Keaven M. Anderson

gsSurv Technical Manual

Placeholder Name

gsSurv Technical Manual

ISBN-1234567891234

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et elementum libero. In hac habitasse platea dictumst. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec sed odio dui. Nullam quis risus eget urna mollis ornare vel eu leo. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Curabitur blandit tempus porttitor. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.

Placeholder for dedication text

Table of contents

Welcome

Welcome to the **gsSurv Technical Manual** by Keaven M. Anderson. You can view the HTML version at https://keaven.github.io/gsSurv/. Thanks to Nan Xiao for help with document production.

Source code and documentation for survival analysis using `gsSurv()` and associated routines.

Overview

This was developed with the literate programming (https://en.wikipedia.org/wiki/Literate_programming) method proposed by Donald Knuth to produce self-documenting code. Currently it is out of date with the associated code in the **gsDesign** R package. Since some the algorithm used for sample size is an extension of the Lachin and Foulkes method, this also serves as documentation of extension of the methods. There is a possible objective of publishing at arXiv.org since the methodology has not otherwise been fully documented.

Ongoing and planned initiatives

- Created repository at https://github.com/keaven/gsSurv (2023/02/05)
- See summary of updates on the gsSurv2.pdf document generated

Preface

We consider a 2-arm group sequential trial with an experimental and control group and a time-to-event endpoint. Such trials are common, for example, in cardiovascular disease and oncology. For instance, a drug for improving lipids could be compared to a standard drug to see if cardiovascular endpoints such as heart attacks or death could be prevented. In oncology trials, a new treatment might be compared to a control to see if survival or time-to-progression might be extended. Such trials can be large and complex. This technical report documents software in the gsDesign R package for designing such trials. In fact, the report is intended as a tool to thoroughly document the sample size functionality in the gsDesign R package that is supported by the `nSurv()` and `gsSurv()` functions. Some common issues that arise that are dealt with by this software are:

- All designs assume a proportional hazards model.
- Piecewise constant enrollment rates are allowed given that enrollment may be slow initially during study ramp-up.
- Piecewise exponential failure rates are allowed to allow changing outcome incidence over time.
- Piecewise exponential censoring rates allow changing dropout rates by treatment group and over time.
- A patient population may be stratified to allow subgroups with different underlying enrollment rates as well as differing incidence rates for endpoints and censoring.
- The software computes expected study durations and timing of interim analyses as well as the expected enrollment and number of events at interim analyses.
- Statistical bounds are given in terms of Z-value cutoffs for test statistics as well as approximate hazard ratios required to cross such bounds; additional computations available for more general group sequential designs are also available from other gsDesign package routines.

In the following, we assume a given Type I error, timing of interim analyses, endpoint event rates and censoring rates. There are two general approaches to carrying out calculations.

1. Assuming fixed accrual duration and minimum follow-up duration determine the accrual rate and sample size required to provide the desired power.
2. Assume given accrual rates over time and vary accrual or follow-up duration to achieve a desired power.

A group sequential design is generally defined by Type I and Type II error, the number and timing of interim analyses and the boundaries at interim analyses. For many endpoint types, timing is linear in the number of evaluable observations. For time-to-event endpoints, timing is related to the number of endpoint events available for analysis, which is generally a non-linear function of accrual rates and duration, endpoint event rates, censoring rates and duration of follow-up after accrual termination. Lachin and Foulkes [1986] have provided a general formulation for calculating the expected number of events over time in the scenarios described above. They also derive the sample size method used here for the case where enrollment and study durations are fixed, which requires deriving enrollment rates required to achieve the desired Type I error and power. Their methods for calculating power are extended to compute power for similarly complex designs, but where enrollment or follow-up duration are varied to achieve power given fixed enrollment rates; such methods are analogous to those of Kim and Tsiatis [1990].

This document is organized based on supporting the following calculations:

- Computing the probability that any individual entering the study will have an endpoint observed as a function of random enrollment, event, and censoring times. This calculation, presented in Section 2, is key to all other quantities considered. The routine provided in the gsDesign package documented in Section 2 is `eEvents()`.
- Designing a fixed (single analysis) trial with a time-to-event endpoint is presented in Section 3 where the `nSurv()` routine in the gsDesign package is defined.
- Designing a group sequential trial with a time-to-event endpoint is presented in Section 4, where the gsDesign function `gsSurv()` is defined. Supporting routines in gsDesign to compute the expected number of events at a given point in time (`nEventsIA()`) or the time required for the expected number of events to reach a given proportion of the final planned number (`tEventsIA()`).

Each section presents extensive examples to show how to use the key routines.

Summary of updates

- August, 2015. Updated `nEventsIA()` to work correctly for stratified populations when the `simple = TRUE` option is chosen. This also impacted `tEventsIA()` for stratified populations.
- August, 2023 (ONGOING)
  - Updated `gsSurv()` to enable power computation
  - Created `gsSurvCalendar()` to enable calendar-based interim analysis planning
  - Incorporated revisions of code that have been put into gsDesign package
  - Fixing stratified sample size computation
  - Added `EDC0`, `EDC1` to output of `gsSurv()` to match help file output specification.
- April, 2025. Migrate from Sweave document to Quarto book.

