Package 'ReliaGrowR'

April 16, 2026

Title Reliability Growth Analysis and Repairable Systems Modeling

Version 0.6

Description Modeling and plotting functions for Reliability Growth Analysis (RGA) and Non-Homogeneous Poisson Process (NHPP) models for repairable systems. RGA models include the Duane (1962) <doi:10.1109/TA.1964.4319640>, NHPP by Crow (1975) (No. AMSAATR138), Piecewise Weibull NHPP by Guo et al. (2010) <doi:10.1109/RAMS.2010.5448029>, and Piecewise Weibull NHPP with Change Point Detection based on the 'segmented' package by Muggeo (2024) <https://cran.r-project.org/package=segmented>. Repairable systems functions include the Mean Cumulative Function (MCF) using the Nelson-Aalen estimator, parametric Power Law and Log-Linear NHPP models, and forecasting.

Imports graphics, grDevices, plumber, segmented, stats

License CC BY 4.0

Encoding UTF-8

Suggests ellmer, knitr, pkgload, rmarkdown, spelling, testthat (>= 3.0.0), vdiffr, WeibullR

Language en-US

URL https://paulgovan.github.io/ReliaGrowR/, https://github.com/paulgovan/ReliaGrowR

Config/testthat/edition 3

VignetteBuilder knitr

BugReports https://github.com/paulgovan/ReliaGrowR/issues

RoxygenNote 7.3.3

Depends R (>= 3.5)

LazyData true

NeedsCompilation no

Author Paul Govan [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-1821-8492>)

Maintainer Paul Govan <paul.govan2@gmail.com>

Repository CRAN

Date/Publication 2026-04-16 07:22:17 UTC

Contents

---

duane *Duane Analysis*

Description

This function performs a Duane analysis (1962) doi:10.1109/TA.1964.4319640 on failure data by fitting a log-log linear regression of cumulative Mean Time Between Failures (MTBF) versus cumulative time. The function accepts either two numeric vectors (`times`, `failures`) or a data frame containing both.

Usage

```
duane(times, failures = NULL, conf.level = 0.95)
```

Arguments

`times`	Either: • A numeric vector of exact failure times, or • A data frame containing two columns: `times` and `failures`. The `times` column contains exact failure times, and the `failures` column contains the number of failures at each corresponding time.
`failures`	A numeric vector of the number of failures at each corresponding time in `times`. Ignored if `times` is a data frame. Must be the same length as `times` if both are vectors. All values must be positive and finite.
`conf.level`	Confidence level for the confidence bounds (default: `0.95`). Must be between 0 and 1 (exclusive).

Details

The scaling relationship between the size of input data (numbers of observations) and speed of algorithm execution is approximately linear (O(n)). The function is efficient and can handle large data sets (e.g., thousands of observations) quickly. The function uses base R functions and does not require any additional packages. The function includes comprehensive input validation and error handling to ensure robustness. The function is tested with a standard data set from a published paper and includes unit tests to verify correctness and performance.

Value

A list of class `"duane"` containing:

`times`	The input exact failure times.
`failures`	The input number of failures.
`n_obs`	The number of observations (failures).
`MTBF`	The cumulative mean time between failures.
`model`	The fitted `lm` (linear model) object containing the regression results.
`logLik`	The log-likelihood of the fitted model.
`AIC`	Akaike Information Criterion (AIC).
`BIC`	Bayesian Information Criterion (BIC).
`conf.level`	The confidence level.
`Cumulative_Time`	The cumulative operating times.
`Cumulative_MTBF`	The cumulative mean time between failures.
`Fitted_Values`	The fitted values on the MTBF scale.
`Confidence_Bounds`	Matrix of fitted values and confidence bounds on the MTBF scale.
`Residuals_Log`	Residuals on the log(MTBF) scale (from the regression).
`Residuals_MTBF`	Residuals on the MTBF scale (observed - fitted).

See Also

Other Duane functions: `plot.duane()`, `print.duane()`

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
fit1 <- duane(times, failures, conf.level = 0.90)
print(fit1)

df <- data.frame(times = times, failures = failures)
fit2 <- duane(df, conf.level = 0.95)
print(fit2)
```

`exposure` *Exposure Analysis for Repairable Systems.*

Description

Computes exposure (total operating time at risk) across one or more repairable systems as a function of time. Exposure is defined as the total accumulated observation time summed across all systems still under observation. The function also computes the number of systems at risk and the event rate (events per unit exposure) at each event time.

Usage

```
exposure(id = NULL, time = NULL, event = NULL, data = NULL)
```

Arguments

`id`	A vector of system/unit identifiers. Each unique value represents a distinct system.
`time`	A numeric vector of event or censoring times. Must be positive and finite.
`event`	An optional numeric vector of event indicators: 1 for an event, 0 for censoring (end of observation). If `NULL` (default), all observations are treated as events, and the maximum time per system is treated as the end of observation.
`data`	An optional data frame containing columns named `id`, `time`, and optionally `event`.

Details

Exposure is the total amount of operating time during which events can occur. For a fleet of k systems observed up to times $T_1, T_2, \ldots, T_k$, the total exposure is $E = \sum_{i=1}^{k} T_i$.

The **cumulative exposure** at time t is $E(t) = \sum_{i=1}^{k} \min(t, T_i)$, i.e., each system contributes time up to the lesser of t or its observation end.

The **event rate** at time t is the cumulative number of events divided by the cumulative exposure: $r(t) = N(t)/E(t)$.

Value

An object of class `exposure` containing:

`time`	Sorted unique event times (excluding censoring-only times).
`n_at_risk`	Number of systems under observation at each event time.
`cum_exposure`	Cumulative total exposure (system-time) up to each event time.
`cum_events`	Cumulative number of events up to each event time.
`event_rate`	Cumulative event rate (cum_events / cum_exposure) at each event time.
`total_exposure`	Total exposure across all systems and the full observation period.
`total_events`	Total number of events.
`n_systems`	Number of distinct systems.
`end_times`	Named numeric vector of end-of-observation times per system. Can be passed directly to `mcf(end_time = ...)` to ensure the MCF properly accounts for system exposure.

See Also

Other Repairable Systems Analysis: `mcf()`, `nhpp()`, `overlay_nhpp()`, `plot.exposure()`, `plot.mcf()`, `plot.nhpp()`, `plot.nhpp_predict()`, `predict_nhpp()`, `print.exposure()`, `print.mcf()`, `print.nhpp()`, `print.nhpp_predict()`

Examples

