,
  algebraic.dist::exponential(1), simplify = FALSE))
mean(sys)  # = 3
```

`component` *The j-th component as a dist*

Description

Returns the j-th component as an `algebraic.dist::dist` object with parameters baked in (so `sampler`, `surv`, etc. work directly on it).

Usage

```
component(x, j, ...)
```

Arguments

`x`	A dist_structure object.
`j`	Integer component index in `1:ncomponents(x)`.
`...`	Implementation-specific arguments (e.g., parameter overrides for lazily parameterized systems).

Value

An object inheriting from `dist`.

`compose_systems.dist_structure` *Compose systems hierarchically*

Description

Produce a new dist_structure by replacing each component of `outer` with a sub-system (either a dist_structure or a plain dist). The composed system's components are the flattened inner components; its min_paths enumerate the Cartesian products of inner min_paths within each outer min_path.

Usage

```
## S3 method for class 'dist_structure'
compose_systems(outer, inner_list)

compose_systems(outer, inner_list)
```

Arguments

`outer`	A dist_structure object.
`inner_list`	A list of length `ncomponents(outer)`. Each element is either a dist_structure or a dist (single-component sub-system).

Details

Computational note: the composed minimal-path enumeration takes the Cartesian product of inner-path choices over each outer path. For an outer system with `p` paths each of length `q`, where each inner has `r` paths, the candidate count grows as `O(p * r^q)` before deduplication. Bridge-of-bridges and similar deeply nested compositions can produce hundreds of candidates; if you find the call slow, build the composed `coherent_dist` directly with a hand-curated `min_paths` list.

Value

A `coherent_dist` representing the composed system.

`consecutive_k_dist` *Consecutive-k-out-of-n system distribution (type G)*

Description

The consecutive-k-out-of-n:G system functions when at least one block of `k` consecutive components all function. Minimal path sets are the `n - k + 1` consecutive blocks of size `k`. (Note: this is the :G variant; the :F variant, "fails when any `k` consecutive fail", has different minimal paths.)

Usage

```
consecutive_k_dist(k, components)
```

Arguments

`k`	Block size.
`components`	List of `dist` objects (length `n`).

Value

A `consecutive_k_dist` inheriting from `coherent_dist`.

critical_states.dist_structure

Critical states of a component

Description

Critical states of a component

Usage

```
## S3 method for class 'dist_structure'
critical_states(x, j)

critical_states(x, j)
```

Arguments

x	A dist_structure object.
j	Component index.

Value

Integer matrix with `m - 1` columns; rows are the states of the other components for which `j` is critical.

criticality_importance

Criticality (Fussell) importance at time t

Description

Probability that component `j` has failed AND is critical given that the system has failed by time `t`. Equals `I_B(j; S(t)) * F_j(t) / F_sys(t)`.

Usage

```
criticality_importance(x, j, t)

## S3 method for class 'dist_structure'
criticality_importance(x, j, t)
```

Arguments

x	A dist_structure object.
j	Component index.
t	Scalar time.

Value

Numeric scalar in `[0, 1]`.

`dual.coherent_dist` *Dual of a coherent_dist: swap cuts and paths*

Description

Overrides the lazy-wrapper default with a proper `coherent_dist`: `min_paths(dual(x))` = `min_cuts(x)`.

Returns a `dual_of_system` object carrying the original structure. `phi_dual(state)` = `1 - phi(original, 1 - state)` is evaluated on demand. All other generics fall through to `dist_structure` defaults.

The dual structure satisfies `phi_dual(state)` = `1 - phi(1 - state)`. The dual of a series system is parallel; the dual of k-out-of-n is (n - k + 1)-out-of-n.

Usage

```
## S3 method for class 'coherent_dist'
dual(x)

## S3 method for class 'dist_structure'
dual(x)

## S3 method for class 'dual_of_system'
dual(x)

dual(x)
```

Arguments

`x`	A dist_structure object.

Value

A dist_structure object representing the dual.

exp_kofn *k-out-of-n system of independent exponential components (closed form)*

Description

Constructs a `dist_structure` for a k-out-of-m system whose components are independent exponentials. Closed-form methods are provided for `surv`, `cdf`, `sampler`, `density`, and `hazard`. `mean` falls back to numerical integration via the `dist_structure` default.

Usage

