# Package 'DesignComplexSurvival'

November 23, 2026

**Type** Package

**Title** Simulation-Free Design of Complex Time-to-Event Trials

**Version** 0.4.0

**Description** Computes the expected numbers of events, the mean, variance and covariance of the log-rank statistic over calendar time, and the power, type I error, required number of events, sample size, accrual duration and follow-up of two-group time-to-event trials, without patient-level simulation. Each group follows a multistate model (progression-free, progressed, switched, dead, and progression-free after a switch before progression) with piecewise constant intensities, which covers progression-free survival, overall survival and treatment switching before, at or after progression, with an effect of the switch on the death hazard or an accelerated failure time effect, together with piecewise uniform accrual, piecewise exponential dropout, a censoring of progression-free survival only, switching that starts at a calendar time and group sequential designs with efficacy bounds and binding or non-binding futility bounds. Progression-free and overall survival can be tested together with a group sequential fixed-sequence or Bonferroni procedure, the latter with recycling of the level after a rejection as in Maurer and Bretz (2013) <doi:10.1080/19466315.2013.807748>, using the joint distribution of the two log-rank statistics over calendar time. For designs with one endpoint, the expected study duration, numbers of events and enrolled patients when the trial stops at its bounds, the conditional power at an interim analysis, and plots of the bounds, the probabilities of stopping and the survival functions are also given. The variance and covariance of the log-rank score under the alternative follow Luo et al. (2019) <doi:10.1002/sim.7975>.

**License** GPL (>= 2)

**URL** https://gosukehommaEX.github.io/DesignComplexSurvival/, https://github.com/gosukehommaEX/DesignComplexSurvival

**BugReports** https://github.com/gosukehommaEX/DesignComplexSurvival/issues

**Depends** R (>= 4.1.0)

**Imports** graphics, grDevices, mvtnorm (>= 1.2-0), stats

**Suggests** gsDesign, gsDesign2, knitr, lrstat, rmarkdown, rpact, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Gosuke Homma [aut, cre] (ORCID: <https://orcid.org/0000-0002-6854-5627>)

**Maintainer** Gosuke Homma <my.name.is.gosuke@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-11-23 14:40:02 UTC

# Contents

as.data.frame.ms_power . . . 3
gs_bounds . . . 4
gs_prob . . . 6
gs_prob_mtp . . . 7
gs_spend . . . 10
lr_cov . . . 11
lr_moments . . . 13
ms_accrual . . . 15
ms_cond_power . . . 16
ms_events . . . 18
ms_event_time . . . 19
ms_fleischer . . . 20
ms_looks . . . 21
ms_model . . . 22
ms_null . . . 25
ms_power . . . 26
ms_prob . . . 29
ms_scenario . . . 30
ms_size . . . 32
ms_surv . . . 34
ms_test . . . 35
plot.ms_power . . . 38
print.ms_accrual . . . 40
print.ms_cond_power . . . 40
print.ms_looks . . . 41
print.ms_model . . . 42
print.ms_power . . . 42
print.ms_scenario . . . 43

print.ms_size . . . . . 44
print.ms_test . . . . . 44
print.summary.ms_power . . . . . 45
summary.ms_power . . . . . 46

**Index** **47**

---

`as.data.frame.ms_power` *Per-look results of an ms_power object as a data frame*

---

## Description

Per-look results of an ms_power object as a data frame

## Usage

```
## S3 method for class 'ms_power'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `x` | An `ms_power` object. |
| `row.names, optional` | Not used. |
| `...` | Not used. |

## Value

The data frame `x$looks`.

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
res <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
                test = ms_test(), events = 300)
as.data.frame(res)
```

gs_bounds *Bounds of a group sequential design*

## Description

Computes one-sided efficacy bounds by alpha spending and, optionally, futility bounds that are fixed or obtained by beta spending, for a vector of standardized statistics with a multivariate normal distribution.

## Usage

```
gs_bounds(
  info_frac,
  alpha = 0.025,
  sf_eff = "OF",
  param_eff = NULL,
  futility = c("none", "fixed", "beta"),
  fut_bound = NULL,
  sf_fut = "OF",
  param_fut = NULL,
  beta = 0.1,
  binding = FALSE,
  mean = NULL,
  corr = NULL,
  sigma_alt = NULL
)
```

## Arguments

| | |
|---|---|
| info_frac | Information fractions used for spending (strictly increasing and ending at 1). |
| alpha | One-sided type I error. |
| sf_eff, param_eff | Alpha spending function and its parameter (see gs_spend). |
| futility | "none", "fixed" or "beta". |
| fut_bound | Fixed futility bounds on the Z scale at looks 1 to K - 1 (for futility = "fixed"). |
| sf_fut, param_fut | Beta spending function and its parameter. |
| beta | Total type II error for beta spending. |
| binding | Whether the futility bounds are binding. |
| mean | Mean vector of the statistics under the alternative (required for futility = "beta"). |
| corr | Correlation matrix of the statistics under the null hypothesis. Defaults to sqrt(t_j / t_k) for j <= k with t = info_frac. |
| sigma_alt | Covariance matrix of the statistics under the alternative. Defaults to corr. |

## Details

The efficacy bound at look k solves, under the null hypothesis (mean 0, correlation `corr`), the equation that the probability of continuing to look k and then crossing the bound equals the alpha spent at look k. With `binding = TRUE` the continuation region at the earlier looks includes the futility bounds; with `binding = FALSE` the futility bounds are ignored for the efficacy bounds.

With `futility = "beta"`, the futility bound at look k solves, under the alternative (mean `mean`, covariance `sigma_alt`), the equation that the probability of continuing to look k and then falling below the bound equals the beta spent at look k. The mean may take any value at each look, so that non-proportional hazards are allowed. A futility bound above the efficacy bound is set to the efficacy bound. At the last look the futility bound equals the efficacy bound. With `binding = TRUE`, a futility bound set to the efficacy bound at an interim look stops every trial at that look, so that no alpha can be spent at the later looks; the function then stops with an error that names that look.

The probabilities are computed as in `gs_prob`: with the Miwa algorithm for rectangles of at most three coordinates and for covariance matrices of Markov sequences, such as the default `corr`, except for rectangles of nine or more coordinates with two finite limits for some coordinate (nine or more looks with futility bounds at the earlier looks), and with the Genz-Bretz algorithm otherwise (for example for `sigma_alt` of `variance = "alternative"` or `"studentized"` with four or more looks).

## Value

A data frame with one row per look: `look`, `info_frac`, cumulative `alpha_spent` and `beta_spent`, the efficacy bound `upper`, the futility bound `lower` (`-Inf` when there is none) and the one-sided nominal p-value of the efficacy bound `nominal_p`.

## References

Lan KKG, DeMets DL (1983). Discrete sequential boundaries for clinical trials. Biometrika 70:659-663.

## See Also

`gs_spend`, `gs_prob`, `ms_power`

## Examples

```
# O'Brien-Fleming type efficacy bounds at 50, 75 and 100 percent
gs_bounds(c(0.5, 0.75, 1))

# Non-binding futility by beta spending with a delayed effect
gs_bounds(c(0.5, 0.75, 1), futility = "beta", sf_fut = "HSD",
          param_fut = -2, mean = c(0.8, 1.9, 2.8))
```

gs_prob *Stopping probabilities of a group sequential design*

## Description

For statistics with a multivariate normal distribution, computes at each look the probability of continuing to the look and then crossing the efficacy bound (`prob_upper`) or falling below the futility bound (`prob_lower`).

## Usage

```
gs_prob(lower, upper, mean = 0, sigma)
```

## Arguments

| | |
|---|---|
| `lower` | Futility bounds (`-Inf` for none). At the last look the bound is usually equal to the efficacy bound. |
| `upper` | Efficacy bounds (`Inf` for none). |
| `mean` | Mean vector (recycled if of length 1). |
| `sigma` | Covariance matrix. |

## Details

Rectangle probabilities are computed with `mvtnorm::pmvnorm()`. The Miwa algorithm, which is deterministic, is used for rectangles of at most three coordinates and when the covariance matrix is that of a Markov sequence (its inverse is tridiagonal), as for statistics with independent increments. For four or more coordinates with other covariance matrices, such as those of `variance = "alternative"` and `"studentized"` in `ms_test`, the Miwa algorithm was found to be inaccurate, and for nine or more coordinates with two finite limits for some coordinate (futility bounds at eight or more earlier looks), for which mvtnorm leaves out terms of the inclusion-exclusion of the Miwa algorithm, the Genz-Bretz algorithm is used, with a fixed seed, so that the results are reproducible and the random number generator of the session is not changed, and with an absolute error tolerance of 1e-5 for each rectangle, which can be changed with `options(DesignComplexSurvival.genz_abseps = ...)`.

## Value

A data frame with `look`, `prob_upper`, `prob_lower` and their cumulative sums `cum_upper` and `cum_lower`.

## See Also

`gs_bounds`

## Examples

```
b <- gs_bounds(c(0.5, 1))
corr <- matrix(c(1, sqrt(0.5), sqrt(0.5), 1), 2)
gs_prob(b$lower, b$upper, mean = 0, sigma = corr)
gs_prob(b$lower, b$upper, mean = c(2, 2.9), sigma = corr)
```

---

`gs_prob_mtp` *Rejection probabilities of a group sequential design with two endpoints*

---

## Description

For the standardized statistics of two endpoints with a joint multivariate normal distribution, computes under a multiple testing procedure the probability of rejecting each null hypothesis at each look, the probability of rejecting both and the probability of rejecting at least one.