```
id   <- c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3)
time <- c(100, 350, 500, 80, 300, 600, 150, 250, 400, 700)
result <- exposure(id, time)
print(result)
plot(result)

# With censoring
id    <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
time  <- c(100, 350, 500, 80, 300, 400, 150, 250, 700)
event <- c(  1,   1,   0,  1,   1,   0,   1,   1,   1)
result2 <- exposure(id, time, event)
print(result2)
```

`grwr_api` *ReliaGrowR API*

Description

This function provides an interface to the ReliaGrowR API.#' This function provides an interface to the ReliaGrowR API.

Usage

```
grwr_api()
```

Value

Launches the ReliaGrowR API on a local server.

Examples

```
## Not run:
grwr_api()

## End(Not run)
```

`mcf` *Mean Cumulative Function for Repairable Systems.*

Description

Computes the non-parametric Mean Cumulative Function (MCF) for recurrent event data from one or more repairable systems, using the Nelson-Aalen estimator. The MCF estimates the expected cumulative number of events per system as a function of time, properly accounting for system exposure (observation windows).

Usage

```
mcf(
  id = NULL,
  time = NULL,
  event = NULL,
  end_time = NULL,
  data = NULL,
  conf_level = 0.95
)
```

Arguments

`id`	A vector of system/unit identifiers. Each unique value represents a distinct system. Ignored if `data` is provided.
`time`	A numeric vector of event or censoring times. Must be positive and finite. Ignored if `data` is provided.
`event`	An optional numeric vector of event indicators: 1 for an event, 0 for censoring (end of observation). If `NULL` (default), all observations are treated as events.
`end_time`	An optional named numeric vector of end-of-observation times per system, where names correspond to system identifiers. This defines the actual exposure window for each system. When provided, a system remains in the risk set until its `end_time`, even if its last event occurred earlier. If `NULL` (default), the end of observation is inferred as the maximum time recorded for each system (from both events and censoring records).
`data`	An optional data frame containing columns named `id`, `time`, and optionally `event` and `end_time`.
`conf_level`	Confidence level for bounds (default 0.95).

Details

The MCF at time t is estimated as:

$$\hat{M}(t) = \sum_{t_j \le t} \frac{d_j}{n_j}$$

where d_j is the number of events at time t_j and n_j is the number of systems still under observation at t_j. Variance is estimated as $\hat{V}(t) = \sum_{t_j \le t} d_j / n_j^2$.

The risk set n_j is determined by each system's **exposure window**. A system is considered at risk at time t_j if its end-of-observation time (from `end_time`, censoring records, or last event) is $\geq t_j$. Specifying `end_time` is important when systems were observed beyond their last event – without it, the MCF may overestimate the true recurrence rate because systems with no late events are assumed to have left observation at their last event time.

Value

An object of class `mcf` containing:

`time`	Unique event times.
`mcf`	MCF values at each event time.
`variance`	Variance estimates at each event time.
`lower_bounds`	Lower confidence bounds.
`upper_bounds`	Upper confidence bounds.
`conf_level`	Confidence level used.
`n_systems`	Number of distinct systems.
`n_events`	Total number of events.
`end_times`	Named vector of end-of-observation times per system.

See Also

Other Repairable Systems Analysis: `exposure()`, `nhpp()`, `overlay_nhpp()`, `plot.exposure()`, `plot.mcf()`, `plot.nhpp()`, `plot.nhpp_predict()`, `predict_nhpp()`, `print.exposure()`, `print.mcf()`, `print.nhpp()`, `print.nhpp_predict()`

Examples

```
# Basic usage (end of observation inferred from last event)
id <- c(1, 1, 1, 2, 2, 3, 3, 3, 3)
time <- c(100, 300, 500, 150, 400, 50, 200, 350, 600)
result <- mcf(id, time)
print(result)
plot(result, main = "Mean Cumulative Function")

# With explicit end-of-observation times (exposure-adjusted)
end_time <- c("1" = 800, "2" = 800, "3" = 800)
result2 <- mcf(id, time, end_time = end_time)
print(result2)
```

```
df <- data.frame(id = id, time = time)
result3 <- mcf(data = df)
print(result3)
```

---

nhpp *Non-Homogeneous Poisson Process Model for Repairable Systems.*

Description

Fits a parametric NHPP model to recurrent event data from repairable systems. Supported models include the Power Law process and the Log-Linear process. The Power Law model can also be fit as a piecewise (segmented) model with automatic change point detection or user-specified breakpoints.

Usage