```
exp_kofn(k, rates)

## S3 method for class 'exp_kofn'
surv(x, ...)

## S3 method for class 'exp_kofn'
sampler(x, ...)

## S3 method for class 'exp_kofn'
hazard(x, ...)

## S3 method for class 'exp_kofn'
density(x, ...)
```

Arguments

`k`	Minimum number of functioning components for system operation.
`rates`	Positive numeric vector of length `m` with `m >= k`.
`x`	An `exp_kofn` object.
`...`	Ignored.

Value

`exp_kofn()` returns an object of class `c("exp_kofn", "kofn_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist")`.

The associated S3 methods return:

- `surv()`, `density()`, `hazard()`: a closure `function(t, ...)`.
- `cdf()` is derived via the `dist_structure` default and returns a closure `function(t, ...)` equal to `1 - surv(x)(t)`.
- `sampler()`: a closure `function(n, ...)` returning `n` random variates from the system lifetime distribution.

Examples

```
sys <- exp_kofn(k = 2, rates = c(1, 2, 3))
algebraic.dist::surv(sys)(1)
```

`exp_parallel` *Parallel of exponential components (closed form)*

Description

Constructs a `dist_structure` representing a parallel system whose components are independent exponentials. Closed-form methods are provided for `surv`, `cdf`, `sampler`, and `mean` (the last via inclusion-exclusion over the `2^m - 1` non-empty component subsets).

Usage

```
exp_parallel(rates)

## S3 method for class 'exp_parallel'
surv(x, ...)

## S3 method for class 'exp_parallel'
sampler(x, ...)

## S3 method for class 'exp_parallel'
mean(x, ...)
```

Arguments

`rates`	Positive numeric vector of length `m`.
`x`	An `exp_parallel` object.
`...`	Ignored.

Value

`exp_parallel()` returns an object of class `c("exp_parallel", "parallel_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist")`.

The associated S3 methods return:

- `surv()`: a closure `function(t, ...)`.
- `cdf()` is derived via the `dist_structure` default and returns a closure `function(t, ...)` equal to `1 - surv(x)(t)`.
- `sampler()`: a closure `function(n, ...)` returning `n` random variates from the system lifetime distribution.
- `mean()`: a numeric scalar (the mean system lifetime, computed in closed form via inclusion-exclusion).

Examples

```
sys <- exp_parallel(c(1, 2, 3))
algebraic.dist::surv(sys)(1)
```

`exp_series` *Series of exponential components (closed form)*

Description

Constructs a `dist_structure` representing a series system whose components are independent exponentials. The system lifetime is itself an `Exp(sum(rates))` distribution; all dist-level queries have closed-form expressions that bypass the general default methods.

Usage

```
exp_series(rates)

## S3 method for class 'exp_series'
surv(x, ...)

## S3 method for class 'exp_series'
sampler(x, ...)

## S3 method for class 'exp_series'
mean(x, ...)

## S3 method for class 'exp_series'
density(x, ...)

## S3 method for class 'exp_series'
hazard(x, ...)
```

Arguments

`rates`	Positive numeric vector of length `m`: per-component exponential rates.
`x`	An `exp_series` object.
`...`	Ignored.

Value

`exp_series()` returns an object of class `c("exp_series", "series_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist")`.

The associated S3 methods return:

- `surv()`, `density()`, `hazard()`: a closure `function(t, ...)` evaluating the named quantity at `t`.

- `cdf()` is derived via the `dist_structure` default and returns a closure `function(t, ...)` equal to `1 - surv(x)(t)`.
- `sampler()`: a closure `function(n, ...)` returning `n` random variates from the system lifetime distribution.
- `mean()`: a numeric scalar (the mean system lifetime).

Examples

```
sys <- exp_series(c(0.5, 0.3, 0.2))
algebraic.dist::surv(sys)(1)  # equals exp(-sum(rates) * 1)
mean(sys)                     # equals 1 / sum(rates)
```

---

format.dist_structure *Format a dist_structure object*

Description

Format a dist_structure object

Usage

```
## S3 method for class 'dist_structure'
format(x, ...)
```

Arguments

`x`	A dist_structure object.
`...`	Ignored.

Value

Character vector suitable for `cat()`.

---

gamma_series *Series of independent Gamma components (closed form)*

Description

Constructs a `dist_structure` for a series system whose components are independent Gammas. Closed-form `surv` is evaluated by the product of per-component upper-tail probabilities; `cdf` is `1 - surv`; `sampler` generates m independent Gammas and takes the min.

Usage