## Usage

```
gs_prob_mtp(
  upper,
  lower = NULL,
  mean = 0,
  sigma,
  analysis = NULL,
  mtp = c("fixed_sequence", "bonferroni"),
  hierarchical = c("overall", "stagewise"),
  upper_recycled = NULL
)
```

## Arguments

| | |
|---|---|
| `upper` | List of two numeric vectors: the efficacy bounds of the first and of the second endpoint at their looks (`Inf` for no efficacy test at a look). |
| `lower` | Futility bounds of the first endpoint at its looks (`-Inf` for none), or `NULL` for no futility bounds. |
| `mean` | Mean vector of the statistics, first the looks of the first endpoint and then those of the second (recycled if of length 1). |
| `sigma` | Covariance matrix of the statistics in the same order, for example `sigma` of `lr_cov` with two endpoints. |
| `analysis` | List of two integer vectors: the analysis at which each look of the first and of the second endpoint takes place (strictly increasing within an endpoint; equal numbers mean the same analysis). `NULL` means that both endpoints are tested at analyses 1 to K. |
| `mtp` | `"fixed_sequence"` or `"bonferroni"`. |

`hierarchical` `"overall"` or `"stagewise"` (used with `mtp = "fixed_sequence"`).

`upper_recycled` For `mtp = "bonferroni"`, an optional list of two numeric vectors: the efficacy bounds of the first and of the second endpoint at their looks at the level after the other hypothesis has been rejected (not above `upper`). `NULL` means no recycling.

## Details

Each endpoint is tested at its own looks, and `analysis` places the looks of both endpoints on one sequence of analyses; an analysis may include looks of one endpoint or of both. A hypothesis is rejected at the first look at which its statistic is at or above its efficacy bound and is not tested again. The trial stops for futility at an interim look of the first endpoint at which its hypothesis has not been rejected and its statistic is below the futility bound. The futility bound at the last look of the first endpoint is not used.

With `mtp = "bonferroni"`, each endpoint is tested at all its looks until the trial stops, with bounds computed at levels that add up to alpha. The trial stops when both hypotheses are rejected or for futility.

With `mtp = "bonferroni"` and `upper_recycled`, the level of a rejected hypothesis is passed on to the other hypothesis (the graphical procedure of Maurer and Bretz 2013, Algorithm 1, for two hypotheses). Each hypothesis is tested with its bounds in `upper` until the other hypothesis is rejected, and with its bounds in `upper_recycled` from the analysis at which the other is rejected on: a hypothesis rejected with its bound in `upper` at an analysis makes the other be tested with its bound in `upper_recycled` at the same analysis. The bounds in `upper_recycled` are those of the level after recycling, for example `(w1 + g21 w2) alpha` for the first hypothesis with the weights w and the fraction g21 of the level of the second hypothesis that is passed on to the first. They must not exceed the bounds in `upper`, which holds when the nominal levels of the spending function do not decrease with the level (Maurer and Bretz 2013, Section 3.1). The futility stop of the first endpoint also applies after the second hypothesis has been rejected.

With `mtp = "fixed_sequence"`, the second hypothesis is tested only after the first has been rejected. With `hierarchical = "overall"` it is tested at the analysis at which the first hypothesis is rejected and at the later looks of the second endpoint, until it is rejected (strategy (b) of Glimm et al. 2010). With `hierarchical = "stagewise"` the trial stops at the analysis at which the first hypothesis is rejected, and the second hypothesis is tested only at that analysis (strategy (a) of Glimm et al. 2010, the procedure of Tamhane et al. 2010).

The probabilities are sums of rectangle probabilities of the multivariate normal distribution, one for each combination of the outcome of the first endpoint (rejection or stop for futility at a look, or no rejection) and the look at which the second hypothesis is rejected. With `upper_recycled`, the probabilities are summed over the outcomes at each analysis: at an analysis at which both hypotheses are tested and neither has been rejected, with the bounds A1, A2 in `upper` and B1, B2 in `upper_recycled`, both hypotheses are rejected when Z1 >= A1 and Z2 >= B2 or when B1 <= Z1 < A1 and Z2 >= A2, only the first when Z1 >= A1 and Z2 < B2, only the second when Z1 < B1 and Z2 >= A2, and neither when Z1 < A1 and Z2 < A2; at an analysis at which one hypothesis is tested, it is rejected or not. The futility stop splits the outcomes in which the first hypothesis is tested and not rejected at an interim look. The rectangle probabilities are computed with `mvtnorm::pmvnorm()` (at most 20 looks in total): with the Genz-Bretz algorithm for the rectangles that involve both endpoints and have four or more coordinates and, as in `gs_prob`, for the other rectangles of four or more coordinates whose covariance matrix is not that of a Markov sequence and of nine or more coordinates with two finite limits for some coordinate, for which the Miwa algorithm can

be inaccurate, and with the Miwa algorithm otherwise. The Genz-Bretz algorithm uses a fixed seed, so that the results are reproducible and the random number generator of the session is not changed, and an absolute error tolerance of 1e-5 for each rectangle, which can be changed with `options(DesignComplexSurvival.genz_abseps = ...)`.

## Value

A list with

`looks` A data frame with one row per look: `endpoint` (1 or 2), `look`, `analysis`, the probability of rejecting the hypothesis of the endpoint at the look (`prob_upper`), its cumulative sum (`cum_upper`) and the probability of stopping for futility at the look (`prob_lower`, 0 for the second endpoint and for the last look of the first).

`reject` The probabilities of rejecting the first hypothesis (`first`), the second (`second`), both (`both`) and at least one (`any`).

## References

Glimm E, Maurer W, Bretz F (2010). Hierarchical testing of multiple endpoints in group-sequential trials. Statistics in Medicine 29:219-228.

Maurer W, Bretz F (2013). Multiple testing in group sequential trials using graphical approaches. Statistics in Biopharmaceutical Research 5:311-320.

Tamhane AC, Mehta CR, Liu L (2010). Testing a primary and a secondary endpoint in a group sequential design. Biometrics 66:1174-1184.

## See Also

`gs_prob`, `gs_bounds`, `lr_cov`

## Examples

```
# Two looks at information fractions 0.5 and 1 for both endpoints and
# correlation 0.5 between the endpoints
tt <- c(0.5, 1)
r <- sqrt(outer(tt, tt, pmin) / outer(tt, tt, pmax))
sigma <- rbind(cbind(r, 0.5 * r), cbind(0.5 * r, r))
b1 <- gs_bounds(tt, sf_eff = "OF")$upper
b2 <- gs_bounds(tt, sf_eff = "Pocock")$upper
gs_prob_mtp(list(b1, b2), mean = c(2, 2.9, 1.5, 2.1), sigma = sigma)$reject
# Probability of rejecting the second hypothesis when it is true
gs_prob_mtp(list(b1, b2), mean = c(2, 2.9, 0, 0), sigma = sigma)$reject
# Bonferroni with the level alpha / 2 for each endpoint and the level
# alpha after the other hypothesis has been rejected (full recycling)
h1 <- gs_bounds(tt, alpha = 0.0125, sf_eff = "OF")$upper
h2 <- gs_bounds(tt, alpha = 0.0125, sf_eff = "Pocock")$upper
gs_prob_mtp(list(h1, h2), mean = c(2, 2.9, 1.5, 2.1), sigma = sigma,
            mtp = "bonferroni", upper_recycled = list(b1, b2))$reject
```

gs_spend *Error spending functions*

## Description

Cumulative error spent at information fractions `t` for a total (one-sided) error `alpha`.

## Usage

```
gs_spend(
  t,
  alpha = 0.025,
  type = c("OF", "Pocock", "HSD", "power", "user"),
  param = NULL
)
```

## Arguments

| | |
|---|---|
| `t` | Information fractions between 0 and 1. |
| `alpha` | Total error to spend. |
| `type` | Type of spending function: `"OF"`, `"Pocock"`, `"HSD"`, `"power"` or `"user"`. |
| `param` | Parameter of the spending function (required for `"HSD"`, `"power"` and `"user"`). |

## Details

The available types are

- `"OF"`: Lan-DeMets spending function of O'Brien-Fleming type, `2 - 2 pnorm(qnorm(1 - alpha / 2) / sqrt(t))`.
- `"Pocock"`: Lan-DeMets spending function of Pocock type, `alpha log(1 + (exp(1) - 1) t)`.
- `"HSD"`: Hwang-Shih-DeCani family with parameter `param` (gamma), `alpha (1 - exp(-gamma t)) / (1 - exp(-gamma))`, and `alpha t` for gamma = 0.
- `"power"`: Kim-DeMets power family with parameter `param` (rho), `alpha t^rho`.
- `"user"`: cumulative proportions of `alpha` given in `param`, one value per element of `t`.

The same functions are used for beta spending, with `alpha` replaced by the total type II error.

## Value

Numeric vector of cumulative error spent.

## References

Lan KKG, DeMets DL (1983). Discrete sequential boundaries for clinical trials. Biometrika 70:659-663.

## See Also

gs_bounds

## Examples

```
gs_spend(c(0.5, 0.75, 1), alpha = 0.025, type = "OF")
gs_spend(c(0.5, 0.75, 1), alpha = 0.1, type = "HSD", param = -2)
```

---

# lr_cov *Joint distribution of the log-rank statistic at several calendar times*

---

## Description

Returns the mean vector and covariance matrix of the standardized log-rank statistics Z(t_1), ..., Z(t_K) of one endpoint, or of PFS and OS together.

## Usage

```
lr_cov(
  scenario,
  n,
  calendar_time,
  endpoint = "os",
  mean = c("direct", "schoenfeld"),
  variance = c("null", "alternative", "studentized"),
  trigger = NULL
)
```

## Arguments

| | |
|---|---|
| scenario | An ms_scenario object. |
| n | Total sample size. |
| calendar_time | Strictly increasing calendar times (positive). |
| endpoint | "os", "pfs" or both (for example c("pfs", "os")). |
| mean | "direct" or "schoenfeld". |
| variance | "null", "alternative" or "studentized". |
| trigger | Endpoint whose observed number of events determines each calendar time ("os" or "pfs"; one value or one per time), or NULL or NA for fixed calendar times. Used only with variance = "studentized", and with switching from a calendar time. |

## Details

With `variance = "null"` the statistics have unit variances and correlations `sqrt(V(t_j) / V(t_k))` for t_j <= t_k, from the independent increments of the score (Tsiatis 1981). With `variance = "alternative"` the covariance of the score under the alternative is that of Luo et al. (2019), computed here from the per-patient influence functions of the score, and is divided by `sqrt(V(t_j) V(t_k))`.