```
nhpp(
  time,
  event = NULL,
  data = NULL,
  model_type = "Power Law",
  breaks = NULL,
  method = c("MLE", "LS"),
  conf_level = 0.95
)
```

Arguments

`time`	A numeric vector of cumulative event times, or a data frame containing columns `time` and optionally `event`. All values must be positive, finite, and strictly increasing.
`event`	An optional numeric vector of event counts at each time. If `NULL` (default), each time is treated as a single event.
`data`	An optional data frame containing columns `time` and optionally `event`.
`model_type`	Model type: `"Power Law"` (default) or `"Log-Linear"`.
`breaks`	Optional vector of breakpoints for piecewise Power Law model.
`method`	Estimation method: `"MLE"` (default) or `"LS"`. `"LS"` is not supported for `"Log-Linear"` models.
`conf_level`	Confidence level for bounds (default 0.95).

Details

The **Power Law NHPP** models the cumulative number of events as $N(t) = \lambda t^{\beta}$. The parameter $\beta > 1$ indicates a deteriorating system (increasing event rate), $\beta < 1$ an improving system, and $\beta = 1$ a constant rate (HPP).

The **Log-Linear NHPP** models the intensity as $\lambda(t) = \exp(a + bt)$ with cumulative function $\Lambda(t) = \frac{e^a}{b}(e^{bt} - 1)$.

Value

An object of class nhpp containing:

time	The input cumulative event times.
event	The event counts.
cum_events	Cumulative event counts.
n_obs	Number of observations.
model	Fitted model object (lm or segmented), or NULL for MLE.
model_type	"Power Law" or "Log-Linear".
method	"MLE" or "LS".
params	Named list of estimated parameters.
params_se	Named list of standard errors.
vcov	Variance-covariance matrix (MLE only).
fitted_values	Fitted cumulative events.
lower_bounds	Lower confidence bounds.
upper_bounds	Upper confidence bounds.
residuals	Model residuals.
logLik	Log-likelihood.
AIC	Akaike Information Criterion.
BIC	Bayesian Information Criterion.
breakpoints	Breakpoints (log scale) if piecewise model.
conf_level	Confidence level used.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), overlay_nhpp(), plot.exposure(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.exposure(), print.mcf(), print.nhpp(), print.nhpp_predict()

Examples

```
time <- c(200, 400, 600, 800, 1000)
event <- c(3, 5, 4, 7, 6)
result <- nhpp(time, event)
print(result)
plot(result, main = "Power Law NHPP")

result_ll <- nhpp(time, event, model_type = "Log-Linear")
print(result_ll)
```

overlay_nhpp *Overlay Plot for Multiple NHPP Models*

Description

Plots multiple fitted nhpp objects on a single set of axes, using distinct colors per model. Observed data points, fitted lines, and optional confidence bounds are drawn for every model. Models may have been fit to different datasets.

Usage

```
overlay_nhpp(
  models,
  conf_bounds = TRUE,
  legend = TRUE,
  legend_pos = "topleft",
  colors = NULL,
  ...
)
```

Arguments

models	A named or unnamed list of objects of class nhpp. At least one model must be provided. If the list is named, those names are used as legend labels; otherwise labels default to "Model 1", "Model 2", etc.
conf_bounds	Logical; draw confidence bounds for each model (default: TRUE).
legend	Logical; draw a legend (default: TRUE).
legend_pos	Legend position keyword (default: "topleft").
colors	Optional character vector of colors, one per model. If NULL (default), palette() colors are cycled.
...	Additional arguments passed to the initial plot() call (e.g., main, xlab, ylab). Not forwarded to subsequent lines() or points() calls.

Value

Invisibly returns NULL.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), plot.exposure(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.exposure(), print.mcf(), print.nhpp(), print.nhpp_predict()

Examples

```
t1 <- c(200, 400, 600, 800, 1000)
e1 <- c(3, 5, 4, 7, 6)
t2 <- c(300, 600, 900, 1200, 1500)
e2 <- c(4, 6, 5, 8, 7)
m1 <- nhpp(t1, e1)
m2 <- nhpp(t2, e2)
overlay_nhpp(list(System_A = m1, System_B = m2),
  main = "NHPP Overlay", xlab = "Time",
  ylab = "Cumulative Events"
)
```

---

overlay_rga *Overlay Plot for Multiple RGA Models*

Description

Plots multiple fitted `rga` objects on a single set of axes, using distinct colors per model. Observed data points, fitted lines, and optional confidence bounds are drawn for every model. Models may have been fit to different datasets.

Usage

```
overlay_rga(
  models,
  conf_bounds = TRUE,
  legend = TRUE,
  legend_pos = "bottomright",
  colors = NULL,
  log = FALSE,
  ...
)
```

Arguments

`models`	A named or unnamed list of objects of class `rga`. At least one model must be provided. If the list is named, those names are used as legend labels; otherwise labels default to `"Model 1"`, `"Model 2"`, etc.
`conf_bounds`	Logical; draw confidence bounds for each model (default: `TRUE`).
`legend`	Logical; draw a legend (default: `TRUE`).
`legend_pos`	Legend position keyword (default: `"bottomright"`).
`colors`	Optional character vector of colors, one per model. If `NULL` (default), `palette()` colors are cycled.
`log`	Logical; use log-log axes (default: `FALSE`).
`...`	Additional arguments passed to the initial `plot()` call (e.g., `main`, `xlab`, `ylab`). Not forwarded to subsequent `lines()` or `points()` calls.

Value

Invisibly returns NULL.

See Also

Other Reliability Growth Analysis: plot.rga(), plot.rga_predict(), predict_rga(), print.rga(), print.rga_predict(), rga()

Examples