```
gamma_series(shapes, rates)

## S3 method for class 'gamma_series'
surv(x, ...)

## S3 method for class 'gamma_series'
sampler(x, ...)
```

Arguments

shapes	Positive numeric vector of length m: per-component Gamma shape parameters.
rates	Positive numeric vector of length m: per-component Gamma rate parameters.
x	A gamma_series object.
...	Ignored.

Value

gamma_series() returns an object of class c("gamma_series", "series_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist").

The associated S3 methods return:

- surv(): a closure function(t, ...).
- cdf() is derived via the dist_structure default and returns a closure function(t, ...) equal to 1 - surv(x)(t).
- sampler(): a closure function(n, ...) returning n random variates from the system lifetime distribution.

Examples

```
sys <- gamma_series(shapes = c(2, 3), rates = c(1, 2))
algebraic.dist::surv(sys)(1)
```

---

is_coherent.dist_structure

Coherence axiom check

Description

Coherence axiom check

Usage

```
## S3 method for class 'dist_structure'
is_coherent(x)

is_coherent(x)
```

Arguments

x	A dist_structure object.

Value

TRUE if monotone and every component relevant.

---

is_dist_structure *Predicate for dist_structure objects*

Description

Predicate for dist_structure objects

Usage

```
is_dist_structure(x)
```

Arguments

x	Any object.

Value

TRUE if x inherits from "dist_structure".

---

kofn_dist *k-out-of-n system distribution*

Description

A k-out-of-n system functions if at least k of its m components function. Equivalent to the (m - k + 1)-th order statistic of component lifetimes.

Usage

```
kofn_dist(k, components)
```

Arguments

k	Minimum functioning components for system operation (:G).
components	List of dist objects (length m).

Details

This constructor uses the **:G** convention: `k` is the number of components that must remain **functioning** for the system to function. `k = 1` is parallel; `k = m` is series. The companion `kofn` package (which depends on `dist.structure`) uses the **:F** convention, where `k` is the number of failures that trigger system failure; conversion is `k_dist = m - k_kofn + 1`. When in doubt, draw a small example: `kofn_dist(k = 2, ...)` for `m = 3` functions until two of the three components have failed.

Value

A `kofn_dist` inheriting from `coherent_dist`.

See Also

`order_statistic()` for the closely-related order-statistic parameterization.

`lognormal_series` *Series of independent Lognormal components (closed form)*

Description

Constructs a `dist_structure` for a series system whose components are independent Lognormals. Closed-form `surv` is the product of per-component upper-tail probabilities; `cdf` is `1 - surv`; `sampler` generates m independent Lognormals and takes the min.

Usage

```
lognormal_series(meanlogs, sdlogs)

## S3 method for class 'lognormal_series'
surv(x, ...)

## S3 method for class 'lognormal_series'
sampler(x, ...)
```

Arguments

`meanlogs`	Numeric vector of length `m`: per-component meanlog parameters.
`sdlogs`	Positive numeric vector of length `m`: per-component sdlog parameters.
`x`	A `lognormal_series` object.
`...`	Ignored.

Value

`lognormal_series()` returns an object of class `c("lognormal_series", "series_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist")`.

The associated S3 methods return:

- `surv()`: a closure `function(t, ...)`.
- `cdf()` is derived via the `dist_structure` default and returns a closure `function(t, ...)` equal to `1 - surv(x)(t)`.
- `sampler()`: a closure `function(n, ...)` returning `n` random variates from the system lifetime distribution.

Examples

```
sys <- lognormal_series(meanlogs = c(0, 1), sdlogs = c(1, 0.5))
algebraic.dist::surv(sys)(1)
```

`max_iid` *Parallel system of m iid components*

Description

Equivalent to the maximum of `m` iid random variables from `d`. Preserves the parallel topology for structural queries.

Usage