With `variance = "studentized"` the covariance is the asymptotic covariance of Z = -U / sqrt(V-hat) itself, where V-hat is the usual variance estimator. The per-patient influence function of Z adds to that of the score a term for the variability of V-hat and, for times determined by the observed number of events of an endpoint (`trigger`), a term for the variability of the analysis time. Both terms vanish under the null hypothesis.

For PFS and OS together (`endpoint = c("pfs", "os")`) the statistics are ordered by endpoint and then by time. With `"studentized"` all covariances come from the influence functions. With `"null"` the statistics of one endpoint have the covariances described above and the correlations between the endpoints are those of the studentized statistics. With `"alternative"` the covariances between the endpoints are the covariances of the scores divided by `sqrt(V_P V_O)`. Under the null hypothesis the three choices agree, and the covariance between the endpoints is that of Danzer et al. (2025, Theorem 1).

With switching from a calendar time (`switch_start` of `ms_scenario`), the hazard of OS depends on the entry time and the score has no independent increments, even under the null hypothesis. The means, information and covariances of OS then integrate over the entry time, with the proportion in the risk set and the pooled hazard of each calendar time. With `"null"` the statistics have unit variances and the correlations of the studentized statistics (which, under the null hypothesis, are the exact correlations), with `"alternative"` the covariances of the scores divided by `sqrt(V_j V_k)`, and with `"studentized"` the covariances of the studentized statistics; `trigger` is used with all three. Only OS alone with `mean = "direct"` is covered; PFS is not affected.

## Value

A list with `time`, `endpoint`, `events`, `info`, `mean` (mean vector of Z) and `sigma` (covariance matrix of Z), with one element (row) per endpoint and time.

## References

Tsiatis AA (1981). The asymptotic joint distribution of the efficient scores test for the proportional hazards model calculated over time. Biometrika 68:311-315.

Luo X, Mao X, Chen X, Qiu J, Bai S, Quan H (2019). Design and monitoring of survival trials in complex scenarios. Statistics in Medicine 38:192-209.

Danzer MF, Rufibach K, Beyersmann J, Schmidt R (2025). Exhausting the type I error level in event-driven group-sequential designs with a closed testing procedure for progression-free and overall survival. arXiv:2512.08658.

## See Also

`lr_moments`, `gs_prob`

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
lr_cov(scn, n = 400, calendar_time = c(24, 36))
fl <- ms_scenario(ms_fleischer(3, 9), ms_fleischer(4.3, 12.9),
                  accrual = ms_accrual(rate = 20))
lr_cov(fl, n = 400, calendar_time = 30, endpoint = c("pfs", "os"),
       variance = "studentized")
```

---

# lr_moments *Mean and variance of the log-rank statistic over calendar time*

---

## Description

Computes, at each calendar time, the expected events, the mean of the log-rank score U (observed minus expected events in the treatment group), its information V (the limit of the usual variance estimator) and the mean and variance of the standardized statistic Z = -U / sqrt(V), which is positive when the treatment is better.

## Usage

```
lr_moments(
  scenario,
  n,
  calendar_time,
  endpoint = c("os", "pfs"),
  mean = c("direct", "schoenfeld"),
  variance = c("null", "alternative", "studentized"),
  trigger = NULL
)
```

## Arguments

| | |
|---|---|
| scenario | An ms_scenario object. |
| n | Total sample size. |
| calendar_time | Calendar times (positive). |
| endpoint | "os" or "pfs". |
| mean | "direct" or "schoenfeld". |
| variance | "null", "alternative" or "studentized". |
| trigger | Endpoint whose observed number of events determines each calendar time ("os" or "pfs"; one value or one per time), or NULL or NA for fixed calendar times. Used only with variance = "studentized", and with switching from a calendar time. |

## Details

Let y0 and y1 be the probabilities of being at risk at follow-up time s in each group, multiplied by the allocation probabilities and by the enrolled fraction at t - s. Then E(U) = n times the integral of y0 y1 / (y0 + y1) (lambda1 - lambda0) and V = n times the integral of y0 y1 / (y0 + y1)^2 (y0 lambda0 + y1 lambda1), from 0 to t.

The mean of Z is

- `mean = "direct"`: -E(U) / sqrt(V), the mean of Yung and Liu (2020, Section 2.3.3);
- `mean = "schoenfeld"`: -sqrt(p0 p1 D) times the event-weighted average of the log hazard ratio, where D is the expected number of events (Schoenfeld 1981).

The variance of Z is 1 for `variance = "null"`. For `variance = "alternative"` it is the asymptotic variance of the score under the alternative divided by V, from Luo et al. (2019), which accounts for the variability of the numbers at risk. This variance assumes simple randomization; Yung and Liu (2020) show that block randomization gives a variance that is not larger. For `variance = "studentized"` it is the asymptotic variance of Z itself, which also accounts for the variability of V and, for times determined by the observed number of events of an endpoint (`trigger`), of the analysis time (see `lr_cov`).

With switching from a calendar time (`switch_start` of `ms_scenario`), the moments of OS integrate over the entry time; see `lr_cov`.

## Value

A data frame with `time`, `events_control`, `events_treatment`, `events`, `score_mean`, `info`, `z_mean` and `z_var`.

## References

Luo X, Mao X, Chen X, Qiu J, Bai S, Quan H (2019). Design and monitoring of survival trials in complex scenarios. Statistics in Medicine 38:192-209.

Schoenfeld D (1981). The asymptotic properties of nonparametric tests for comparing survival distributions. Biometrika 68:316-319.

Yung G, Liu Y (2020). Sample size and power for the weighted log-rank test and Kaplan-Meier based tests with allowance for nonproportional hazards. Biometrics 76:939-950.

## See Also

`lr_cov`, `ms_events`

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
lr_moments(scn, n = 400, calendar_time = c(24, 36))
```

`ms_accrual` *Piecewise uniform accrual*

## Description

Defines the accrual pattern either by accrual rates or by the proportions of patients enrolled in intervals of a given accrual duration.

## Usage

```
ms_accrual(time = 0, rate = NULL, prop = NULL, duration = NULL)
```

## Arguments

| | |
|---|---|
| `time` | Interval start times (calendar time from the start of the study). Must start at 0 and be strictly increasing. |
| `rate` | Accrual rates in each interval (patients per unit time). |
| `prop` | Proportions of patients enrolled in each interval (summing to 1). |
| `duration` | Accrual duration (with `prop`). |

## Details

With `rate`, patients are enrolled at `rate[j]` patients per unit time from `time[j]` on, and the accrual duration follows from the sample size. The last rate applies until the sample size is reached.

With `prop` and `duration`, a proportion `prop[j]` of the patients is enrolled uniformly in the interval starting at `time[j]`, and the last interval ends at `duration`. The accrual rates then scale with the sample size. When `ms_size` solves for the accrual duration, all interval limits are scaled by the same factor.

## Value

An object of class `ms_accrual`.

## See Also

`ms_scenario`

## Examples

```
# 10 patients per month in the first 3 months, then 30 per month
ms_accrual(time = c(0, 3), rate = c(10, 30))

# 30 percent of the patients in the first 6 months of an 18-month accrual
ms_accrual(time = c(0, 6), prop = c(0.3, 0.7), duration = 18)
```

`ms_cond_power` *Conditional power at an interim look*

## Description

Computes, for a design with one endpoint, the probability that the trial crosses an efficacy bound at a later look, given the standardized log-rank statistic observed at an interim look, under a chosen distribution of the statistics.

## Usage

```
ms_cond_power(
  x,
  look,
  z,
  under = c("design", "null", "trend", "hr", "scenario"),
  hr = NULL,
  scenario = NULL
)
```

## Arguments

| | |
|---|---|
| `x` | An `ms_power` object with one endpoint (from `ms_power` or `ms_size`). |
| `look` | Interim look at which `z` is observed (from 1 to the number of looks minus 1). |
| `z` | Observed standardized log-rank statistic at `look` (positive when the treatment is better), or the observed statistics at the looks 1 to `look`. It must be below the efficacy bounds and, with binding futility bounds, not below the futility bounds. |
| `under` | Distribution of the statistics: `"design"`, `"null"`, `"trend"`, `"hr"` or `"scenario"` (see Details). |
| `hr` | Hazard ratio (treatment versus control) for `under = "hr"`. |
| `scenario` | An `ms_scenario` object with the allocation and accrual of the design scenario, for `under = "scenario"`. |

## Details

At the interim look j with the observed statistic z, the statistics of the later looks given $Z_j = z$ have the conditional normal distribution of the joint normal distribution of the statistics chosen by `under`. The conditional power is the probability that, at some later look k, the statistic crosses the efficacy bound after the statistics of the looks between j and k have stayed between the futility and efficacy bounds, as in `gsCP()` of gsDesign; `cond_power_nonbinding` leaves out the futility bounds of the later interim looks (a trial that continues despite non-binding futility bounds). The bounds are those of the design (`x$looks`). When the events observed at the interim look differ from those planned, the bounds of the design no longer apply: compute the design again with `ms_power` with looks that give the observed events and the remaining planned events, and use the new design.

With `z` of length `look`, the statistics of the looks 1 to `look` are all conditioned on. For statistics with independent increments (`variance = "null"` of `ms_test`, except OS with switching from a

calendar time) the result is the same as with the statistic of the look alone; with `"alternative"` and `"studentized"` it can differ. In both cases the conditioning does not use the fact that the trial did not stop at the earlier looks.

The distributions (`under`):

`"design"` The mean and covariance of the statistics under the design scenario (`x$mean`, `x$sigma`).