Chapter 1

Computing expected number of events over time

1.1 General formulation

We will consider a single treatment group and stratum and assume subjects enroll according to a Poisson process with entry rate $g(t) \geq 0$ for $t > 0$. Denote the number of subjects enrolled at or before time t by $N(t)$ for $t \geq 0$. The expected number of subjects enrolled by time t is simply

$$E\{N(t)\} = G(t) = \int_0^t g(s)ds. \tag{1.1}$$

We assume the study runs from time 0 to some time $t_m > 0$ where m is a positive integer we will further define shortly. We assume the time to event is independent and identically distributed for all subjects enrolled. We also assume the censoring time is independent and identically distributed for all subjects enrolled. For an individual, let $X > 0$ denote the duration from entry into the trial until an event and $Y > 0$ denote the duration from entry into the trial until loss-to-follow-up. The duration of time a subject is followed is $T = \min(X, Y)$. We assume the pair X, Y is independent of entry time.

Denote the number of events observed by time t by $D(t)$ for $t \geq 0$. The expected number of events observed by time t_m is

$$E\{D(t_m)\} = \int_0^{t_m} P\{X \leq t_m - u, X \leq Y\}g(u)du. \tag{1.2}$$

With a change of variables from u to $t = t_m - u$ we have

$$E\{D(t_m)\} = \int_0^{t_m} P\{X \leq t, X \leq Y\}g(U \leq t_m - t)dt. \tag{1.3}$$

1.2 Piecewise failure and enrollment distibutions

Here we define a piecewise random variable and key related probabilities and follow with a corresponding general formulation for calculating $P\{\delta = 1\}$.

1.2.1 General definitions

Recall that m is a positive integer. For $i = 1, 2, \ldots, m$ we assume We assume $0 = t_0 < t_1 < \ldots < t_m$ and for $i = 1, 2, \ldots, m$ let $s_i = t_i - t_{i-1}$. We define random variables that will be used to define X and Y on a piecewise basis over the intervals $(t_{i-1}, t_i]$, $i = 1, 2, \ldots, m$ as follows.

- For $i = 1, 2, \ldots, m$, we assume $X_i > 0$, $Y_i > 0$ are random variables that are independent of the study entry time U.
- We let X_i and Y_i define X and Y, respectively, on the interval $(t_{i-1}, t_i]$, $i = 1, 2, \ldots, m$, as follows.
  - If $X_1 \leq t_1 = s_1$, then $X = X_1$.
  - For $i = 2, 3, \ldots, m-1$ recursively define $X = t_{i-1} + X_i$ if X has not been yet defined and $X_i \leq s_i$.
  - If X has not been otherwise defined above, let $X = t_{m-1} + X_m$.

Y is defined analogously by $Y_1, Y_2, \ldots, Y_m$.

For $i = 1, 2, \ldots, m$ we assume

$$q_i = P\{X_i > s_i, Y_i > s_i\} \tag{1.4}$$

$$Q_i \equiv P\{X > t_i, Y > t_i\} = \prod_{j=1}^{i} P\{X_i > s_i, Y_i > s_i\} = \prod_{j=1}^{i} q_j. \tag{1.5}$$

$$p_i = P\{t_{i-1} < X \leq t_i, X \leq Y\} = Q_{i-1} P\{X_i \leq s_i, X_i \leq Y_i\} \tag{1.6}$$

and let

$$P_i \equiv P\{T \leq t_i, X \leq Y\} = \sum_{j=1}^{i} p_j. \tag{1.7}$$

As an example, if X_i, Y_i, $i = 1, 2, \ldots, m$ are all independent these probability assumptions are automatically satisified.

We will break the calculation of interest into intervals. Denote

$$E\{D(t_m)\} = \sum_{i=1}^{m} A_i \tag{1.8}$$

where for $i = 1, 2, \ldots, m$

$$A_i = \int_{t_{i-1}}^{t_i} P\{X \le t_i, X \le Y\} g(t_m - t) dt$$
$$= P_{i-1}(G(t_m - t_{i-1}) - G(t_m - t_i)) + Q_{i-1} B_i \tag{1.9}$$

where

$$B_i = \int_0^{s_i} P\{X_i \le s, X_i \le Y_i\} g(t_m - (t_{i-1} + s)) ds \tag{1.10}$$

We are now left to calculate p_i, q_i, and B_i, $i = 1, 2, \ldots, m$ given specific distributional assumptions. Applying equations (1.4)-(1.10) we can then compute $E\{D(t_m)\}$.

1.2.2 Definition of a piecewise exponential failure time

For $i = 1, 2, \ldots, m$ we assume X_i and Y_i are independent and exponentially distributed with failure rates λ_i and η_i, respectively, where X_i will be referred to as an event time and Y_i a censoring or lost-to-follow-up time. The random variables X_i and Y_i will be used to define T and δ over the interval $(t_{i-1}, t_i]$. We define $\delta_i = 1$ if $\min(X_i, Y_i) \le s_i$ and $\delta_i = 0$ otherwise, $i = 1, 2, \ldots, m$. We will let J be the first interval i in which $\min(X_i, Y_i) \le s_i$ and $J = m + 1$ if $\min(X_i, Y_i) > s_i$, $i = 1, 2, \ldots, m$. We define the time to event $T \equiv t_m$ if $J = m + 1$ and $T \equiv t_{J-1} + \min(X_J, Y_J)$, otherwise. We define the indicator that T is an event time rather than a lost-to-follow-up time by the random variable $\delta' = 1$ if $J < m + 1$ and $T = t_{J-1} + X_J$. The random variable T will be referred to as a piecewise exponential failure time with event indicator δ'.