```
max_iid(d, m)
```

Arguments

`d`	A `dist` object.
`m`	Number of components.

Value

A `parallel_dist`.

`mean.cold_standby_dist` *Mean of a cold-standby system: sum of component means*

Description

Computes `sum_j E[T_j]` exactly when every component implements a `mean()` method. Falls back to a Monte Carlo estimate from the sampler when any component lacks an exact mean (e.g., a `dist_structure` component whose `mean` would route through `algebraic.dist::univariate_dist` and require `density` / `sup`, which are not provided on `dist_structure` by default).

Usage

```
## S3 method for class 'cold_standby_dist'
mean(x, ...)
```

Arguments

`x`	A `cold_standby_dist` object.
`...`	Passed to the Monte Carlo fallback as `mc` (default 1e5).

Value

Numeric scalar.

`min_cuts.dist_structure` *Minimal cut sets*

Description

Minimal cut sets

Usage

```
## S3 method for class 'dist_structure'
min_cuts(x)

min_cuts(x)
```

Arguments

`x`	A dist_structure object.

Value

A list of integer vectors.

min_iid *Series system of m iid components*

Description

Equivalent to the minimum of `m` iid random variables from `d`. Unlike `min` in the base distribution algebra, `min_iid` preserves the series topology so structural queries (phi, min_paths, structural_importance) remain available.

Usage

```
min_iid(d, m)
```

Arguments

`d`	A `dist` object (the common component distribution).
`m`	Number of components (positive integer).

Value

A `series_dist`.

Examples

```
sys <- min_iid(algebraic.dist::exponential(1), m = 3)
# System survival at t=0.5 equals (exp(-t))^3 = exp(-1.5)
```

min_paths.dist_structure *Minimal path sets*

Description

Returns the minimal subsets of `1:m` whose joint functioning is sufficient for the system to function. `min_paths` and `phi()` are dual primitives in the `dist_structure` protocol: providing one is enough, the other has an enumerative default.

Usage

```
## S3 method for class 'dist_structure'
min_paths(x)

min_paths(x)
```

Arguments

x	A dist_structure object.

Value

A list of integer vectors.

See Also

phi() for the dual primitive; min_cuts() for the dual topology query (minimal subsets whose joint failure causes system failure).

---

ncomponents *Number of components*

Description

Number of components

Usage

```
ncomponents(x)
```

Arguments

x	A dist_structure object.

Value

A positive integer m, the number of components.

---

order_statistic *k-th order statistic of m iid components*

Description

Constructs a kofn_dist whose system lifetime equals T_(k), the k-th order statistic of m iid draws from d. Under the k-of-m parametrization, the system fails at the (m - k + 1)-th component failure; setting the threshold to m - k + 1 makes the system lifetime equal T_(k).

Usage

```
order_statistic(d, k, m)
```

Arguments

d	A dist object.
k	The order statistic index (1 = min, m = max).
m	Number of iid components.

Details

The internal call is kofn_dist(m - k + 1, ...): this maps the order-statistic index k (where k = 1 is the minimum, k = m is the maximum) to the :G convention used by kofn_dist (where the k_dist argument is the number of functioning components required).

Value

A kofn_dist.

Examples

```
# Median of 5 iid exponentials
sys <- order_statistic(algebraic.dist::exponential(1), k = 3, m = 5)
```

---

parallel_dist *Parallel system distribution*

Description

A parallel system fails only when all components fail. Equivalent to max(components) but preserves topology.

Usage

```
parallel_dist(components)
```

Arguments

components	List of dist objects.

Value

A parallel_dist inheriting from coherent_dist.

`phi.dist_structure` *Structure function*

Description

Evaluate the coherent structure function `phi: {0, 1}^m -> {0, 1}` at a component state vector `state`. By convention `state[j] = 1` means component `j` is functioning; `phi(state) = 1` means the system is functioning.

Usage

```
## S3 method for class 'dist_structure'
phi(x, state)

phi(x, state)
```

Arguments

`x`	A dist_structure object.
`state`	Integer or logical vector of length `ncomponents(x)` in `{0, 1}`.

Details

`phi` and `min_paths()` are dual primitives: if a subclass provides only `phi.<class>()`, the default `min_paths.dist_structure()` enumerates minimal subsets via `phi`; if a subclass provides only `min_paths.<class>()`, the default `phi.dist_structure()` checks whether `state` covers any minimal path. Subclasses providing both must keep them consistent.

Value

Integer scalar, `0` or `1`.

See Also

`min_paths()` for the dual primitive; `validate_dist_structure()` for construction-time checking.

print.dist_structure *Print a dist_structure object*

Description

Print a dist_structure object

Usage