```
t1 <- c(100, 200, 300, 400, 500)
f1 <- c(1, 2, 1, 3, 2)
t2 <- c(150, 300, 450, 600, 750)
f2 <- c(2, 1, 3, 2, 4)
m1 <- rga(t1, f1)
m2 <- rga(t2, f2)
overlay_rga(list(System_A = m1, System_B = m2),
  main = "RGA Overlay", xlab = "Cumulative Time",
  ylab = "Cumulative Failures"
)
```

---

plot.duane *Plot Method for Duane Analysis*

Description

Generates a Duane plot (log-log or linear scale) with fitted regression line and optional confidence bounds.

Usage

```
## S3 method for class 'duane'
plot(
  x,
  log = TRUE,
  conf.int = TRUE,
  legend = TRUE,
  legend.pos = "topleft",
  ...
)
```

Arguments

x	An object of class "duane".
log	Logical; whether to use logarithmic scales for axes (default: TRUE).
conf.int	Logical; whether to plot confidence bounds (default: TRUE).
legend	Logical; whether to include a legend (default: TRUE).
legend.pos	Position of the legend (default: "topleft").
...	Further arguments passed to plot().

Value

Invisibly returns NULL.

See Also

Other Duane functions: duane(), print.duane()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
fit <- duane(times, failures)
plot(fit, main = "Duane Plot", xlab = "Cumulative Time", ylab = "Cumulative MTBF")
```

---

plot.exposure *Plot Method for exposure Objects.*

Description

Produces a multi-panel plot of exposure analysis results. The default layout shows cumulative exposure and cumulative events versus time (top panel), the number of systems at risk over time (middle panel), and the event rate over time (bottom panel). Alternatively, a single which panel can be selected.

Usage

```
## S3 method for class 'exposure'
plot(
  x,
  which = c("all", "exposure", "at_risk", "event_rate"),
  legend = TRUE,
  legend_pos = "topleft",
  ...
)
```

Arguments

x	An object of class exposure.
which	Character string selecting which panel(s) to plot. One of "all" (default), "exposure", "at_risk", or "event_rate".
legend	Logical; show the legend (default: TRUE).
legend_pos	Position of the legend (default: "topleft").
...	Additional arguments passed to the underlying plot().

Value

Invisibly returns NULL.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), overlay_nhpp(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.exposure(), print.mcf(), print.nhpp(), print.nhpp_predict()

Examples

```
id   <- c(1, 1, 1, 2, 2, 2, 3, 3, 3, 3)
time <- c(100, 350, 500, 80, 300, 600, 150, 250, 400, 700)
result <- exposure(id, time)
plot(result)
plot(result, which = "exposure")
```

---

plot.mcf *Plot Method for mcf Objects.*

Description

Plots the Mean Cumulative Function with optional confidence bounds.

Usage

```
## S3 method for class 'mcf'
plot(x, conf_bounds = TRUE, legend = TRUE, legend_pos = "topleft", ...)
```

Arguments

x	An object of class mcf.
conf_bounds	Logical; include confidence bounds (default: TRUE).
legend	Logical; show the legend (default: TRUE).
legend_pos	Position of the legend (default: "topleft").
...	Additional arguments passed to plot().

Value

Invisibly returns NULL.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), overlay_nhpp(), plot.exposure(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.exposure(), print.mcf(), print.nhpp(), print.nhpp_predict()

Examples

```
id <- c(1, 1, 1, 2, 2, 3, 3, 3, 3)
time <- c(100, 300, 500, 150, 400, 50, 200, 350, 600)
result <- mcf(id, time)
plot(result, main = "Mean Cumulative Function")
```

`plot.nhpp` *Plot Method for nhpp Objects.*

Description

Plots observed cumulative events with the fitted NHPP model and optional confidence bounds.

Usage

```
## S3 method for class 'nhpp'
plot(x, conf_bounds = TRUE, legend = TRUE, legend_pos = "topleft", ...)
```

Arguments

`x`	An object of class nhpp.
`conf_bounds`	Logical; include confidence bounds (default: TRUE).
`legend`	Logical; show the legend (default: TRUE).
`legend_pos`	Position of the legend (default: "topleft").
`...`	Additional arguments passed to `plot()`.

Value

Invisibly returns `NULL`.

See Also

Other Repairable Systems Analysis: `exposure()`, `mcf()`, `nhpp()`, `overlay_nhpp()`, `plot.exposure()`, `plot.mcf()`, `plot.nhpp_predict()`, `predict_nhpp()`, `print.exposure()`, `print.mcf()`, `print.nhpp()`, `print.nhpp_predict()`

Examples

```
time <- c(200, 400, 600, 800, 1000)
event <- c(3, 5, 4, 7, 6)
result <- nhpp(time, event)
plot(result, main = "Power Law NHPP", xlab = "Time", ylab = "Cumulative Events")
```

`plot.nhpp_predict` *Plot Method for nhpp_predict Objects.*

Description

Plots observed data, fitted model, and forecast with optional confidence bounds.

Usage

```
## S3 method for class 'nhpp_predict'
plot(x, conf_bounds = TRUE, legend = TRUE, legend_pos = "topleft", ...)
```

Arguments

`x`	An object of class `nhpp_predict`.
`conf_bounds`	Logical; include confidence bounds (default: TRUE).
`legend`	Logical; show the legend (default: TRUE).
`legend_pos`	Position of the legend (default: "topleft").
`...`	Additional arguments passed to `plot()`.

Value

Invisibly returns `NULL`.