`"null"` Mean zero and the correlations under the null scenario (`x$sigma_null`), at the looks of the type I error (`x$looks_null`); the result is the conditional type I error.

`"trend"` The current trend: a constant hazard ratio `exp(-z / sqrt(p0 p1 d_j))` estimated from z, where p0 and p1 are the allocation probabilities and d the expected events of the tested endpoint at the looks of the design, so that the mean of the statistic at look k is `z sqrt(d_k / d_j)`, with the correlations `sqrt(d_j / d_k)` used for the bounds.

`"hr"` A constant hazard ratio `hr`: the mean at look k is `-log(hr) sqrt(p0 p1 d_k)`, with the correlations `sqrt(d_j / d_k)`.

`"scenario"` The mean and covariance under `scenario`, with the approximations of the test of the design; the event-driven looks take place when the expected events of their trigger endpoint reach the event targets of the design (`x$targets`), also when the final look of the design is fixed by `followup`, and the calendar-driven looks keep their times.

`"trend"` and `"hr"` assume proportional hazards (the Schoenfeld approximation) and do not use the shape of the hazard ratio of the design scenario, such as a delayed effect; `"scenario"` does.

## Value

An object of class `ms_cond_power`, a list with

**cond_power** The conditional power.

**cond_power_nonbinding** The conditional power without the futility bounds of the later interim looks.

**looks** A data frame with one row per later look: `look`, `time` and `events` (calendar time and expected events of the tested endpoint under the distribution of `under`), the bounds `z_upper` and `z_lower`, the conditional mean and standard deviation of the statistic (`cond_mean`, `cond_sd`), the conditional probabilities of stopping for efficacy and for futility at the look (`prob_upper`, `prob_lower`) and the cumulative conditional power `cum_upper`.

**look, z, under** The inputs.

**hr** The hazard ratio of `"trend"` and `"hr"` (`NA` otherwise).

## References

Ortega-Villa AM, Grieco MC, Rubenstein K, Wang J, Proschan MA (2025). Futility monitoring in clinical trials. Statistics in Medicine, 44, e70157.

## See Also

`ms_power`, `gs_prob`

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
des <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
                test = ms_test(), events = 300)
ms_cond_power(des, look = 1, z = 1.2)
ms_cond_power(des, look = 1, z = 1.2, under = "trend")
ms_cond_power(des, look = 1, z = 1.2, under = "hr", hr = 0.8)
```

---

# ms_events *Expected numbers of patients, events and dropouts over calendar time*

---

## Description

Expected numbers of patients, events and dropouts over calendar time

## Usage

```
ms_events(scenario, n, calendar_time, endpoint = c("os", "pfs"))
```

## Arguments

| | |
|---|---|
| scenario | An ms_scenario object. |
| n | Total sample size. |
| calendar_time | Calendar times from the start of the study. |
| endpoint | "os" or "pfs". |

## Details

For group g with allocation probability p_g, the expected number of events by calendar time t is n p_g times the integral from 0 to t of A(t - s) f_g(s) G_g(s), where A is the enrolled fraction, f_g the density of the endpoint and G_g the probability of no dropout. The expected number of dropouts is obtained in the same way with the dropout density. For PFS, the dropouts include the censoring of PFS only (dropout_pfs of ms_model) and, with pfs_switch = "censor" in ms_scenario, the censoring at a switch before progression.

## Value

A data frame with time, the numbers enrolled (enrolled, enrolled_control, enrolled_treatment), the expected events (events_control, events_treatment, events) and the expected dropouts (dropouts_control, dropouts_treatment).

## See Also

ms_event_time, lr_moments

## Examples

```
ctl <- ms_model(death = log(2) / 12, dropout = 0.005)
trt <- ms_model(death = 0.7 * log(2) / 12, dropout = 0.005)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
ms_events(scn, n = 400, calendar_time = c(12, 24, 36))
```

---

# ms_event_time *Calendar time at which a number of events is expected*

---

## Description

Solves for the calendar time at which the expected total number of events of an endpoint reaches each target.

## Usage

```
ms_event_time(scenario, n, events, endpoint = c("os", "pfs"))
```

## Arguments

| | |
|---|---|
| scenario | An ms_scenario object. |
| n | Total sample size. |
| events | Target numbers of events (positive). |
| endpoint | "os" or "pfs". |

## Value

Numeric vector of calendar times. An error is raised when a target cannot be reached.

## See Also

ms_events

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
ms_event_time(scn, n = 400, events = c(150, 300))
```

`ms_fleischer` *Exponential model for PFS and OS of Fleischer et al. (2009)*

## Description

Creates the `ms_model` in which the time to progression and the time to death are exponential and the hazard of death does not change at progression. Then OS is exponential with median `median_os` and PFS is exponential with median `median_pfs`.

## Usage

```
ms_fleischer(median_pfs, median_os, dropout = 0)
```

## Arguments

| | |
|---|---|
| `median_pfs` | Median PFS. |
| `median_os` | Median OS (not smaller than `median_pfs`). |
| `dropout` | Dropout hazard. |

## Details

With death hazard `lambda_d = log(2) / median_os` and PFS hazard `log(2) / median_pfs`, the progression hazard is `log(2) / median_pfs - lambda_d`. This is the model of Section 2 of Fleischer et al. (2009) with the hazard of death before and after progression equal to `lambda_d`.

## Value

An object of class `ms_model`.

## References

Fleischer F, Gaschler-Markefski B, Bluhmki E (2009). A statistical model for the dependence between progression-free survival and overall survival. Statistics in Medicine 28:2669-2686.

## See Also

`ms_model`

## Examples

```
m <- ms_fleischer(median_pfs = 6, median_os = 15)
ms_surv(m, time = c(6, 15), endpoint = "pfs")
ms_surv(m, time = c(6, 15), endpoint = "os")
```

`ms_looks` *Timing of the analyses*

## Description

Defines when the interim and final analyses take place: when the expected number of events of an endpoint reaches given fractions of its final number of events, or at given calendar times, and which endpoints are tested at each analysis.

## Usage

```
ms_looks(
  trigger = c("os", "pfs", "calendar"),
  fraction = NULL,
  time = NULL,
  test = NULL
)
```

## Arguments

| | |
|---|---|
| `trigger` | `"os"` or `"pfs"` for event-driven looks, or `"calendar"`; one value for all looks or one value per look. |
| `fraction` | Event fractions of the event-driven looks: with one trigger, strictly increasing and ending at 1 (defaults to 1, a single analysis); with one trigger per look, one value per look in (0, 1] (`NA` at calendar-driven looks). |
| `time` | Strictly increasing calendar times of the calendar-driven looks (with one trigger per look, one value per look, `NA` at event-driven looks). |
| `test` | `NULL` (all endpoints at all looks), or a list with one element per look giving the endpoints tested at the look (`"os"`, `"pfs"` or both). A character vector with one endpoint per look is also accepted. |

## Details

With one value of `trigger`, all looks are driven by the events of the same endpoint (`"os"` or `"pfs"`, with `fraction`) or take place at calendar times (`"calendar"`, with `time`). The final number of events is given in `ms_power` through `events` or `followup`, and the look k takes place when the expected number of events of the trigger endpoint reaches `fraction[k]` times the final number.

With one value of `trigger` per look, each look has its own trigger. `fraction` then has one value per look, `NA` at the calendar-driven looks, and `time` one value per look, `NA` at the event-driven looks. The fractions of the looks driven by the events of an endpoint refer to the final number of events of that endpoint (see `ms_power`) and must be strictly increasing; an event-driven final look must have the fraction 1. The calendar times of the looks, which depend on the scenario, must be strictly increasing.

By default every look tests all endpoints of the test (`ms_test`). `test` gives the endpoints tested at each look, for example `list("pfs", c("pfs", "os"), "os")` for an interim analysis of PFS, the final analysis of PFS with an interim analysis of OS, and the final analysis of OS.

## Value

An object of class `ms_looks`.

## See Also

`ms_power`, `ms_size`

## Examples

```
ms_looks("os", fraction = c(0.5, 0.75, 1))
ms_looks("calendar", time = c(24, 36))
# PFS interim and final analyses driven by PFS events, an OS interim
# analysis at the final PFS analysis and the final OS analysis driven by
# OS events
ms_looks(c("pfs", "pfs", "os"), fraction = c(0.6, 1, 1),
         test = list("pfs", c("pfs", "os"), "os"))
```

---

# ms_model *Multistate model for one treatment group*

---

## Description

Defines the multistate model of one group: progression-free and alive (state 1), progressed without switch (state 2), switched (state 3) and dead (state 4), and, when `switch_pre` is positive, progression-free after a switch before progression (state 5). Progression-free survival (PFS) is the time to leave the progression-free states and overall survival (OS) is the time to enter state 4.