```
## S3 method for class 'dist_structure'
print(x, ...)
```

Arguments

x	A dist_structure object.
...	Passed to `format()`.

Value

x, invisibly.

reliability.dist_structure *System reliability polynomial*

Description

`R(p) = E[phi(X)]` where `X_j ~ Bernoulli(p_j)` independent. The multilinear extension of `phi` to component reliabilities.

Usage

```
## S3 method for class 'dist_structure'
reliability(x, p)

reliability(x, p)
```

Arguments

x	A dist_structure object.
p	Numeric vector of length `ncomponents(x)` or a scalar recycled to all components; values in `[0, 1]`.

Value

Numeric scalar in `[0, 1]`.

series_dist *Series system distribution*

Description

A series system fails if any single component fails. Equivalent to `min(components)` as random variables; unlike `min` in the base algebra, `series_dist` preserves the component decomposition so topology queries work.

Usage

```
series_dist(components)
```

Arguments

`components`	List of `dist` objects.

Value

A `series_dist` inheriting from `coherent_dist`.

Examples

```
sys <- series_dist(replicate(3, algebraic.dist::exponential(1), simplify = FALSE))
algebraic.dist::surv(sys)(0.5)
```

structural_importance.dist_structure *Birnbaum structural importance*

Description

Fraction of the `2^(m - 1)` states of the other components for which component `j` is critical.

Usage

```
## S3 method for class 'dist_structure'
structural_importance(x, j)

structural_importance(x, j)
```

Arguments

`x`	A dist_structure object.
`j`	Component index.

Value

Numeric scalar in `[0, 1]`.

`substitute_component.dist_structure` *Substitute a component*

Description

Return a new dist_structure with the j-th component replaced by `new_component`. Topology is preserved; the returned object is a `coherent_dist` with the same min_paths and the modified component list.

Usage

```
## S3 method for class 'dist_structure'
substitute_component(x, j, new_component)

substitute_component(x, j, new_component)
```

Arguments

`x`	A dist_structure object.
`j`	Component index.
`new_component`	A dist-compatible object to install at position j.

Value

A new dist_structure object.

`surv.dist_structure` *System survival via reliability composition*

Description

Returns a closure `function(t, ...)` where `surv(x)(t)` equals `reliability(x, surv(component(x, j))(t) for each j`. This is the classical identity `S_sys(t) = R(S(t))` realized as a composition.

Returns a closure `function(n, ...)` that draws `n` system lifetimes by sampling each component independently and applying `system_lifetime()` to combine.

`dist_structure` is the virtual S3 class for distributions whose random variable has internal component structure: coherent reliability systems decomposed into components arranged by a structure function. Every concrete implementation (`coherent_dist`, `series_dist`, `parallel_dist`, `kofn_dist`, `bridge_dist`, or user-defined subclasses) should include `"dist_structure"`, the `algebraic.dist` ancestor `"univariate_dist"`, and `"dist"` in its class vector.

Usage

```
## S3 method for class 'dist_structure'
surv(x, ...)

## S3 method for class 'dist_structure'
cdf(x, ...)

## S3 method for class 'dist_structure'
sampler(x, ...)
```

Arguments

x	A dist_structure object.
...	Ignored.

Details

Concrete implementations provide S3 methods for the generics in this package. The minimum required methods are `ncomponents()`, `component()`, and one of `phi()` or `min_paths()`; every other generic has a default method on `dist_structure` that composes the primitives.

If both `phi.<class>()` and `min_paths.<class>()` are provided, the implementor is responsible for keeping them consistent: `phi` derives the in-package generics `reliability`, `critical_states`, and `is_coherent`; `min_paths` derives `min_cuts` and `system_signature`. Inconsistent implementations produce silently inconsistent results.

Value

This help topic documents the virtual base class together with the three distribution-algebra default methods that compose component distributions through the topology:

- `surv.dist_structure()` returns a closure `function(t, ...)` that evaluates the system survival function via the reliability identity `S_sys(t) = R(S_1(t), ..., S_m(t))`.
- `cdf.dist_structure()` returns a closure `function(t, ...)` equal to `1 - surv(x)(t)`.
- `sampler.dist_structure()` returns a closure `function(n, ...)` that draws `n` independent system lifetimes by sampling each component and combining via `system_lifetime()`.

Concrete subclasses override any of these for closed-form speed; see the closed-form specializations under the See Also entries.

See Also

`validate_dist_structure()` for an implementor-side construction-time validator. `phi()` and `min_paths()` for the bidirectional protocol primitives. The closed-form families (`exp_series()`, `wei_kofn()`, etc.) for reference implementations.

system_censoring.dist_structure

Per-component censoring from system observation

Description

Per-component censoring from system observation

Usage