1.2.3 Exponential failure and censoring

We assume $m = 1$, $X \sim$ Exponential(λ) and $Y \sim$ Exponential(η). We assume uniform entry over a time $(0, r]$ where $0 < r \le t_1$ which implies $g(u) = 1/r$ on $(0, r]$ and is 0 elsewhere. It is straightforward to calculate that

$$P\{X \le t, X \le Y\} = \frac{\lambda}{\lambda + \eta}(1 - e^{-(\lambda+\eta)t})$$

From (1.9) we only need calculate

$$\begin{aligned} E\{D(t_m)\} &= B_1 \\ &= \int_0^{t_1} \frac{\lambda}{\lambda+\eta}\left(1-e^{-(\lambda+\eta)t}\right) g(t_1-t)dt \\ &= \frac{\lambda}{(\lambda+\eta)r}\int_{t_1-r}^{t_1}\left(1-e^{-(\lambda+\eta)t}\right)dt \\ &= \frac{\lambda}{\lambda+\eta}\left(1-\frac{e^{-(\lambda+\eta)(t_1-r)}-e^{-(\lambda+\eta)t_1}}{(\lambda+\eta)r}\right). \end{aligned} \tag{1.11}$$

which corresponds with Lachin and Foulkes [1986].

With entry following a truncated exponential distribution for $0 < u \le r < t_1$ and $\gamma \neq 0$ and $\gamma \neq \lambda+\eta$

$$g(u) = \frac{e^{-\gamma u}}{1-e^{-\gamma r}}$$

it is straightforward to show

$$E\{D(t_1)\} = \int_{t_1-r}^{t_1} \frac{\lambda}{\lambda+\eta}(1-e^{-(\lambda+\eta)t})\gamma e^{-\gamma(t_1-t)}dt \tag{1.12}$$

$$= \frac{\lambda}{\lambda+\eta}\left(1+\frac{\gamma e^{-(\lambda+\eta)t_1}}{\lambda+\eta-\gamma}\left(\frac{1-e^{(\lambda+\eta-\gamma)r}}{1-e^{-\gamma r}}\right)\right) \tag{1.13}$$

which again is consistent with Lachin and Foulkes [1986]. If $\gamma = \lambda+\eta$, it is straightforward to show either directly by integrating (1.12) or by applying l'Hôpital's rule to (1.13) that

$$E\{D(t_1)\} = \lambda\left(\frac{1}{\gamma}-\frac{re^{-\gamma t_1}}{1-e^{-\gamma r}}\right). \tag{1.14}$$

1.2.4 Piecewise constant enrollment

We now assume subjects enroll at a constant rate in each of m intervals. Without loss of generality, we assume that for $i = 0, 1, 2, \ldots, m$, that the enrollment rate is [g(t)= _{m+1-i} 0] for t in the interval $(t_m - t_i, t_m - t_{i-1})$. If subject recruiting does not occur throughout the course of the trial, some of the γ_i values may be 0.

1.2.5 Expected number of events

We now apply equations (1.4)-(1.10) to compute $P\{\delta = 1\}$ using the following.

$$q_i = e^{-(\lambda_i+\eta_i)s_i} \tag{1.15}$$

$$p_i = Q_{i-1}\frac{\lambda_i}{\lambda_i+\eta_i}(1-e^{-(\lambda_i+\eta_i)s_i}) \tag{1.16}$$

For $0 \le s < s_i$ we have $t_m - t_i < t_m - (t_{i-1} - s) \le t_m - t_i$ which implies $g(t_m - (t_{i-1} - s)) = \gamma_{m+1-i}$ which implies

$$\begin{aligned} B_i &= \int_0^{s_i} \frac{\gamma_{m+1-i}\lambda_i}{\lambda_i+\eta_i}(1-e^{-(\lambda_i+\eta_i)s})ds \\ &= \frac{\gamma_{m+1-i}\lambda_i}{\lambda_i+\eta_i}\left(s_i - \frac{1-e^{-(\lambda_i+\eta_i)s_i}}{\lambda_i+\eta_i}\right). \end{aligned} \tag{1.17}$$

We note that when $\lambda_i + \eta_i = 0$ that $p_i = 0$ and $B_i = 0$ by definition; this is accounted for in `eEvents1()`. As an exercise, show that (1.15)-(1.17) can be used along with (1.4)-(1.10) to produce (1.11).

1.2.6 Testing event count calculations

We assume a study duration of $T = 20$ months. Enrollment is assumed to occur over the course of 5 months at a rate of 5 per month for the first 2 months, 10 per month for 1 month, and 20 per month for months 4 and 5. This results in an expected enrollment of 60. The uniform rates to reflect enrollment probability over time for a given study subject are 1/12 from 0 to 2, 2/12=1/6 from 2 to 3, and 4/12=1/3 from 3 to 5; enrollment rates are 0 thereafter. The exponential event rate is assumed to be high initially with a rate of .05 in the first month, .02 for the second month, and .01 per month thereafter. The rate of censoring is assumed to be exponential with a rate of .03 per month from time of enrollment through 20 months. Based on this, the t_i values are 0, 1, 2, 3, 5 and corresponding timepoints at the end of the study 20, 19, 18, 17, 15. Summing A_i from 1 to 6 yields $E\{D(20)\} = 11.023$. The calculations in the following table were computed in an Excel spreadsheet and can be used for checking the following calculations in R.

i	t_i	γ_{m+1-i}	λ_i	η_i	q_i	Q_i	p_i	P_i	B_i	A_i
1	1	0	.05	.01	0.9418	0.9418	0.0485	0.0485	0.0000	0.0000
2	2	0	.02	.01	0.9704	0.9139	0.0186	0.0671	0.0000	0.0000
3	15	0	.01	.01	0.7711	0.7047	0.1046	0.1717	0.0000	0.0000
4	17	20	.01	.01	0.9608	0.6771	0.0138	0.1855	0.3947	7.1464
5	18	10	.01	.01	0.9802	0.6637	0.0067	0.1922	0.0497	1.8889
6	20	5	.01	.01	0.9608	0.6376	0.0130	0.2052	0.0987	1.9877

1.2.7 R routine for computing expected number of events for a single stratum

Calculations for the expected number of events as just outlined are performed using the function `eEvents1()`. This is not accessible when the gsDesign package is loaded since input variable checks are not performed for this lower-level function. The more general function `eEvents()` will be presented in the next section. It performs the same calculations for a stratified population.