## Usage

```
ms_model(
  time = 0,
  prog = 0,
  death,
  death_prog = NULL,
  switch_prob = 0,
  switch_rate = 0,
  death_switch = NULL,
  dropout = 0,
  time_post = 0,
  clock = c("reset", "forward"),
  switch_clock = c("same", "switch"),
  switch_af = NULL,
  switch_pre = 0,
  prog_switch = NULL,
  death_switch_pre = NULL,
  dropout_pfs = 0
)
```

## Arguments

| | |
|---|---|
| time | Interval start times for the pre-progression intensities and the dropout hazard. Must start at 0 and be strictly increasing. |
| prog | Progression hazard in each interval (length 1 or length(time)). |
| death | Hazard of death before progression in each interval. |
| death_prog | Hazard of death after progression without switch, in each interval of time_post. Required when prog is positive in some interval. |
| switch_prob | Probability of switching immediately at progression, in each interval of time. |
| switch_rate | Hazard of switching after progression, in each interval of time_post. |
| death_switch | Hazard of death after a switch, in each interval of time_post. Defaults to death_prog, that is, switching has no effect on survival. |
| dropout | Dropout hazard in each interval of time. |
| time_post | Interval start times for the post-progression intensities. Must start at 0 and be strictly increasing. |
| clock | Clock of the post-progression intensities: "reset" (time since progression) or "forward" (time since randomization). |
| switch_clock | Clock of death_switch: "same" (the clock of the other post-progression intensities) or "switch" (time since the switch). |
| switch_af | Acceleration factor of an accelerated failure time effect of the switch (positive), or NULL (default) for the hazard death_switch. When given, death_switch must not be given and switch_clock must be "same". |
| switch_pre | Hazard of switching before progression, in each interval of time. |
| prog_switch | Hazard of progression after a switch before progression, in each interval of time. Defaults to prog. |
| death_switch_pre | Hazard of death before progression after a switch, in each interval of time. Defaults to death. |
| dropout_pfs | Hazard of a censoring of PFS only, in each interval of time. |

## Details

Before progression, the intensities are piecewise constant in the time since randomization, with intervals starting at time: progression (prog), death before progression (death), and the probability switch_prob of switching immediately at progression (evaluated at the time of progression). After progression without switch, the patient dies with intensity death_prog or switches with intensity switch_rate; after a switch, the patient dies with intensity death_switch. The post-progression intensities are piecewise constant with intervals starting at time_post.

With clock = "reset" (default) the post-progression intensities are functions of the time since progression (a semi-Markov model). With clock = "forward" they are functions of the time since randomization (a Markov model). The two clocks give the same model when the post-progression intensities are constant.

The death hazard after a switch at the post-progression clock value $\sigma$ is, at the clock value $\tau$, death_switch at $\tau$ with switch_clock = "same" (default), and death_switch at $\tau - \sigma$ (the time

since the switch) with switch_clock = "switch". With clock = "reset" the two differ only for switches after progression (switch_rate); with clock = "forward" they differ also for switches at progression. With switch_af given, the effect of a switch is an accelerated failure time effect: the death hazard after the switch is $\phi c(\sigma + \phi(\tau - \sigma))$, where $\phi$ is switch_af and $c$ is death_prog, so that the remaining time is that without switch divided by $\phi$ (the rank preserving structural failure time model of Robins and Tsiatis 1991; $\phi < 1$ prolongs survival).

A switch before progression moves a patient from state 1 to state 5 with hazard switch_pre. In state 5 the patient progresses with hazard prog_switch and dies with hazard death_switch_pre; at progression the patient enters state 3 and then dies with the hazard death_switch on the post-progression clock, whatever switch_clock (with switch_af, the hazard switch_af * death_prog). Whether a switch before progression censors PFS is set in ms_scenario (pfs_switch).

Dropout is piecewise exponential with intervals starting at time, independent of the event process, and censors both endpoints. The hazard dropout_pfs adds a censoring that acts on PFS only (for example the end of tumor assessments), independent of everything else.

A model without progression (prog = 0) is a piecewise exponential model for a single endpoint with hazard death. With death_prog = death, switch_prob = 0 and switch_rate = 0, the model is that of Fleischer et al. (2009); see ms_fleischer.

## Value

An object of class ms_model: a list with the intensity vectors expanded to the lengths of time and time_post, clock, switch_clock, switch_af and pfs_switch ("event"; set by ms_scenario). With switch_af, the element death_switch holds switch_af * death_prog, the death hazard after a switch when death_prog is constant.

## References

Fleischer F, Gaschler-Markefski B, Bluhmki E (2009). A statistical model for the dependence between progression-free survival and overall survival. Statistics in Medicine 28:2669-2686.

Robins JM, Tsiatis AA (1991). Correcting for non-compliance in randomized trials using rank preserving structural failure time models. Communications in Statistics - Theory and Methods 20:2609-2631.

## See Also

ms_fleischer, ms_scenario, ms_surv, ms_prob

## Examples

```
# Single endpoint: exponential survival with a median of 12 months
ms_model(death = log(2) / 12)

# Progression, with 30 percent of the patients switching at progression
ms_model(time = c(0, 6), prog = c(0.10, 0.08), death = 0.01,
         death_prog = 0.05, switch_prob = 0.3, death_switch = 0.03)

# Switching after progression with an accelerated failure time effect
ms_model(prog = 0.1, death = 0.01, death_prog = 0.05, switch_rate = 0.04,
         switch_af = 0.7)
```

```
# Switching before progression, and censoring of PFS only
ms_model(prog = 0.1, death = 0.01, death_prog = 0.05, death_switch = 0.03,
         switch_pre = 0.02, dropout_pfs = 0.01)
```

---

`ms_null` *Null scenario*

---

## Description

Replaces the model of the treatment group by the model of the control group, keeping the allocation and the accrual.

## Usage

```
ms_null(scenario)
```

## Arguments

| | |
|---|---|
| `scenario` | An `ms_scenario` object. |

## Value

An `ms_scenario` object under the null hypothesis.

## See Also

`ms_scenario`

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
lr_moments(ms_null(scn), n = 400, calendar_time = 30)
```

`ms_power` *Power of a group sequential log-rank design*

## Description

Computes the analysis times, expected events, bounds, stopping probabilities, power and type I error of a design with a given sample size, without simulation.

## Usage

```
ms_power(
  scenario,
  n,
  looks,
  test,
  events = NULL,
  followup = NULL,
  null_scenarios = NULL
)
```

## Arguments

| | |
|---|---|
| `scenario` | An `ms_scenario` object. |
| `n` | Total sample size. |
| `looks` | An `ms_looks` object. |
| `test` | An `ms_test` object. |
| `events` | Final number of events of the trigger endpoint (for event-driven looks), or final numbers of events named by endpoint (`"os"`, `"pfs"`). |
| `followup` | Time from the end of accrual to the final analysis (for an event-driven final look, instead of `events`). |
| `null_scenarios` | For two endpoints, an optional list of `ms_scenario` objects with the allocation and accrual of `scenario`, each named by the endpoint (`"os"` or `"pfs"`) whose null hypothesis is true in the scenario, for example `list(os = s_os)`. |

## Details

For event-driven looks, the final analysis takes place when the expected number of events of the trigger endpoint reaches `events`, or at `followup` after the end of accrual. The interim analyses take place when the expected number of events reaches the fractions given in `looks`. For calendar-driven looks, `events` and `followup` are not used.

At each look, the standardized log-rank statistics have the mean and covariance of `lr_cov` with the approximations chosen in `test`. The bounds are computed by `gs_bounds` and the stopping probabilities by `gs_prob`. The power is the probability of crossing an efficacy bound when the futility bounds are applied. The type I error is computed under `ms_null` applied to the scenario,

with event-driven looks at the same numbers of events of the trigger endpoint; the non-binding value ignores the futility bounds and the binding value applies them.

The power and the type I error rest on the normal approximation of the log-rank statistics. With unequal allocation and few events the statistic is skewed, so that the probability of rejecting in favor of the treatment exceeds the computed type I error when the treatment group is the larger group, and falls below it when the control group is: by simulation (`inst/validation` of the source package), about 0.028 instead of 0.025 with 2:1 allocation and looks at 75 and 150 deaths.

The hazard ratios at the bounds are `exp(-z / sqrt(p0 p1 d))`, where d is the expected number of events of the tested endpoint.

The expected study duration, number of events of the tested endpoint and number of enrolled patients (`expected`) are the sums over the looks of the probability of stopping at the look times the calendar time, the expected events and the enrolled patients at the look, under the design scenario and under `ms_null` applied to the scenario (at the looks of the type I error, `looks_null`). The trial stops at an interim look when the statistic crosses the efficacy bound or falls below the futility bound, also when the futility bounds are non-binding, as in `gsProbability()` of gsDesign; the probability of stopping at the last look is one minus the probability of stopping earlier. With non-binding futility bounds the efficacy bounds do not depend on the futility bounds, so that the expected values without stopping for futility are those of the same design with `futility = "none"`. At an event-driven look the events are the target, and the calendar time and the enrolled patients are those at which the expected events reach the target: the variability of the analysis time and its correlation with the decision to stop are ignored, as in the expected study durations of rpact and lrstat. At a calendar-driven look the time is exact and the events are expected. A final look at `followup` after the end of accrual takes place under the null scenario when the expected events reach those of the design scenario, as for the type I error.

When the looks have different triggers (see `ms_looks`), each trigger endpoint has its own final number of events, and a look driven by it takes place when its expected number of events reaches the fraction of the look times that number. The final numbers are given by `events`, named by endpoint when the looks are driven by the events of two endpoints, for example `c(pfs = 300, os = 250)`. With `followup`, or without `events` when the final look is calendar-driven, they are the expected numbers of events at the final analysis; the looks driven by an endpoint other than that of the final look then need fractions below 1.

With two endpoints (`test` from `ms_test` with two endpoints), the statistics of the two endpoints at the looks at which they are tested have the joint distribution of `lr_cov`, with the observed events of the trigger endpoints determining the event-driven looks for `variance = "studentized"`. Each endpoint has its own bounds, and the rejection probabilities under the multiple testing procedure are computed by `gs_prob_mtp`; with recycling of the level (`transition` of `ms_test`), each endpoint also has bounds at its level after recycling. The familywise error rate is computed under `ms_null` applied to the scenario (the global null hypothesis), where the correlations between the endpoints are those of Danzer et al. (2025, Theorem 1), and under each scenario of `null_scenarios`, where the hypothesis of one endpoint is true and that of the other is not (a partial null hypothesis). Such scenarios cannot be derived from the multistate model of the design scenario and are given by the user, for example a model with the same OS in both groups and a difference in PFS. Under each null scenario the event-driven looks take place at the same numbers of events of their trigger endpoints.

With switching from a calendar time (`switch_start` of `ms_scenario`), only a test of one endpoint with `mean = "direct"` for OS is covered, and an OS test cannot use looks driven by PFS events. For OS the covariance of the statistics comes from the per-patient influence functions (see `lr_cov`); the bounds still come from the information or event fractions, and the type I error under `ms_null`

uses the exact correlations under the null hypothesis, so it can differ from alpha when the switching changes the death hazard.

## Value

An object of class ms_power, a list with

**looks** A data frame with one row per look: time, enrolled, the expected events of the tested endpoint (events, events_control, events_treatment), info_frac, the efficacy bound (z_upper, nominal one-sided p-value p_upper, hazard ratio hr_upper), the futility bound (z_lower, hr_lower), the mean and standard deviation of the statistic (z_mean, z_sd), the probabilities of stopping for efficacy and futility at the look (prob_upper, prob_lower), the cumulative power cum_power, the cumulative alpha spent alpha_spent and the cumulative non-binding type I error cum_alpha. When the looks are driven by another endpoint, events_trigger gives the events of that endpoint.

**power** Overall power.

**type1** Type I error, non-binding and binding.

**n, accrual_duration, study_duration, events** Sample size, accrual duration, calendar time of the final analysis and expected events of the tested endpoint at the final analysis.

**expected** A data frame with the expected values when the trial stops at the bounds (see Details), with one row per hypothesis (hypothesis: "design" for the design scenario and "null" for the null scenario): the probabilities of stopping at an interim look for efficacy (prob_efficacy_early) and for futility (prob_futility_early), and the expected study_duration, events of the tested endpoint and enrolled patients.

**mean, sigma** The mean vector and covariance matrix of the statistics at the looks under the design scenario.

**targets** The numbers of events of the trigger endpoints at which the event-driven looks take place (NA for calendar-driven looks).

**settings, scenario, looks_def, test** The inputs.

**looks_null** A data frame with one row per look under the null scenario: look, time, enrolled, events and the probabilities of stopping for efficacy and for futility at the look with the futility bounds applied (prob_upper, prob_lower).

**sigma_null** The correlation matrix of the statistics under the null scenario.

With two endpoints, looks has one row per endpoint and look at which the endpoint is tested, with the columns above and endpoint, look, trigger and events_trigger (the events of the trigger endpoint at the look, NA for calendar-driven looks); prob_upper is the probability of rejecting the hypothesis of the endpoint at the look under the multiple testing procedure (with recycling of the level, z_upper_recycled and p_upper_recycled give the efficacy bound and its nominal p-value at the level after the other hypothesis has been rejected, after hr_upper), prob_lower the probability of stopping for futility at the look (first endpoint), cum_power the cumulative sum of prob_upper, alpha_spent the cumulative alpha of the bounds of the endpoint and cum_alpha the cumulative non-binding probability of rejecting the hypothesis of the endpoint under the global null hypothesis. power has the powers of the two endpoints (named by endpoint), the probability of rejecting both hypotheses (both) and of rejecting at least one (any); type1 is the familywise error rate under the global null hypothesis, non-binding and binding; fwer is a data frame with the familywise error rate under the global null hypothesis and under each scenario of null_scenarios;

`events` has the expected events of each endpoint at its last look; `final_events` the final numbers of events of the trigger endpoints; and `mean` and `sigma` are the mean vector and covariance matrix of the statistics in the order of the rows of `looks`. The expected values (`expected`, `looks_null`, `sigma_null`) and `targets` are not given.

## References

Danzer MF, Rufibach K, Beyersmann J, Schmidt R (2025). Exhausting the type I error level in event-driven group-sequential designs with a closed testing procedure for progression-free and overall survival. arXiv:2512.08658.

## See Also

`ms_size`, `ms_looks`, `ms_test`, `gs_prob_mtp`

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
         test = ms_test(), events = 300)

# PFS first, then OS: final PFS analysis at 300 PFS events with an OS
# interim analysis, final OS analysis at 220 OS events
fl <- ms_scenario(ms_fleischer(4, 12), ms_fleischer(6, 16),
                  accrual = ms_accrual(rate = 25))
lk <- ms_looks(c("pfs", "os"), fraction = c(1, 1), test = list(c("pfs", "os"), "os"))
ms_power(fl, n = 450, looks = lk,
         test = ms_test(endpoints = c("pfs", "os"), mtp = "fixed_sequence"),
         events = c(pfs = 300, os = 220))
```