```
## S3 method for class 'dist_structure'
system_censoring(x, times)

system_censoring(x, times)
```

Arguments

x	A dist_structure object.
times	Non-negative numeric vector of length ncomponents(x).

Value

A list with system_time (scalar) and component_status (character vector of length m, values in "exact", "left", "right").

system_lifetime.dist_structure

System lifetime from component times

Description

System lifetime from component times

Usage

```
## S3 method for class 'dist_structure'
system_lifetime(x, times)

system_lifetime(x, times)
```

Arguments

x	A dist_structure object.
times	Non-negative numeric vector of length ncomponents(x).

Value

Scalar system lifetime.

`system_signature.dist_structure` *System signature*

Description

Samaniego's signature: `s = (s_1, ..., s_m)` where `s_k` is the probability the system fails at the k-th component failure under i.i.d. absolutely-continuous component lifetimes. Only depends on the structure (phi), not on the component distribution. A default method on `dist_structure` enumerates the `m!` orderings; this is feasible for `m` up to about 8 or 9. Specialized subclasses should override with closed-form expressions.

Usage

```
## S3 method for class 'dist_structure'
system_signature(x)

system_signature(x)
```

Arguments

`x`	A dist_structure object.

Value

Numeric vector of length `ncomponents(x)` summing to 1.

`validate_dist_structure` *Validate that an object satisfies the dist_structure protocol*

Description

Checks that `x` declares `"dist_structure"` in its class chain and provides the three required generics: `ncomponents()`, `component()`, and at least one of `phi()` or `min_paths()`.

Usage

```
validate_dist_structure(x)
```

Arguments

`x`	An object claiming to satisfy the `dist_structure` protocol.

Details

Useful in subclass constructors to fail fast with a clear error message rather than at first method dispatch (where the user sees opaque "no applicable method" errors). A typical pattern:

```
my_subclass <- function(...) {
  obj <- structure(list(...), class = c("my_subclass", "dist_structure",
                                        "univariate_dist", "continuous_dist", "dist"))
  validate_dist_structure(obj)
  obj
}
```

Value

TRUE (invisibly) if all checks pass. Stops with an informative error otherwise.

Examples

```
# Success: a valid dist_structure (any built-in topology shortcut works).
validate_dist_structure(series_dist(replicate(3,
  algebraic.dist::exponential(1), simplify = FALSE)))

# Failure: an object that declares dist_structure but provides no
# primitives. Wrapped in tryCatch for the example's success status.
bad <- structure(list(),
                 class = c("not_a_real_class", "dist_structure",
                           "univariate_dist", "continuous_dist", "dist"))
tryCatch(validate_dist_structure(bad),
         error = function(e) conditionMessage(e))
```

---

vesely_fussell_importance

Vesely-Fussell importance at time t

Description

Probability that at least one minimal cut set containing `j` has all its components failed, given the system has failed by time `t`. Computed exactly via inclusion-exclusion over subsets of cuts that contain `j`.

Usage

```
vesely_fussell_importance(x, j, t)

## S3 method for class 'dist_structure'
vesely_fussell_importance(x, j, t)
```

Arguments

x	A dist_structure object.
j	Component index.
t	Scalar time.

Value

Numeric scalar in [0, 1].

---

wei_homogeneous_series *Series of Weibull components with common shape (closed form as a Weibull)*

Description

Constructs a dist_structure for a series of Weibull components sharing a common shape parameter. By the standard identity, the system lifetime is itself Weibull with the common shape and an aggregate scale (sum(1 / scale^shape))^(-1 / shape). Methods for surv, cdf, sampler, and mean forward to this aggregate Weibull, giving exact closed-form values.

Usage

```
wei_homogeneous_series(shape, scales)