Following are the arguments for `eEvents1()`:

- `lambda`: a vector of event rates
- `eta`: a vector of dropout rates; if length is 1, this will be transformed to the length of `lambda`
- `gamma`: a vector of enrollment rates
- `R`: a vector of durations for enrollment rates; should have the same length as `gamma`; note that if `sum(R) > Tfinal - minfup` then the actual accrual is limited to `Tfinal - minfup` but that the output `R` is still as input
- `S`: a vector of durations of failure rate periods. If `lambda` has length 1, this should be `NULL`; otherwise, this should have length 1 less than `lambda`. The final failure rate in `lambda` is assumed to continue indefinitely, and thus is not specified in `S`
- `T`: a scalar indicating the time at which the expected number of events and enrollment is to be computed
- `Tfinal`: a scalar indicating the time at which the trial is to be completed. If `NULL`, this is assumed the same as `T`
- `minfup`: a scalar indicating the minimum follow-up for each observation at the end of the trial. Note that `min(Tfinal - minfup, sum(R))` is when enrollment ends and thus we must have `Tfinal > minfup`.

The reader may skip the next section as examples later demonstrate application of the routing.

1.2.8 The eEvents1 source code

```
# Do not edit source code in gsSurv.R
# This should be modified in https://github.com/keaven/gsSurv
# eEvents1 function [sinew] ----
eEvents1 <- function(lambda = 1, eta = 0, gamma = 1, R = 1, S = NULL,
                     T = 2, Tfinal = NULL, minfup = 0, simple = TRUE) {
  if (is.null(Tfinal)) {
    Tfinal <- T
    minfupia <- minfup
  } else {
```

```
  minfupia <- max(0, minfup - (Tfinal - T))
}

nlambda <- length(lambda)
if (length(eta) == 1 & nlambda > 1) {
  eta <- rep(eta, nlambda)
}
T1 <- cumsum(S)
T1 <- c(T1[T1 < T], T)
T2 <- T - cumsum(R)
T2[T2 < minfupia] <- minfupia
i <- 1:length(gamma)
gamma[i > length(unique(T2))] <- 0
T2 <- unique(c(T, T2[T2 > 0]))
T3 <- sort(unique(c(T1, T2)))
if (sum(R) >= T) T2 <- c(T2, 0)
nperiod <- length(T3)
s <- T3 - c(0, T3[1:(nperiod - 1)])

lam <- rep(lambda[nlambda], nperiod)
et <- rep(eta[nlambda], nperiod)
gam <- rep(0, nperiod)

for (i in length(T1):1)
{
  indx <- T3 <= T1[i]
  lam[indx] <- lambda[i]
  et[indx] <- eta[i]
}
for (i in min(length(gamma) + 1, length(T2)):2) {
  gam[T3 > T2[i]] <- gamma[i - 1]
}
q <- exp(-(c(lam) + c(et)) * s)
Q <- cumprod(q)
indx <- 1:(nperiod - 1)
Qm1 <- c(1, Q[indx])
p <- c(lam) / (c(lam) + c(et)) * c(Qm1) * (1 - c(q))
p[is.nan(p)] <- 0
P <- cumsum(p)
B <- c(gam) / (c(lam) + c(et)) * c(lam) * (c(s) - (1 - c(q)) / (c(lam) + c(et)))
B[is.nan(B)] <- 0
A <- c(0, P[indx]) * c(gam) * c(s) + c(Qm1) * c(B)
if (!simple) {
  return(list(
```

```
      lambda = lambda, eta = eta, gamma = gamma, R = R, S = S,
      T = T, Tfinal = Tfinal, minfup = minfup, d = sum(A),
      n = sum(gam * s), q = q, Q = Q, p = p, P = P, B = B, A = A, T1 = T1,
      T2 = T2, T3 = T3, lam = lam, et = et, gam = gam
    ))
  } else {
    return(list(
      lambda = lambda, eta = eta, gamma = gamma, R = R, S = S,
      T = T, Tfinal = Tfinal, minfup = minfup, d = sum(A),
      n = sum(c(gam) * c(s))
    ))
  }
}
```

```
eEvents1(
  lambda = c(1, 2, 3, 4), eta = 0, gamma = c(1, 2), R = c(1, 5), S = c(1.5, 2.5, 3.5),
  T = 10, Tfinal = NULL, minfup = 0, simple = FALSE
)
```

```
$lambda
[1] 1 2 3 4

$eta
[1] 0 0 0 0

$gamma
[1] 1 2

$R
[1] 1 5

$S
[1] 1.5 2.5 3.5

$T
[1] 10

$Tfinal
[1] 10

$minfup
[1] 0

$d
[1] 10.999
```

```
$n
[1] 11

$q
[1] 2.231302e-01 6.737947e-03 2.753645e-05 2.478752e-03 1.831564e-02

$Q
[1] 2.231302e-01 1.503439e-03 4.139938e-08 1.026188e-10 1.879529e-12

$p
[1] 7.768698e-01 2.216267e-01 1.503398e-03 4.129676e-08 1.007393e-10

$P
[1] 0.7768698 0.9984966 1.0000000 1.0000000 1.0000000

$B
[1] 0.0000000 0.0000000 6.3333517 2.5012394 0.7545789

$A
[1] 0.000000 0.000000 6.998998 3.000000 1.000000

$T1
[1]  1.5  4.0  7.5 10.0

$T2
[1] 10  9  4

$T3
[1]  1.5  4.0  7.5  9.0 10.0

$lam
[1] 1 2 3 4 4

$et
[1] 0 0 0 0 0

$gam
[1] 0 0 2 2 1
```