---

# ms_prob *State occupation probabilities*

---

## Description

Probabilities of being in each state of the multistate model at given times since randomization, without dropout.

## Usage

```
ms_prob(model, time)
```

## Arguments

| | |
|---|---|
| `model` | An `ms_model` object. |
| `time` | Times since randomization (nonnegative). |

### Details

The probability of being progression-free and alive is `exp(-H(t))`, where H is the cumulative hazard of progression and death. The probabilities of the progressed and switched states are integrals over the time of progression of the progression density times the transition probabilities after progression, which have closed forms for piecewise constant intensities. The integrals are computed with Gauss-Legendre rules. With a switch before progression, the probability of the switched progression-free state also has a closed form, and the switched state includes the patients who progressed after such a switch.

### Value

A data frame with `time` and the probabilities `progression_free`, `progressed` (without switch), `switched` and `dead`. With `switch_pre` positive in some interval, `progression_free` is the probability of being progression-free without a switch, and the column `progression_free_switched` (progression-free after a switch before progression) follows it.

### See Also

`ms_model`, `ms_surv`

### Examples

```
m <- ms_model(time = c(0, 6), prog = c(0.10, 0.08), death = 0.01,
              death_prog = 0.05, switch_prob = 0.3, death_switch = 0.03)
ms_prob(m, time = c(6, 12, 24))
```

---

## `ms_scenario` *Two-group trial scenario*

---

### Description

Combines the models of the control and treatment groups, the allocation ratio and the accrual.

### Usage

```
ms_scenario(
  control,
  treatment,
  allocation = 1,
  accrual,
  pfs_switch = c("event", "censor"),
  switch_start = 0
)
```

## Arguments

| | |
|---|---|
| control, treatment | ms_model objects of the control and treatment groups. |
| allocation | Allocation ratio, treatment to control (for example 2 for 2:1). |
| accrual | An ms_accrual object. |
| pfs_switch | Definition of PFS at a switch before progression: "event" (followed up to progression or death) or "censor" (censored at the switch). It applies to both groups. |
| switch_start | Calendar time (time since the start of accrual) from which switching after progression is possible, nonnegative. The default 0 means that switching is possible from randomization, as in the models. |

## Details

pfs_switch defines PFS for patients who switch before progression (switch_pre in ms_model). With "event", the switch does not change the definition: PFS is the time to progression or death. With "censor", PFS is censored at the switch (for example when the start of a new treatment censors PFS); the log-rank test of PFS then compares the hazard prog + death of the patients who are progression-free and have not switched.

switch_start is a calendar time (time since the start of accrual) before which nobody switches, for example when switching is allowed only after an interim analysis. A patient enrolled at time e can then switch from the follow-up time max(switch_start - e, 0) on: progressions before that time do not lead to a switch at progression, and patients who progressed before it can switch with switch_rate from that time on. The distribution of OS then depends on the entry time, and the computations for OS integrate over the entry time. They cover the OS log-rank statistic alone with mean = "direct"; two endpoints together, an OS test at looks driven by PFS events, and a PFS test at looks driven by OS events with a variance other than "null" give an error. PFS is not affected by the switching. Under the null hypothesis the score no longer has independent increments when the switching changes the death hazard: the correlations between the looks are computed from the per-patient influence functions instead of the information (see lr_cov). switch_start cannot be combined with switching before progression (switch_pre).

## Value

An object of class ms_scenario, a list with the two models, allocation, the allocation probabilities prob (control, treatment), accrual, pfs_switch and switch_start. The models carry pfs_switch as well.

## See Also

ms_model, ms_accrual, ms_null

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
```

```
# Switching after progression in the control group, possible only from
# month 15 (for example after an interim analysis)
ctl_sw <- ms_model(prog = 0.1, death = 0.01, death_prog = 0.05,
                   switch_prob = 0.3, switch_rate = 0.04, death_switch = 0.03)
trt_sw <- ms_model(prog = 0.06, death = 0.008, death_prog = 0.045)
ms_scenario(ctl_sw, trt_sw, accrual = ms_accrual(rate = 20), switch_start = 15)
```

---

`ms_size` *Number of events, sample size, accrual duration or follow-up for a target power*

---

## Description

Solves for one design parameter so that the design of `ms_power` has the target power, with the other parameters fixed.

## Usage