## S3 method for class 'wei_homogeneous_series'
surv(x, ...)

## S3 method for class 'wei_homogeneous_series'
sampler(x, ...)

## S3 method for class 'wei_homogeneous_series'
mean(x, ...)
```

Arguments

shape	Positive scalar: common Weibull shape.
scales	Positive numeric vector: per-component Weibull scales.
x	A wei_homogeneous_series object.
...	Ignored.

Value

`wei_homogeneous_series()` returns an object of class `c("wei_homogeneous_series", "wei_series", "series_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist")`.

The associated S3 methods return:

- `surv()`: a closure `function(t, ...)`.
- `cdf()` is derived via the `dist_structure` default and returns a closure `function(t, ...)` equal to `1 - surv(x)(t)`.
- `sampler()`: a closure `function(n, ...)` returning `n` random variates from the system lifetime distribution.
- `mean()`: a numeric scalar (the mean system lifetime, `aggregate_scale * gamma(1 + 1 / shape)`).

Examples

```
sys <- wei_homogeneous_series(shape = 2, scales = c(1, 2, 3))
# System lifetime is Weibull(shape = 2, scale = aggregate_scale)
algebraic.dist::surv(sys)(1)
```

`wei_kofn` *k-out-of-n system of independent Weibull components (closed form)*

Description

Constructs a `dist_structure` for a k-out-of-m system whose components are independent (possibly heterogeneous) Weibulls. Closed-form `surv`, `cdf`, `sampler`, `density`, and `hazard` via subset enumeration, the critical-state density formula, and component order statistics.

Usage

```
wei_kofn(k, shapes, scales)

## S3 method for class 'wei_kofn'
surv(x, ...)

## S3 method for class 'wei_kofn'
sampler(x, ...)

## S3 method for class 'wei_kofn'
hazard(x, ...)

## S3 method for class 'wei_kofn'
density(x, ...)
```

Arguments

k	Minimum functioning components for system operation.
shapes	Positive numeric vector of length m.
scales	Positive numeric vector of length m.
x	A wei_kofn object.
...	Ignored.

Value

wei_kofn() returns an object of class c("wei_kofn", "kofn_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist").

The associated S3 methods return:

- surv(), density(), hazard(): a closure function(t, ...).
- cdf() is derived via the dist_structure default and returns a closure function(t, ...) equal to 1 - surv(x)(t).
- sampler(): a closure function(n, ...) returning n random variates from the system lifetime distribution.

Examples

```
sys <- wei_kofn(k = 2, shapes = c(1, 2, 3), scales = c(1, 2, 3))
algebraic.dist::surv(sys)(1)
```

---

wei_series *Series of heterogeneous Weibull components (closed form)*

Description

Constructs a dist_structure representing a series system whose components are independent Weibull distributions with possibly different shapes and scales. Closed-form methods are provided for surv, cdf, sampler, and algebraic.dist::hazard.

Usage

```
wei_series(shapes, scales)

## S3 method for class 'wei_series'
surv(x, ...)

## S3 method for class 'wei_series'
sampler(x, ...)

## S3 method for class 'wei_series'
hazard(x, ...)
```

Arguments

shapes	Positive numeric vector of length m: Weibull shape parameters per component.
scales	Positive numeric vector of length m (same length as shapes): Weibull scale parameters per component.
x	A wei_series object.
...	Ignored.

Value

wei_series() returns an object of class c("wei_series", "series_dist", "coherent_dist", "dist_structure", "univariate_dist", "continuous_dist", "dist").

The associated S3 methods return:

- surv(), hazard(): a closure function(t, ...).
- cdf() is derived via the dist_structure default and returns a closure function(t, ...) equal to 1 - surv(x)(t).
- sampler(): a closure function(n, ...) returning n random variates from the system lifetime distribution.

Examples