We provide an example where both $\lambda_i = 0$ and $\eta_i = 0$ for some i. This option will be used later when follow-up is cut off at some point in time for a study.

```
eEvents1(
  lambda = c(1, 2, 3, 0), eta = 0, gamma = c(1, 2), R = c(1, 5), S = c(1.5, 2.5, 3.5),
```

```
  T = 10, Tfinal = NULL, minfup = 0, simple = FALSE
)
```

```
$lambda
[1] 1 2 3 0

$eta
[1] 0 0 0 0

$gamma
[1] 1 2

$R
[1] 1 5

$S
[1] 1.5 2.5 3.5

$T
[1] 10

$Tfinal
[1] 10

$minfup
[1] 0

$d
[1] 10.999

$n
[1] 11

$q
[1] 2.231302e-01 6.737947e-03 2.753645e-05 1.000000e+00 1.000000e+00

$Q
[1] 2.231302e-01 1.503439e-03 4.139938e-08 4.139938e-08 4.139938e-08

$p
[1] 0.776869840 0.221626721 0.001503398 0.000000000 0.000000000

$P
[1] 0.7768698 0.9984966 1.0000000 1.0000000 1.0000000
```

```
$B
[1] 0.000000 0.000000 6.333352 0.000000 0.000000

$A
[1] 0.000000 0.000000 6.998998 3.000000 1.000000

$T1
[1]  1.5  4.0  7.5 10.0

$T2
[1] 10  9  4

$T3
[1]  1.5  4.0  7.5  9.0 10.0

$lam
[1] 1 2 3 0 0

$et
[1] 0 0 0 0 0

$gam
[1] 0 0 2 2 1
```

1.2.9 Extending expected number of events calculations to a stratified population

We extend our notation to describe a stratified population here, to show the calculations we wish to perform.

The stratified case is made reasonably general, allowing enrollment, event rates and dropout rates to change over time in whatever piecewise fashion is required. Without loss of generality, we assume the same time intervals for constant enrollment and failure rates as before. If these differ by strata, then any time where a change is made for any stratum is included for all strata in the notation. Whereas we had vectors representing enrollment rates, failure rates and censoring rates previously we now have matrices with the column of each matrix representing a vector for each stratum. In an effort to make this explicit, we provide notation. We assume there are $s \geq 1$ strata in the population. For $1 \leq i \leq m$ we previously let λ_i represent a failure rate in the i-th time period. We now let λ represent a $m \times s$ matrix with failure rates for a stratum in each column and for $1 \leq i \leq m$, $1 \leq j \leq s$ we let λ_{ij} of λ represent the failure rate in the i-th time period and j-th stratum. We let the

$m \times s$ matrix η represent the analogous set of dropout rates. We assume n enrollment rate periods and we let γ represent an $n \times s$ matrix of enrollment rates for each time period.