See Also

Other Repairable Systems Analysis: `exposure()`, `mcf()`, `nhpp()`, `overlay_nhpp()`, `plot.exposure()`, `plot.mcf()`, `plot.nhpp()`, `predict_nhpp()`, `print.exposure()`, `print.mcf()`, `print.nhpp()`, `print.nhpp_predict()`

Examples

```
time <- c(200, 400, 600, 800, 1000)
event <- c(3, 5, 4, 7, 6)
fit <- nhpp(time, event)
fc <- predict_nhpp(fit, time = c(1500, 2000))
plot(fc, main = "NHPP Forecast", xlab = "Time", ylab = "Cumulative Events")
```

plot.rga *Plot Method for RGA Objects*

Description

This function generates plots for objects of class rga.

Usage

```
## S3 method for class 'rga'
plot(
  x,
  conf_bounds = TRUE,
  legend = TRUE,
  log = FALSE,
  legend_pos = "bottomright",
  ...
)
```

Arguments

x	An object of class rga, which contains the results from the RGA model.
conf_bounds	Logical; include confidence bounds (default: TRUE).
legend	Logical; show the legend (default: TRUE).
log	Logical; use a log-log scale (default: FALSE).
legend_pos	Position of the legend (default: "bottomright").
...	Additional arguments passed to plot().

Value

Invisibly returns NULL.

See Also

Other Reliability Growth Analysis: overlay_rga(), plot.rga_predict(), predict_rga(), print.rga(), print.rga_predict(), rga()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
result <- rga(times, failures)
plot(result,
  main = "Reliability Growth Analysis",
  xlab = "Cumulative Time", ylab = "Cumulative Failures"
)
```

`plot.rga_predict` *Plot Method for rga_predict Objects*

Description

Plots observed data, the fitted reliability growth curve, and the forecast with optional confidence bounds for an `rga_predict` object.

Usage

```
## S3 method for class 'rga_predict'
plot(x, conf_bounds = TRUE, legend = TRUE, legend_pos = "bottomright", ...)
```

Arguments

`x`	An object of class `rga_predict`.
`conf_bounds`	Logical; include confidence bounds (default: `TRUE`).
`legend`	Logical; show the legend (default: `TRUE`).
`legend_pos`	Position of the legend (default: `"bottomright"`).
`...`	Additional arguments passed to `plot()`.

Value

Invisibly returns `NULL`.

See Also

Other Reliability Growth Analysis: `overlay_rga()`, `plot.rga()`, `predict_rga()`, `print.rga()`, `print.rga_predict()`, `rga()`

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
fit <- rga(times, failures)
fc <- predict_rga(fit, times = c(1500, 2000))
plot(fc, main = "RGA Forecast", xlab = "Cumulative Time", ylab = "Cumulative Failures")
```

ppplot.rga *P-P Plot for RGA Objects*

Description

This function creates a P-P plot for a fitted Reliability Growth Analysis (RGA) model. Currently only supports the Crow-AMSAA model. A P-P plot compares the empirical cumulative distribution function (CDF) to the theoretical CDF specified by the model. If the model fits well, the points should fall approximately along a straight line.

Usage

```
ppplot.rga(x, main = "P-P Plot", ...)
```

Arguments

x	An object of class rga.
main	Title of the plot.
...	Additional arguments passed to plot().

Value

A P-P plot comparing empirical and theoretical CDFs.

See Also

Other goodness-of-fit: qqplot.rga()

Examples

```
times <- c(5, 10, 15, 20, 25)
failures <- c(1, 2, 1, 3, 2)
fit <- rga(times, failures)
ppplot.rga(fit)
```

predict_nhpp *Forecast Cumulative Events from an NHPP Model.*

Description

Takes a fitted nhpp object and a vector of future cumulative times, returning predicted cumulative events with confidence bounds.

Usage

```
predict_nhpp(object, time, conf_level = 0.95)
```

Arguments

object	An object of class nhpp returned by nhpp().
time	A numeric vector of cumulative times at which to forecast. All values must be finite and > 0.
conf_level	Confidence level (default 0.95).

Value

An object of class nhpp_predict containing:

time	Forecast times.
cum_events	Predicted cumulative events.
lower_bounds	Lower confidence bounds.
upper_bounds	Upper confidence bounds.
conf_level	Confidence level used.
model_type	Model type.
nhpp_object	The original nhpp object.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), overlay_nhpp(), plot.exposure(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), print.exposure(), print.mcf(), print.nhpp(), print.nhpp_predict()

Examples

```
time <- c(200, 400, 600, 800, 1000)
event <- c(3, 5, 4, 7, 6)
fit <- nhpp(time, event)
fc <- predict_nhpp(fit, time = c(1500, 2000))
print(fc)
plot(fc, main = "NHPP Forecast", xlab = "Time", ylab = "Cumulative Events")
```

---

predict_rga *Forecast Cumulative Failures from a Reliability Growth Model*

Description

Takes a fitted rga object and a vector of cumulative times, returning predicted cumulative failures with confidence bounds as an rga_predict S3 object.

Usage

```
predict_rga(object, times, conf_level = 0.95)
```

Arguments

object	An object of class rga returned by rga().
times	A numeric vector of cumulative times at which to forecast. All values must be finite and > 0. A warning is issued if any value is at or below the maximum observed cumulative time (hindcasting).
conf_level	The desired confidence level (default 0.95). Must be a single finite numeric in (0, 1).

Value

An object of class rga_predict containing:

times	The forecast cumulative times.
cum_failures	Predicted cumulative failures.
lower_bounds	Lower confidence bounds.
upper_bounds	Upper confidence bounds.
conf_level	The confidence level used.
model_type	Either "Crow-AMSAA" or "Piecewise NHPP".
rga_object	The original rga object (used by the plot method).