```
sys <- wei_series(shapes = c(1, 2, 3), scales = c(1, 2, 3))
algebraic.dist::surv(sys)(1)
```

Index

as_dist_structure
(as_dist_structure.dist_structure), 3
as_dist_structure.dist_structure, 3

birnbaum_importance, 3
bridge_dist, 4

cat(), *14*
cdf.cold_standby_dist
(cold_standby_dist), 5
cdf.dist_structure
(surv.dist_structure), 26
cdf.dist_structure(), *27*
coherent_dist, 5
cold_standby_dist, 5
component, 7
component(), *6*, *27*, *29*
compose_systems
(compose_systems.dist_structure), 7
compose_systems.dist_structure, 7
consecutive_k_dist, 8
critical_states
(critical_states.dist_structure), 9
critical_states.dist_structure, 9
criticality_importance, 9

density.exp_kofn (exp_kofn), 11
density.exp_series (exp_series), 13
density.wei_kofn (wei_kofn), 32
dist_structure, *3*, *4*, *7–10*, *14*, *16*, *19*, *21*, *23–29*, *31*
dist_structure (surv.dist_structure), 26
dual (dual.coherent_dist), 10
dual.coherent_dist, 10

exp_kofn, 11
exp_parallel, 12
exp_series, 13
exp_series(), *27*

format(), *24*
format.dist_structure, 14

gamma_series, 14

hazard.exp_kofn (exp_kofn), 11
hazard.exp_series (exp_series), 13
hazard.wei_kofn (wei_kofn), 32
hazard.wei_series (wei_series), 33

is_coherent
(is_coherent.dist_structure), 15
is_coherent.dist_structure, 15
is_dist_structure, 16

kofn_dist, 16

lognormal_series, 17

max_iid, 18
mean.cold_standby_dist, 19
mean.exp_parallel (exp_parallel), 12
mean.exp_series (exp_series), 13
mean.wei_homogeneous_series
(wei_homogeneous_series), 31
min_cuts (min_cuts.dist_structure), 19
min_cuts(), *21*
min_cuts.dist_structure, 19
min_iid, 20
min_paths (min_paths.dist_structure), 20
min_paths(), *23*, *27*, *29*
min_paths.dist_structure, 20
min_paths.dist_structure(), *23*

ncomponents, 21
ncomponents(), *6*, *27*, *29*

order_statistic, 21

order_statistic(), *17*

parallel_dist, 22
phi *(*phi.dist_structure*)*, 23
phi(), *20, 21*, *27*, *29*
phi.dist_structure, 23
phi.dist_structure(), *23*
print.dist_structure, 24

reliability
*(*reliability.dist_structure*)*, 24
reliability.dist_structure, 24

sampler.cold_standby_dist
*(*cold_standby_dist*)*, 5
sampler.dist_structure
*(*surv.dist_structure*)*, 26
sampler.dist_structure(), *27*
sampler.exp_kofn *(*exp_kofn*)*, 11
sampler.exp_parallel *(*exp_parallel*)*, 12
sampler.exp_series *(*exp_series*)*, 13
sampler.gamma_series *(*gamma_series*)*, 14
sampler.lognormal_series
*(*lognormal_series*)*, 17
sampler.wei_homogeneous_series
*(*wei_homogeneous_series*)*, 31
sampler.wei_kofn *(*wei_kofn*)*, 32
sampler.wei_series *(*wei_series*)*, 33
series_dist, *3*, 25
structural_importance
*(*structural_importance.dist_structure*)*, 25
structural_importance.dist_structure, 25
substitute_component
*(*substitute_component.dist_structure*)*, 26
substitute_component.dist_structure, 26
surv.cold_standby_dist
*(*cold_standby_dist*)*, 5
surv.dist_structure, 26
surv.dist_structure(), *27*
surv.exp_kofn *(*exp_kofn*)*, 11
surv.exp_parallel *(*exp_parallel*)*, 12
surv.exp_series *(*exp_series*)*, 13
surv.gamma_series *(*gamma_series*)*, 14
surv.lognormal_series
*(*lognormal_series*)*, 17
surv.wei_homogeneous_series
*(*wei_homogeneous_series*)*, 31
surv.wei_kofn *(*wei_kofn*)*, 32
surv.wei_series *(*wei_series*)*, 33
system_censoring
*(*system_censoring.dist_structure*)*, 28
system_censoring.dist_structure, 28
system_lifetime
*(*system_lifetime.dist_structure*)*, 28
system_lifetime(), *26, 27*
system_lifetime.dist_structure, 28
system_signature
*(*system_signature.dist_structure*)*, 29
system_signature.dist_structure, 29

validate_dist_structure, 29
validate_dist_structure(), *23*, *27*
vesely_fussell_importance, 30

wei_homogeneous_series, 31
wei_kofn, 32
wei_kofn(), *27*
wei_series, 33