1.2.10 eEvents source code

The routine `eEvents()` works for a stratified as well as an unstratified population. It is accessible to users when the gsDesign package is loaded as checking is performed on input variables. The generality implied in the matrix notation just defined is available through `eEvents()`.

```
# eEvents roxy [sinew] ----
#' Expected number of events for a time-to-event study
#'
#' `eEvents()` is used to calculate the expected number of events for a
#' population with a time-to-event endpoint.  It is based on calculations
#' demonstrated in Lachin and Foulkes (1986) and is fundamental in computations
#' for the sample size method they propose. Piecewise exponential survival and
#' dropout rates are supported as well as piecewise uniform enrollment. A
#' stratified population is allowed. Output is the expected number of events
#' observed given a trial duration and the above rate parameters.
#'
#' \code{eEvents()} produces an object of class \code{eEvents} with the number
#' of subjects and events for a set of pre-specified trial parameters, such as
#' accrual duration and follow-up period. The underlying power calculation is
#' based on Lachin and Foulkes (1986) method for proportional hazards assuming
#' a fixed underlying hazard ratio between 2 treatment groups. The method has
#' been extended here to enable designs to test non-inferiority. Piecewise
#' constant enrollment and failure rates are assumed and a stratified
#' population is allowed. See also \code{\link{nSurvival}} for other Lachin and
#' Foulkes (1986) methods assuming a constant hazard difference or exponential
#' enrollment rate.
#'
#' \code{print.eEvents()} formats the output for an object of class
#' \code{eEvents} and returns the input value.
#'
#' @param lambda scalar, vector or matrix of event hazard rates; rows represent
#' time periods while columns represent strata; a vector implies a single
#' stratum.
#' @param eta scalar, vector or matrix of dropout hazard rates; rows represent
#' time periods while columns represent strata; if entered as a scalar, rate is
#' constant across strata and time periods; if entered as a vector, rates are
#' constant across strata.
```

```
#' @param gamma a scalar, vector or matrix of rates of entry by time period
#' (rows) and strata (columns); if entered as a scalar, rate is constant
#' across strata and time periods; if entered as a vector, rates are constant
#' across strata.
#' @param R a scalar or vector of durations of time periods for recruitment
#' rates specified in rows of \code{gamma}. Length is the same as number of
#' rows in \code{gamma}. Note that the final enrollment period is extended as
#' long as needed.
#' @param S a scalar or vector of durations of piecewise constant event rates
#' specified in rows of \code{lambda}, \code{eta} and \code{etaE}; this is NULL
#' if there is a single event rate per stratum (exponential failure) or length
#' of the number of rows in \code{lambda} minus 1, otherwise.
#' @param T time of analysis; if \code{Tfinal=NULL}, this is also the study
#' duration.
#' @param Tfinal Study duration; if \code{NULL}, this will be replaced with
#' \code{T} on output.
#' @param minfup time from end of planned enrollment (\code{sum(R)} from output
#' value of \code{R}) until \code{Tfinal}.
#' @param x an object of class \code{eEvents} returned from \code{eEvents()}.
#' @param digits which controls number of digits for printing.
#' @param ... Other arguments that may be passed to the generic print function.
#' @return \code{eEvents()} and \code{print.eEvents()} return an object of
#' class \code{eEvents} which contains the following items: \item{lambda}{as
#' input; converted to a matrix on output.} \item{eta}{as input; converted to a
#' matrix on output.} \item{gamma}{as input.} \item{R}{as input.} \item{S}{as
#' input.} \item{T}{as input.} \item{Tfinal}{planned duration of study.}
#' \item{minfup}{as input.} \item{d}{expected number of events.}
#' \item{n}{expected sample size.} \item{digits}{as input.}
#' @examples
#'
#' # 3 enrollment periods, 3 piecewise exponential failure rates
#' str(eEvents(
#'   lambda = c(.05, .02, .01), eta = .01, gamma = c(5, 10, 20),
#'   R = c(2, 1, 2), S = c(1, 1), T = 20
#' ))
#'
#' # control group for example from Bernstein and Lagakos (1978)
#' lamC <- c(1, .8, .5)
#' n <- eEvents(
#'   lambda = matrix(c(lamC, lamC * 2 / 3), ncol = 6), eta = 0,
#'   gamma = matrix(.5, ncol = 6), R = 2, T = 4
#' )
#'
#' @aliases print.eEvents
```

```
#' @author Keaven Anderson \email{keaven_anderson@@merck.com}
#' @seealso \link{gsDesign package overview}, \link{plot.gsDesign},
#' \code{\link{gsDesign}}, \code{\link{gsHR}},
#' \code{\link{nSurvival}}
#' @references Lachin JM and Foulkes MA (1986), Evaluation of Sample Size and
#' Power for Analyses of Survival with Allowance for Nonuniform Patient Entry,
#' Losses to Follow-Up, Noncompliance, and Stratification. \emph{Biometrics},
#' 42, 507-519.
#'
#' Bernstein D and Lagakos S (1978), Sample size and power determination for
#' stratified clinical trials. \emph{Journal of Statistical Computation and
#' Simulation}, 8:65-73.
#' @keywords design
#' @rdname eEvents
#' @export
# eEvents function [sinew] ----
eEvents <- function(lambda = 1, eta = 0, gamma = 1, R = 1, S = NULL, T = 2,
                    Tfinal = NULL, minfup = 0, digits = 4) {
  if (is.null(Tfinal)) {
    if (minfup >= T) {
      stop("Minimum follow-up greater than study duration.")
    }
    Tfinal <- T
    minfupia <- minfup
  } else {
    minfupia <- max(0, minfup - (Tfinal - T))
  }

  if (!is.matrix(lambda)) {
    lambda <- matrix(lambda, nrow = length(lambda))
  }
  if (!is.matrix(eta)) {
    eta <- matrix(eta, nrow = nrow(lambda), ncol = ncol(lambda))
  }
  if (!is.matrix(gamma)) {
    gamma <- matrix(gamma, nrow = length(R), ncol = ncol(lambda))
  }
  n <- rep(0, ncol(lambda))
  d <- n
  for (i in 1:ncol(lambda))
  {
    # KA: updated following line with as.vector statements 10/16/2017
    a <- eEvents1(
      lambda = as.vector(lambda[, i]), eta = as.vector(eta[, i]),
```

```
      gamma = as.vector(gamma[, i]), R = R, S = S, T = T,
      Tfinal = Tfinal, minfup = minfup
    )
    n[i] <- a$n
    d[i] <- a$d
  }
  T1 <- cumsum(S)
  T1 <- unique(c(0, T1[T1 < T], T))
  nper <- length(T1) - 1
  names1 <- round(T1[1:nper], digits)
  namesper <- paste("-", round(T1[2:(nper + 1)], digits), sep = "")
  namesper <- paste(names1, namesper, sep = "")
  if (nper < dim(lambda)[1]) {
    lambda <- matrix(lambda[1:nper, ], nrow = nper)
  }
  if (nper < dim(eta)[1]) {
    eta <- matrix(eta[1:nper, ], nrow = nper)
  }
  rownames(lambda) <- namesper
  rownames(eta) <- namesper
  colnames(lambda) <- paste("Stratum", 1:ncol(lambda))
  colnames(eta) <- paste("Stratum", 1:ncol(eta))
  T2 <- cumsum(R)
  T2[T - T2 < minfupia] <- T - minfupia
  T2 <- unique(c(0, T2))
  nper <- length(T2) - 1
  names1 <- round(c(T2[1:nper]), digits)
  namesper <- paste("-", round(T2[2:(nper + 1)], digits), sep = "")
  namesper <- paste(names1, namesper, sep = "")
  if (nper < length(gamma)) {
    gamma <- matrix(gamma[1:nper, ], nrow = nper)
  }
  rownames(gamma) <- namesper
  colnames(gamma) <- paste("Stratum", 1:ncol(gamma))
  x <- list(
    lambda = lambda, eta = eta, gamma = gamma, R = R,
    S = S, T = T, Tfinal = Tfinal,
    minfup = minfup, d = d, n = n, digits = digits
  )
  class(x) <- "eEvents"
  return(x)
}
```