```
ms_size(
  scenario,
  looks,
  test,
  power = 0.9,
  solve_for = c("events", "n", "accrual_duration", "followup"),
  n = NULL,
  followup = NULL,
  interval = NULL,
  round_up = TRUE,
  target = c("first", "each", "both", "any"),
  null_scenarios = NULL
)
```

## Arguments

| | |
|---|---|
| `scenario` | An `ms_scenario` object. |
| `looks` | An `ms_looks` object. |
| `test` | An `ms_test` object. |
| `power` | Target power. |
| `solve_for` | `"events"`, `"n"`, `"accrual_duration"` or `"followup"`. |
| `n` | Total sample size (fixed for `"events"`, `"followup"` and `"accrual_duration"`). |
| `followup` | Time from the end of accrual to the final analysis (fixed for `"n"` and `"accrual_duration"` with event-driven looks). |
| `interval` | Optional search interval for the unknown (for `"events"` and `"followup"`, an interval of the calendar time of the final analysis). |

| | |
|---|---|
| round_up | Whether to round the number of events or the sample size up to an integer. |
| target | For two endpoints, "first", "each", "both" or "any" (see Details). |
| null_scenarios | For two endpoints, null scenarios for the familywise error rate of the solved design (see ms_power). |

## Details

The unknown is one of

- "events": the final number of events of the trigger endpoint, with n and the accrual fixed (event-driven looks);
- "followup": the time from the end of accrual to the final analysis, with n and the accrual fixed (event-driven looks);
- "n": the sample size, with the accrual rates (or, for an accrual given by proportions, the accrual duration) fixed and the final analysis at followup after the end of accrual (event-driven looks) or at the calendar times of the looks;
- "accrual_duration": the accrual duration of an accrual given by proportions, with n fixed and the final analysis at followup after the end of accrual (event-driven looks) or at the calendar times of the looks.

For "events" and "followup" the search is over the calendar time of the final analysis, so that the two give the same design (before rounding). The search doubles or halves the unknown until the power brackets the target, evaluates the power on a grid within the bracket (or within interval) and refines the first grid interval in which the target is reached, so that the smallest solution is returned when the power is not monotone. Beta spending uses one minus power as the total type II error.

When the looks are driven by the events of two endpoints, the final numbers of events of both are their expected numbers at the final analysis (see ms_power), so that the looks driven by the endpoint other than that of the final look need fractions below 1. With "events" the solution then has the final numbers of events of both trigger endpoints, each rounded up with round_up = TRUE.

With two endpoints, target chooses the quantity that reaches power: the power of the first endpoint ("first"), the smaller of the powers of the two endpoints ("each"), the probability of rejecting both hypotheses ("both") or that of rejecting at least one ("any"), all under the multiple testing procedure of test.

## Value

An object of class c("ms_size", "ms_power"): the result of ms_power for the solved design, with solve_for, the unrounded solution (named by endpoint for the final numbers of events of two trigger endpoints), the target_power and, for two endpoints, the target.

## See Also

ms_power

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
ms_size(scn, looks = ms_looks("os", fraction = c(0.5, 1)),
        test = ms_test(), power = 0.9, solve_for = "events", n = 500)


# Sample size for a power of 80 per cent for both PFS and OS, tested in
# this order at one analysis 12 months after the end of accrual
fl <- ms_scenario(ms_fleischer(4, 12), ms_fleischer(6, 16),
                  accrual = ms_accrual(rate = 25))
ms_size(fl, looks = ms_looks("os"),
        test = ms_test(endpoints = c("pfs", "os"), mtp = "fixed_sequence"),
        power = 0.8, solve_for = "n", followup = 12, target = "each")
```

---

# ms_surv *Survival function, hazard and density of an endpoint*

---

## Description

Evaluates the distribution of progression-free survival or overall survival implied by an `ms_model`, without dropout.

## Usage

```
ms_surv(model, time, endpoint = c("os", "pfs"))
```

## Arguments

| | |
|---|---|
| `model` | An `ms_model` object. |
| `time` | Times since randomization (nonnegative). |
| `endpoint` | `"os"` or `"pfs"`. |

## Details

PFS has the hazard `prog` + `death`. OS combines death before progression with death after progression, with or without switch; its hazard is not piecewise constant when `prog` is positive. With a switch before progression (`switch_pre`), PFS is the time to progression or death whatever `pfs_switch` of `ms_scenario`, and its hazard is not piecewise constant.

## Value

A data frame with `time`, the survival function `surv`, the hazard `haz` (`NA` where `surv` is 0) and the density `dens`.

## See Also

ms_model, ms_prob

## Examples

```
m <- ms_fleischer(median_pfs = 6, median_os = 15)
ms_surv(m, time = c(6, 15), endpoint = "os")
```

---

`ms_test` *Test and stopping rules*

---

## Description

Defines the one-sided log-rank test, the efficacy and futility bounds and the approximation of the distribution of the statistics.

## Usage

```
ms_test(
  alpha = 0.025,
  endpoints = "os",
  sf_eff = "OF",
  param_eff = NULL,
  futility = c("none", "fixed", "beta"),
  fut_bound = NULL,
  fut_scale = c("z", "hr"),
  sf_fut = "OF",
  param_fut = NULL,
  beta = 0.1,
  binding = FALSE,
  mean = c("direct", "schoenfeld"),
  variance = c("null", "alternative", "studentized"),
  spending_time = c("events", "information"),
  mtp = c("none", "bonferroni", "fixed_sequence"),
  weights = NULL,
  hierarchical = c("overall", "stagewise"),
  transition = c(0, 0)
)
```

## Arguments

| | |
|---|---|
| `alpha` | One-sided type I error (with two endpoints, the familywise error rate). |
| `endpoints` | Tested endpoint, `"os"` or `"pfs"`, or both in the order of testing (for example `c("pfs", "os")`). With two endpoints the futility bounds are placed on the first, and with `mtp = "fixed_sequence"` the first is tested first. |

sf_eff, param_eff
: Alpha spending function and its parameter (see gs_spend). For "user", param_eff gives the cumulative proportions of alpha at the looks. With two endpoints, sf_eff has one value or one value per endpoint, and param_eff is a list with one element per endpoint, or a number or one number per endpoint.

futility
: "none", "fixed" or "beta" (for the first endpoint).

fut_bound
: Fixed futility bounds at the interim looks (for futility = "fixed").

fut_scale
: Scale of fut_bound: "z" or "hr".

sf_fut, param_fut
: Beta spending function and its parameter.

beta
: Total type II error for beta spending in ms_power; ms_size uses one minus the target power instead.

binding
: Whether the futility bounds are binding.

mean
: Approximation of the mean of the statistic: "direct" or "schoenfeld" (see lr_moments).

variance
: Approximation of the covariance of the statistics: "null", "alternative" or "studentized" (see lr_cov).

spending_time
: "events" or "information".

mtp
: Multiple testing procedure: "none" (one endpoint), "bonferroni" or "fixed_sequence" (two endpoints).

weights
: Bonferroni weights of the two endpoints (positive, adding up to 1; defaults to equal weights).

hierarchical
: "overall" or "stagewise" (for mtp = "fixed_sequence").

transition
: For mtp = "bonferroni", the fractions of the level passed on after a rejection, c(g12, g21): g12 from the first endpoint to the second after the first hypothesis is rejected and g21 from the second to the first after the second is rejected (numbers between 0 and 1). The default c(0, 0) means no recycling.

## Details

The efficacy bounds are obtained by alpha spending under the null hypothesis. The correlation of the statistics under the null hypothesis is taken as sqrt(d_j / d_k), where d are the expected numbers of events of the tested endpoint at the looks; it is exact under the null hypothesis when the looks are driven by the events of the tested endpoint. The spending time is the event fraction (spending_time = "events") or the information fraction under the design scenario ("information").

Futility bounds may be fixed, on the Z scale (fut_scale = "z") or on the hazard ratio scale (fut_scale = "hr", converted with z = -log(hr) sqrt(p0 p1 d)), or obtained by beta spending under the design scenario. With binding = FALSE the efficacy bounds ignore the futility bounds.

With two endpoints, each endpoint has its own efficacy bounds, computed as above from the looks at which it is tested (see ms_looks). With mtp = "bonferroni", the endpoint k is tested at level weights[k] * alpha. By default the levels are not recycled after a rejection. With transition = c(g12, g21), the level of a rejected hypothesis is passed on to the other in the graphical procedure of Maurer and Bretz (2013, Algorithm 1): after the first hypothesis has been rejected, the second is tested at level (weights[2] + g12 * weights[1]) * alpha, and after the second has been rejected,

the first at level (weights[1] + g21 * weights[2]) * alpha. The bounds at the level after recycling are computed with the spending function of the endpoint for all its looks (Maurer and Bretz 2013, Section 3.2). The procedure controls the familywise error rate when the nominal levels of the spending functions do not decrease with the level (Maurer and Bretz 2013, Section 3.1). The spending functions of gs_spend other than "OF" are proportional to the level and have this property, and "OF" has it for levels below 0.318 (Maurer and Bretz 2013, p. 313). ms_power checks at every look that the bounds after recycling do not exceed the bounds at the initial level and stops with an error otherwise. With binding futility bounds, the bounds of the first endpoint after recycling are computed with the futility bounds of the initial level as binding bounds. A hypothesis rejected at an analysis makes the other be tested at its level after recycling at the same analysis (see gs_prob_mtp). With mtp = "fixed_sequence", both endpoints are tested at level alpha and the second endpoint is tested only after the first has been rejected: with hierarchical = "overall" at the analysis at which the first is rejected and at its later looks until it is rejected (procedure (b) of Glimm et al. 2010), and with "stagewise" only at the analysis at which the first is rejected, where the trial stops (procedure (a) of Glimm et al. 2010, Tamhane et al. 2010). The bounds of the second endpoint do not depend on the correlation between the endpoints; with them the familywise error rate is at most alpha (Glimm et al. 2010, Tamhane et al. 2010). Futility bounds are placed on the first endpoint only: the trial stops when its statistic falls below the futility bound at an interim look at which it has not been rejected. See gs_prob_mtp for the stopping rules.

The efficacy bounds and the non-binding type I error (with two endpoints, the familywise error rate under the global null hypothesis) do not depend on variance; the power, the futility bounds by beta spending and the familywise error rate under null_scenarios of ms_power do. In the simulations of inst/validation/ with PFS and OS tested together, the powers with "null" and "studentized" agreed with the simulated powers within the Monte Carlo error, "alternative" gave slightly larger powers, and "studentized" agreed best with the simulated standard deviations and correlations of the statistics.

With "alternative" and "studentized", the covariance matrix of the statistics of an endpoint is not that of independent increments, and the probabilities over four or more of its looks are computed with the Genz-Bretz algorithm (see gs_prob).

## Value

An object of class ms_test.

## References

Glimm E, Maurer W, Bretz F (2010). Hierarchical testing of multiple endpoints in group-sequential trials. Statistics in Medicine 29:219-228.

Maurer W, Bretz F (2013). Multiple testing in group sequential trials using graphical approaches. Statistics in Biopharmaceutical Research 5:311-320.

Tamhane AC, Mehta CR, Liu L (2010). Testing a primary and a secondary endpoint in a group sequential design. Biometrics 66:1174-1184.