See Also

Other Reliability Growth Analysis: overlay_rga(), plot.rga(), plot.rga_predict(), print.rga(), print.rga_predict(), rga()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
fit <- rga(times, failures)
fc <- predict_rga(fit, times = c(1500, 2000))
print(fc)
```

---

print.duane *Print method for duane objects.*

Description

This function prints a summary of the Duane analysis result.

Usage

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

Arguments

x	An object of class "duane" returned by the duane_plot function.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Duane functions: duane(), plot.duane()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
fit <- duane(times, failures)
print(fit)
```

---

print.exposure *Print Method for exposure Objects.*

Description

Prints a summary of the exposure analysis results.

Usage

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

Arguments

x	An object of class exposure.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), overlay_nhpp(), plot.exposure(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.mcf(), print.nhpp(), print.nhpp_predict()

Examples

```
id   <- c(1, 1, 2, 2)
time <- c(100, 200, 150, 300)
result <- exposure(id, time)
print(result)
```

`print.mcf` *Print Method for mcf Objects.*

Description

Prints a summary of the Mean Cumulative Function results.

Usage

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

Arguments

`x`	An object of class `mcf`.
`...`	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Repairable Systems Analysis: `exposure()`, `mcf()`, `nhpp()`, `overlay_nhpp()`, `plot.exposure()`, `plot.mcf()`, `plot.nhpp()`, `plot.nhpp_predict()`, `predict_nhpp()`, `print.exposure()`, `print.nhpp()`, `print.nhpp_predict()`

Examples

```
id <- c(1, 1, 1, 2, 2, 3, 3, 3, 3)
time <- c(100, 300, 500, 150, 400, 50, 200, 350, 600)
result <- mcf(id, time)
print(result)
```

print.nhpp *Print Method for nhpp Objects.*

Description

Prints a summary of the NHPP model results.

Usage

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

Arguments

x	An object of class nhpp.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), overlay_nhpp(), plot.exposure(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.exposure(), print.mcf(), print.nhpp_predict()

Examples

```
time <- c(200, 400, 600, 800, 1000)
event <- c(3, 5, 4, 7, 6)
result <- nhpp(time, event)
print(result)
```

print.nhpp_predict *Print Method for nhpp_predict Objects.*

Description

Prints a formatted table of forecast cumulative events with confidence bounds.

Usage

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

Arguments

x	An object of class nhpp_predict.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Repairable Systems Analysis: exposure(), mcf(), nhpp(), overlay_nhpp(), plot.exposure(), plot.mcf(), plot.nhpp(), plot.nhpp_predict(), predict_nhpp(), print.exposure(), print.mcf(), print.nhpp()

Examples

```
time <- c(200, 400, 600, 800, 1000)
event <- c(3, 5, 4, 7, 6)
fit <- nhpp(time, event)
fc <- predict_nhpp(fit, time = c(1500, 2000))
print(fc)
```

---

print.rdt *Print method for rdt objects*

Description

This function provides a formatted print method for objects of class rdt.

Usage

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

Arguments

x	An object of class rdt.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

Examples

```
plan <- rdt(target = 0.9, mission_time = 1000, conf_level = 0.9, beta = 1, n = 10)
print(plan)
```

print.rga *Print method for rga objects.*

Description

This function prints a summary of the results from an object of class rga.

Usage

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

Arguments

x	An object of class rga, which contains the results from the RGA model.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Reliability Growth Analysis: overlay_rga(), plot.rga(), plot.rga_predict(), predict_rga(), print.rga_predict(), rga()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
result <- rga(times, failures)
print(result)
```

print.rga_predict *Print Method for rga_predict Objects*

Description

Prints a formatted table of forecast cumulative failures with confidence bounds for an rga_predict object.

Usage

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

Arguments

x	An object of class rga_predict.
...	Additional arguments (not used).

Value

Invisibly returns the input object.

See Also

Other Reliability Growth Analysis: overlay_rga(), plot.rga(), plot.rga_predict(), predict_rga(), print.rga(), rga()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
fit <- rga(times, failures)
fc <- predict_rga(fit, times = c(1500, 2000))
print(fc)
```

---

qqplot.rga *Q-Q Plot for RGA Objects*

Description

This function creates a Q-Q plot for a fitted Reliability Growth Analysis (RGA) model Currently only supports the Crow-AMSAA model. A Q-Q plot compares the quantiles of the empirical data to the quantiles of the theoretical distribution specified by the model. If the model fits well, the points should fall approximately along a straight line.

Usage

```
qqplot.rga(x, main = "Q-Q Plot", ...)
```

Arguments

x	An object of class rga.
main	Title of the plot.
...	Additional arguments passed to stats::qqplot().

Value

A Q-Q plot comparing empirical and theoretical quantiles.

See Also

Other goodness-of-fit: ppplot.rga()

Examples

```
times <- c(5, 10, 15, 20, 25)
failures <- c(1, 2, 1, 3, 2)
fit <- rga(times, failures)
qqplot.rga(fit)
```

---

rdt *Reliability Demonstration Test (RDT) Plan Calculator*

Description

This function calculates the required test time or sample size for a Reliability Demonstration Test (RDT) based on specified reliability, mission time, confidence level, and Weibull shape parameter.