1.2.11 Print function for `eEvents()`

Here we provide a print function for output generated by `eEvents()`. This will be demonstrated along with `eEvents()` in the next section. A supportive function `periods()` used by the print function for `eEvents()` is shown first. This is not made visible to users of the gsDesign package.

```
# periods,
# file = "R/periods.R"
```

```
# print.eEvents roxy [sinew] ----
#' @rdname eEvents
#' @export
# print.eEvents function [sinew] ----
print.eEvents <- function(x, digits = 4, ...) {
  if (!inherits(x, "eEvents")) {
    stop("print.eEvents: primary argument must have class eEvents")
  }
  cat("Study duration:              Tfinal=",
    round(x$Tfinal, digits), "\n",
    sep = ""
  )
  cat("Analysis time:                    T=",
    round(x$T, digits), "\n",
    sep = ""
  )
  cat("Accrual duration:                   ",
    round(min(
      x$T - max(0, x$minfup - (x$Tfinal - x$T)),
      sum(x$R)
    ), digits), "\n",
    sep = ""
  )
  cat("Min. end-of-study follow-up: minfup=",
    round(x$minfup, digits), "\n",
    sep = ""
  )
  cat("Expected events (total):            ",
    round(sum(x$d), digits), "\n",
    sep = ""
  )
  if (length(x$d) > 1) {
    cat(
      "Expected events by stratum:       d=",
      round(x$d[1], digits)
```

```
    )
    for (i in 2:length(x$d)) {
      cat(paste("", round(x$d[i], digits)))
    }
    cat("\n")
  }
  cat("Expected sample size (total):       ",
    round(sum(x$n), digits), "\n",
    sep = ""
  )
  if (length(x$n) > 1) {
    cat(
      "Sample size by stratum:           n=",
      round(x$n[1], digits)
    )
    for (i in 2:length(x$n)) {
      cat(paste("", round(x$n[i], digits)))
    }
    cat("\n")
  }
  nstrata <- dim(x$lambda)[2]
  cat("Number of strata:                   ",
    nstrata, "\n",
    sep = ""
  )
  cat("Accrual rates:\n")
  print(round(x$gamma, digits))
  cat("Event rates:\n")
  print(round(x$lambda, digits))
  cat("Censoring rates:\n")
  print(round(x$eta, digits))
  return(x)
}
```

1.2.12 Example calculations in R

We now perform the computation from Section 1.2.5 using `eEvents()`, the function available for users from the gsDesign package. We begin with a single stratum with piecewise exponential failure and piecewise continuous enrollment.

```
eEvents(
  lambda = c(.05, .02, .01), eta = .01, gamma = c(5, 10, 20),
```

```
  R = c(2, 1, 2), S = c(1, 1), T = 20
)
```

```
  Study duration:              Tfinal=20
  Analysis time:                    T=20
  Accrual duration:                   5
  Min. end-of-study follow-up: minfup=0
  Expected events (total):            11.023
  Expected sample size (total):       60
  Number of strata:                   1
  Accrual rates:
      Stratum 1
  0-2         5
  2-3        10
  3-5        20
  Event rates:
       Stratum 1
  0-1       0.05
  1-2       0.02
  2-20      0.01
  Censoring rates:
       Stratum 1
  0-1       0.01
  1-2       0.01
  2-20      0.01
```

Following is an example with multiple strata.

```
x <- eEvents(
  lambda = matrix(c(.05, .02, .01, .1, .04, .02), nrow = 3),
  eta = .01, gamma = matrix(c(5, 10, 20, 5, 10, 20), nrow = 3),
  R = c(2, 1, 2), S = c(1, 1), T = 20
)
x$n
```

```
  [1] 60 60
```

```
x$d
```

```
  [1] 11.02302 19.95135
```

The variables `lambda`, `eta`, and `gamma` provide the failure rates, censoring rates and enrollment rates, respectively. `R` is a scalar or vector with the duration of each enrollment rate period. The variable `S` specifies the duration of initial failure rate periods; should be `NULL` if only one failure rate period or 1 less than the number of failure rate periods, otherwise. After the initial two failure rate periods, the final failure rate is assumed to continue indefinitely.

Each of these can be a scalar, vector or matrix; see the `eEvents()` help file for further clarification.

An additional capability for these routines is to incorporate a minimum follow-up requirement at the end of the study. We will want to use the routine for both final and interim timepoints. To appropriately incorporate the minimum follow-up at final analysis, the minimum end-of-study follow-up (`minfup` which defaults to 0) and the time at the end of the study (`Tfinal`; default is `NULL`) may be specified. If not specified, the final analysis time `Tfinal` is assumed to be the same as the input variable `T` and will be output as such. `T` is always the study time (time since beginning of enrollment) we are considering for the expected number of events. The minimum follow-up at an interim analysis may be 0 or, if the interim timing is close enough to the final, `minfup - (Tfinal - T)`. The routine will stop accrual according to the minimum of `Tfinal - minfup` and the sum of values in `R`; this is done without changing the input values of either on return from the routine. As a simple example, we consider the following modification of the above.

```
eEvents(
  lambda = c(.05, .02, .01), eta = .01, gamma = c(5, 10, 20),
  R = c(2, 1, 20), S = c(1, 1), T = 18, Tfinal = 22, minfup = 6
)
```

```
  Study duration:              Tfinal=22
  Analysis time:                    T=18
  Accrual duration:                   16
  Min. end-of-study follow-up: minfup=6
  Expected events (total):            35.2387
  Expected sample size (total):       280
  Number of strata:                   1
  Accrual rates:
       Stratum 1
  0-2           5
  2-3          10
  3-16         20
  Event rates:
       Stratum 1
  0-1       0.05
  1-2       0.02
  2-18      0.01
  Censoring rates:
       Stratum 1
  0-1       0.01
  1-2       0.01
  2-18      0.01
```