## See Also

ms_power, gs_bounds, gs_prob_mtp

## Examples

```
ms_test()
ms_test(futility = "beta", sf_fut = "HSD", param_fut = -2)
ms_test(futility = "fixed", fut_bound = 1, fut_scale = "hr")
# PFS first, then OS, with Pocock-type spending for OS
ms_test(endpoints = c("pfs", "os"), sf_eff = c("OF", "Pocock"),
        mtp = "fixed_sequence")
ms_test(endpoints = c("pfs", "os"), mtp = "bonferroni", weights = c(0.8, 0.2))
# The same with the level of PFS passed on to OS after the rejection of
# PFS, and the level of OS to PFS after the rejection of OS
ms_test(endpoints = c("pfs", "os"), mtp = "bonferroni", weights = c(0.8, 0.2),
        transition = c(1, 1))
```

---

# plot.ms_power *Plot an ms_power object*

---

## Description

Draws the bounds of a group sequential design, the cumulative probabilities of stopping at the looks, or the survival functions of the groups of the design scenario.

## Usage

```
## S3 method for class 'ms_power'
plot(
  x,
  type = c("bounds", "power", "survival"),
  scale = c("z", "hr", "p"),
  x_axis = c("events", "time"),
  max_time = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x | An ms_power object. |
| type | "bounds", "power" or "survival". |
| scale | Scale of the bounds for "bounds": "z", "hr" or "p". |
| x_axis | Horizontal axis for "bounds" and "power": the expected events of the tested endpoint ("events") or the calendar time ("time"). |
| max_time | Largest time since randomization for "survival" (defaults to the calendar time of the final analysis, x$study_duration). |
| ... | Named arguments passed to plot(), such as main, xlim, ylim, xlab and ylab, which replace the defaults. |

## Details

"bounds" The efficacy and futility bounds at the looks on the scale scale: the standardized log-rank statistic ("z", with the mean of the statistic under the design scenario as a dashed line), the hazard ratio ("hr", hr_upper and hr_lower of x$looks) or the nominal one-sided p-value ("p", on a log scale, with pnorm(z_lower, lower.tail = FALSE) for the futility bounds). The futility bound of the last look is the efficacy bound, to which the line of the futility bounds is drawn.

"power" The cumulative probabilities of stopping for efficacy (cum_power of x$looks) and for futility at the interim looks under the design scenario, and the cumulative non-binding type I error under the null scenario (cum_alpha), drawn at the looks under the null scenario (x$looks_null).

"survival" The survival functions of OS and PFS of the control and treatment groups (ms_surv) from 0 to max_time. PFS is drawn when it differs from OS in a group, that is when the model allows progression. With switching from a calendar time (switch_start of ms_scenario), OS is that of the patients who enter after the start of switching, as given by ms_surv, which a note above the plot states.

The colors are those of grDevices::palette.colors(palette = "Okabe-Ito"), and the default range of the vertical axis leaves room for the legend above the values of "bounds" (below them for "p") and "power". For designs with two endpoints only "survival" is available.

## Value

The plotted values, invisibly: for "bounds" a data frame with look, x (events or calendar time), the efficacy bound upper, the futility bound lower (NA where there is none) and, for "z", the mean mean; for "power" a data frame with look, x, cum_efficacy, cum_futility, x_null (events or calendar time of the looks under the null scenario) and cum_alpha; for "survival" a data frame with time, group, endpoint, surv and the legend label label, with the note on OS as the attribute "note" when switching starts at a calendar time.

## See Also

ms_power, ms_surv

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
des <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 0.75, 1)),
                test = ms_test(futility = "beta", sf_fut = "HSD", param_fut = -2),
                events = 300)
plot(des)
plot(des, type = "bounds", scale = "hr", x_axis = "time")
plot(des, type = "power")
plot(des, type = "survival", max_time = 48)
```

## print.ms_accrual *Print an ms_accrual object*

### Description

Print an ms_accrual object

### Usage

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

### Arguments

| | |
|---|---|
| x | An ms_accrual object. |
| ... | Not used. |

### Value

x, invisibly.

### Examples

```
print(ms_accrual(time = c(0, 3), rate = c(10, 30)))
```

## print.ms_cond_power *Print an* ms_cond_power *object*

### Description

Print an ms_cond_power object

### Usage

```
## S3 method for class 'ms_cond_power'
print(x, digits = 4, ...)
```

### Arguments

| | |
|---|---|
| x | An ms_cond_power object. |
| digits | Number of significant digits. |
| ... | Not used. |

## Value

x, invisibly.

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
des <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
                test = ms_test(), events = 300)
print(ms_cond_power(des, look = 1, z = 1.2, under = "trend"))
```

---

# print.ms_looks *Print an ms_looks object*

---

## Description

Print an ms_looks object

## Usage

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

## Arguments

| | |
|---|---|
| x | An ms_looks object. |
| ... | Not used. |

## Value

x, invisibly.

## Examples

```
print(ms_looks("os", fraction = c(0.5, 1)))
print(ms_looks(c("pfs", "os"), fraction = c(1, 1), test = list(c("pfs", "os"), "os")))
```

## print.ms_model *Print an ms_model object*

### Description

Print an ms_model object

### Usage

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

### Arguments

| | |
|---|---|
| x | An ms_model object. |
| ... | Not used. |

### Value

x, invisibly.

### Examples

```
print(ms_fleischer(median_pfs = 6, median_os = 15))
```

## print.ms_power *Print an ms_power object*

### Description

Print an ms_power object

### Usage

```
## S3 method for class 'ms_power'
print(x, digits = 4, ...)
```

### Arguments

| | |
|---|---|
| x | An ms_power object. |
| digits | Number of significant digits. |
| ... | Not used. |

## Value

x, invisibly.

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
res <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
                test = ms_test(), events = 300)
print(res)
```

---

# print.ms_scenario *Print an ms_scenario object*

---

## Description

Print an ms_scenario object

## Usage

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

## Arguments

| | |
|---|---|
| x | An ms_scenario object. |
| ... | Not used. |

## Value

x, invisibly.

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
print(ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20)))
```

## print.ms_size *Print an ms_size object*

### Description

Print an ms_size object

### Usage

```
## S3 method for class 'ms_size'
print(x, digits = 4, ...)
```

### Arguments

| | |
|---|---|
| x | An ms_size object. |
| digits | Number of significant digits. |
| ... | Passed to print.ms_power. |

### Value

x, invisibly.

### Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
res <- ms_size(scn, looks = ms_looks("os"), test = ms_test(),
               power = 0.9, solve_for = "events", n = 500)
print(res)
```

## print.ms_test *Print an ms_test object*

### Description

Print an ms_test object

### Usage

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

### Arguments

| | |
|---|---|
| x | An ms_test object. |
| ... | Not used. |

### Value

x, invisibly.

### Examples

```
print(ms_test(futility = "beta"))
print(ms_test(endpoints = c("pfs", "os"), mtp = "fixed_sequence"))
print(ms_test(endpoints = c("pfs", "os"), mtp = "bonferroni",
              weights = c(0.8, 0.2), transition = c(1, 1)))
```

---

## print.summary.ms_power

*Print a summary.ms_power object*

---

### Description

Print a summary.ms_power object

### Usage

```
## S3 method for class 'summary.ms_power'
print(x, digits = 4, ...)
```

### Arguments

| | |
|---|---|
| x | A summary.ms_power object. |
| digits | Number of significant digits. |
| ... | Not used. |

### Value

x, invisibly.

### Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
res <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
                test = ms_test(), events = 300)
print(summary(res))
```

summary.ms_power *Summarize an ms_power object*

## Description

Summarize an ms_power object

## Usage

```
## S3 method for class 'ms_power'
summary(object, ...)
```

## Arguments

| | |
|---|---|
| object | An ms_power object. |
| ... | Not used. |

## Value

An object of class summary.ms_power: a list with the design overview (overview), the per-look table (looks) and the settings (settings); for one endpoint also the expected values when the trial stops at the bounds (expected, see ms_power), and for two endpoints the familywise error rates (fwer).

## Examples

```
ctl <- ms_model(death = log(2) / 12)
trt <- ms_model(death = 0.7 * log(2) / 12)
scn <- ms_scenario(ctl, trt, accrual = ms_accrual(rate = 20))
res <- ms_power(scn, n = 400, looks = ms_looks("os", fraction = c(0.5, 1)),
                test = ms_test(), events = 300)
summary(res)
```

# Index


as.data.frame.ms_power, 3

gs_bounds, 4, *6*, *9*, *11*, *26*, *37*
gs_prob, *5*, 6, *8*, *9*, *12*, *17*, *26*, *37*
gs_prob_mtp, 7, *27*, *29*, *37*
gs_spend, *4*, *5*, 10, *36*, *37*

lr_cov, *7*, *9*, 11, *14*, *26*, *27*, *31*, *36*
lr_moments, *12*, 13, *18*, *36*

ms_accrual, 15, *31*
ms_cond_power, 16
ms_event_time, *18*, 19
ms_events, *14*, 18, *19*
ms_fleischer, 20, *24*
ms_looks, 21, *27*, *29*, *36*
ms_model, *18*, *20*, 22, *30*, *31*, *35*
ms_null, 25, *26*, *27*, *31*
ms_power, *5*, *16*, *17*, *21*, *22*, 26, *32*, *33*, *36*, *37*, *39*, *46*
ms_prob, *24*, 29, *35*
ms_scenario, *12*, *14*, *15*, *18*, *24*, *25*, *27*, 30, *34*, *39*
ms_size, *15*, *16*, *22*, *29*, 32, *36*
ms_surv, *24*, *30*, 34, *39*
ms_test, *6*, *16*, *21*, *27*, *29*, 35

plot.ms_power, 38
print.ms_accrual, 40
print.ms_cond_power, 40
print.ms_looks, 41
print.ms_model, 42
print.ms_power, 42, *44*
print.ms_scenario, 43
print.ms_size, 44
print.ms_test, 44
print.summary.ms_power, 45

summary.ms_power, 46