Usage

```
rdt(
  target,
  mission_time,
  conf_level,
  beta = 1,
  f = 0,
  n = NULL,
  test_time = NULL
)
```

Arguments

target	Required reliability at mission time (0 < target < 1).
mission_time	Mission duration (time units). Must be greater than 0.
conf_level	Desired confidence level (e.g., 0.9 for 90% confidence). The confidence level must be between 0 and 1 (exclusive).
beta	Weibull shape parameter (beta=1 corresponds to exponential distribution). Must be greater than 0. Default is 1.
f	Number of allowable failures during the test (non-negative integer). Default is 0 (zero-failure test plan). Increasing f reduces the required test time or sample size at the cost of accepting more observed failures.
n	Sample size (optional, supply if solving for test_time). Must be a positive integer.
test_time	Test time per unit (optional, supply if solving for n). Must be greater than 0.

Value

The function returns an object of class `rdt` that contains:

`Distribution` Type of distribution used (Exponential or Weibull).

`Beta` Weibull shape parameter.

`Allowed_Failures` Number of allowable failures during the test.

`Target_Reliability` Specified target reliability.

`Mission_Time` Specified mission time.

`Required_Test_Time` Calculated required test time (if n is provided).

`Input_Sample_Size` Provided sample size (if test_time is calculated).

`Required_Sample_Size` Calculated required sample size (if test_time is provided).

`Input_Test_Time` Provided test time (if n is calculated).

Examples

```
#' # Example 1: Calculate required test time
plan1 <- rdt(target = 0.9, mission_time = 1000, conf_level = 0.9, beta = 1, n = 10)
print(plan1)
# Example 2: Calculate required sample size
plan2 <- rdt(target = 0.9, mission_time = 1000, conf_level = 0.9, beta = 1, test_time = 2000)
print(plan2)
```

---

rga *Reliability Growth Analysis.*

Description

This function performs reliability growth analysis using the Crow-AMSAA model by Crow (1975) (AMSAATR138) or piecewise NHPP model by Guo et al. (2010) doi:10.1109/RAMS.2010.5448029. It fits a log-log linear regression of cumulative failures versus cumulative time. The function accepts either two numeric vectors (`times`, `failures`) or a data frame containing both. The `Piecewise NHPP` model can automatically detect change points or use user-specified breakpoints.

Usage

```
rga(
  times,
  failures,
  times_type = c("failure_times", "cumulative_failure_times"),
  model_type = "Crow-AMSAA",
```

```
  breaks = NULL,
  conf_level = 0.95,
  method = c("LS", "MLE")
)
```

Arguments

times	Either a numeric vector of failure-time inputs or a data frame containing both time inputs and failure counts. If times_type = "failure_times" (default), times is treated exactly as in previous versions of the function and is cumulatively summed inside rga(). If times_type = "cumulative_failure_times", times is treated as already cumulative and is used directly without applying cumsum(). If a data frame is provided, it must contain two columns: times and failures.
failures	A numeric vector of the number of failures at each corresponding time in times. Must be the same length as times if both are vectors. All values must be positive and finite. Ignored if times is a data frame.
times_type	Character scalar indicating how to interpret times. "failure_times" (default) preserves the current behavior and cumulatively sums times inside rga(). "cumulative_failure_times" treats times as already cumulative and skips that internal cumsum().
model_type	The model type. Either Crow-AMSAA (default) or Piecewise NHPP with change point detection.
breaks	An optional vector of breakpoints for the Piecewise NHPP model.
conf_level	The desired confidence level, which defaults to 95%. The confidence level is the probability that the confidence interval contains the true mean response.
method	Estimation method: "LS" (default) for least-squares log-log regression, or "MLE" for maximum likelihood estimation of the Crow-AMSAA model. "MLE" is not supported for model_type = "Piecewise NHPP".

Details

The scaling relationship between the size of input data (numbers of observations) and speed of algorithm execution is approximately linear (O(n)). The function is efficient and can handle large data sets (e.g., thousands of observations) quickly. The function uses the segmented package for piecewise regression, which employs an iterative algorithm to estimate breakpoints. The number of iterations required for convergence may vary depending on the data and initial values. In practice, the function typically converges within a few iterations for most data sets. However, in some cases, especially with complex data or poor initial values, it may take more iterations.

Value

The function returns an object of class rga that contains:

times	The input time vector, stored exactly as supplied.
cum_times	The cumulative time vector used for fitting.
times_type	How times was interpreted: "failure_times" or "cumulative_failure_times".

failures	The input number of failures.
n_obs	The number of observations (failures).
cum_failures	Cumulative failures.
model	The fitted model object (lm (linear model) or segmented).
residuals	Model residuals on the log-log scale. These represent deviations of the observed log cumulative failures from the fitted values and are useful for diagnostic checking.
logLik	The log-likelihood of the fitted model. The log-likelihood is a measure of model fit, with higher values indicating a better fit.
AIC	Akaike Information Criterion (AIC). AIC is a measure used for model selection, with lower values indicating a better fit.
BIC	Bayesian Information Criterion(BIC). BIC is another criterion for model selection
breakpoints	Breakpoints (log scale) if applicable.
fitted_values	Fitted cumulative failures on the original scale.
lower_bounds	Lower confidence bounds (original scale).
upper_bounds	Upper confidence bounds (original scale).
betas	Estimated beta(s). Betas are the slopes of the log-log plot.
betas_se	Standard error(s) of the estimated beta(s).
growth_rate	Estimated growth rate(s). Growth rates are calculated as 1 - beta.
lambdas	Estimated lambda(s). Lambdas are the intercepts of the log-log plot.

See Also

Other Reliability Growth Analysis: overlay_rga(), plot.rga(), plot.rga_predict(), predict_rga(), print.rga(), print.rga_predict()

Examples

```
times <- c(100, 200, 300, 400, 500)
failures <- c(1, 2, 1, 3, 2)
result1 <- rga(times, failures)
print(result1)

df <- data.frame(times = times, failures = failures)
result2 <- rga(df)
print(result2)

cum_times <- cumsum(times)
result2b <- rga(cum_times, failures, times_type = "cumulative_failure_times")
print(result2b)

result3 <- rga(times, failures, model_type = "Piecewise NHPP")
print(result3)

result4 <- rga(times, failures, model_type = "Piecewise NHPP", breaks = c(450))
print(result4)
```

`sim_failures` *Simulate Failures from a Conditional Weibull Model*

Description

Simulates which units in a non-failed population fail next by using a Weibull life model conditional on each unit's current runtime. When a positive `window` is supplied, the function calibrates a Weibull scale parameter (unless provided directly) so that the expected number of failures within the window matches `n`. Units are then sampled with probability proportional to their conditional Weibull failure probability over the window, and failure times are drawn from the truncated conditional Weibull distribution. The full fleet is returned: selected units are labelled `"Failure"` and the remaining units are labelled `"Suspension"`.

Usage

```
sim_failures(n, runtimes, replace = FALSE, window = NULL, beta = 1, eta = NULL)
```

Arguments

`n`	Positive integer. Number of failures to simulate.
`runtimes`	Numeric vector of positive values. The current operating runtime of each unit in the non-failed population.
`replace`	Logical scalar. If `TRUE`, sampling is done with replacement (a unit may be selected more than once). Default is `FALSE`.
`window`	`NULL` or a single positive numeric. The width of the observation window. When `NULL` (default), event times equal current runtimes. When provided, failure times are sampled from the conditional Weibull distribution over `(runtime, runtime + window]`, and suspension times are `runtime + window`.
`beta`	Positive numeric scalar. Weibull shape parameter used to model the age-dependent hazard. Defaults to `1`, corresponding to an exponential-process assumption.
`eta`	`NULL` or a single positive numeric. Weibull scale parameter. When `NULL` and `window` is supplied, the scale is calibrated so that the expected number of failures across the fleet during the window matches `n`.

Details

When `window = NULL`, the function returns the current fleet state at the supplied runtimes. In this case, failing units are selected using relative Weibull hazard weights implied by `beta`.

Value

A data frame with `length(runtimes)` rows sorted ascending by `runtime`, containing:

`index`	Integer index of the unit in `runtimes`.
`runtime`	Simulated event time.
`type`	Character; `"Failure"` for selected units, `"Suspension"` for the rest.

The returned object also carries attributes `weibull_beta` and `weibull_eta` describing the Weibull parameters used for the simulation.

See Also

Other data preparation: `weibull_to_rga()`

Examples

```
set.seed(42)
runtimes <- c(100, 500, 200, 800, 300)
result <- sim_failures(2, runtimes, beta = 1.5)
print(result)

# With an observation window
set.seed(42)
result_w <- sim_failures(2, runtimes, window = 50, beta = 1.5)
print(result_w)
```

`testdata` *Reliability Test Data*

Description

A dataset containing example reliability test data from the military report "Reliability Growth Prediction" (1986) by The Analytical Sciences Corporation. This dataset includes cumulative ETI, failure counts, cumulative MTBF, report numbers, flags, and causes for two different LRUs (G1 and G2).

Usage

```
testdata
```

Format

@format ## `testdata` A data frame with 25 rows and 6 variables:

LRU The Line Replaceable Unit identifier (G1 or G2).

Cum_ETI Cumulative Equivalent Test Hours (ETI).

Failure_Count Cumulative number of failures observed.

Cum_MTBF Cumulative Mean Time Between Failures (MTBF).

Report_No Report number associated with the failure.

Flag A flag indicating special conditions or notes.

Cause Cause of the failure (e.g., D for Design, M for Manufacturing, R for Random, NR for No Report).

@usage data(testdata)

Examples

```
data(testdata)
head(testdata)
summary(testdata)
str(testdata)
```

`weibull_to_rga` *Weibull to RGA*

Description

Converts Weibull data (failure, suspension, and interval-censored times) into a format suitable for reliability growth analysis (RGA). The function handles exact failure times, right-censored suspensions, and interval-censored data. It approximates interval-censored failures by placing them at the midpoint of the interval. The output is a data frame with cumulative time and failure counts. This format can be used with RGA models such as Crow-AMSAA.

Usage

```
weibull_to_rga(
  failures,
  suspensions = NULL,
  interval_starts = NULL,
  interval_ends = NULL
)
```

Arguments

`failures`	A numeric vector of exact failure times. Each failure time indicates when an item failed during the observation period.
`suspensions`	A numeric vector of suspension (right-censored) times. A suspension indicates that the item was removed from observation at that time without failure. This parameter is optional and can be NULL if there are no suspensions.
`interval_starts`	A numeric vector of interval start times (lower bound of censoring). This parameter is optional and can be NULL if there are no interval-censored data. If provided, it must be the same length as `interval_ends`.
`interval_ends`	A numeric vector of interval end times (upper bound of censoring). This parameter is optional and can be NULL if there are no interval-censored data. If provided, it must be the same length as `interval_starts`.

Value

The data frame contains two columns:

`CumulativeTime`	Cumulative time at each failure event.
`Failures`	Number of failures at each cumulative time point.

The function approximates interval-censored failures by placing them at the midpoint of the interval.

See Also

Other data preparation: `sim_failures()`

Examples

```
failures <- c(100, 200, 200, 400)
suspensions <- c(250, 350, 450)
interval_starts <- c(150, 300)
interval_ends <- c(180, 320)
result <- weibull_to_rga(failures, suspensions, interval_starts, interval_ends)
print(result)
```

Index