1.2.13 Literature example: stratified with exponential failure, no dropouts

Bernstein and Lagakos [1978] provided FORTRAN code and examples for a stratified sample size, but did not provide for random dropouts and only accounted for exponential failure rates. This is equivalent to $\eta = 0$ for the more general case here. Their example assumed 2 years of accrual, 2 years of follow-up, 3 strata with 40%, 40% and 20% of enrollment, equal randomization between 2 arms (control and experimental), control group failure rates in the three strata of 1, .8 and .5, a hazard ratio of 2/3 (experimental to control). The expected proportion of deaths by treatment group and stratum are noted as .941, .899, .767 (control) and .854, .788, .625 (experimental). To have an expected number of enrollees of 1 in each group, we require an enrollment rate of .5 over the 2 years, or $\gamma = .5$. We calculate these numbers as follows

```
lamC <- c(1, .8, .5)
n <- eEvents(
  lambda = matrix(c(lamC, lamC * 2 / 3), ncol = 6), eta = 0,
  gamma = matrix(.5, ncol = 6), R = 2, T = 4
)
n$d
```

```
[1] 0.9414902 0.8992911 0.7674558 0.8544147 0.7883950 0.6252700
```

Now for each patient enrolled in the study overall, we compute the expected number events per treatment group given the above enrollment proportions and event rates. First, we do this calculations using the above computation. Then, we follow with direct computation from `eEvents()` by setting the rates so there is an expected enrollment of 1 patient.

```
sum(n$d[1:3] * c(.4, .4, .2)) / 2
```

```
[1] 0.4449018
```

```
sum(n$d[4:6] * c(.4, .4, .2)) / 2
```

```
[1] 0.391089
```

```
n <- eEvents(
  lambda = matrix(c(lamC, lamC * 2 / 3), ncol = 6), eta = 0,
  gamma = matrix(.25 * c(.4, .4, .2, .4, .4, .2), ncol = 6), R = 2, T = 4
)
c(sum(n$d[1:3]), sum(n$d[4:6]))
```

```
[1] 0.4449018 0.3910890
```

Chapter 2

Sample size for a fixed design

There are two approaches to sizing a study from the literature that are presented here. Both cases will be developed for cases where time-to-event distribution assumptions include a stratified population, proportional hazards and piecewise-exponential failure rates.

1. The Lachin and Foulkes [1986] approach where enrollment and study duration are fixed while the enrollment rates are changed proportionately to obtain sufficient enrollment and endpoints to ensure the desired power.
2. The Kim and Tsiatis [1990] method where enrollment rates are fixed and duration of enrollment is varied to ensure power.

2.1 Notation

For the remainder of this document we consider two treatment groups: an experimental group with parameters denoted with a subscript E and a control group with parameters denoted with a subscript C. We will work just with piecewise exponential failure rates and piecewise continuous enrollment. The proportion of patients randomized to the experimental group will be denoted by ξ. Recall that we have used $g()$ to denote the enrollment rate over time. We will now assume the enrollment rate in the experimental group is $\xi g()$, while $(1-\xi)g()$ is the enrollment rate in the control group. For piecewise constant enrollment we assumed $g(t) = \gamma_i$ for t in $(t_{i-1}, t_i]$, for i from 1 to m and $0 = t_0 < t_1 ... < t_m$.

We will use matrix to notation to abbeviate many of the assumptions for each treatment group.

2.2 The basic equation

The two methods will be developed in separate functions initially. Then a single function `nSurv()` will be developed that is intended for to be made visible to users from the gsDesign package.

2.3 Asymptotic approximation for power and sample size

We will use the formulation of Lachin and Foulkes [1986] and their equation (2.4) for the asymptotic approximation relating power to sample size for a proportional hazards model. We generalize their results for the case when the null hypothesis does not specify equal hazards. Denote the null hypothesis failure rates for control and experimental treatment groups as λ_{00} and λ_{01}, respectively. Denote the alternate hypothesis rates as λ_{10} and λ_{11}. Further, denote the alternate hypothesis hazard ratio $h_1 = \lambda_{11}/\lambda_{10}$, and the null hypothesis hazard ratio $h_0 = \lambda_{01}/\lambda_{00}$. We let censoring rates be specific to the control (η_0) and experimental (η_1) groups; these values are only implicit in the equations below. Lachin and Foulkes assumed a null hypothesis with no difference between failure rates in the control and experimental rates and test for superiority. That is, $\lambda_{00} = \lambda_{01}$ ($h_0 = 1$) and $\lambda_{10} < \lambda_{01}$ ($h_1 < 1$). They set event rates under the null hypothesis so that the the weighted average event rate is the same under the null and alternate hypotheses:

$$\lambda_{00} = \lambda_{01} = \bar{\lambda} = (1-\xi)\lambda_{10} + \xi\lambda_{11}. \tag{2.1}$$

The apparent intent of this is to equalize the variance for the log hazard ratio under null and alternative hypotheses. While this will not be exactly the case, we will check the expected number of events under the null and alternate hypotheses later to see that the approximation works reasonably well. The Lachin and Foulkes power equation for proportional hazards translates in our notation to:

$$\begin{aligned}\sqrt{N}\ln(HR) = {} & Z_\alpha\sqrt{E\left\{\delta|\bar{\lambda},\eta\right\}^{-1}\left(\xi^{-1}+(1-\xi)^{-1}\right)} \\ & +Z_\beta\sqrt{E\left\{\delta|\lambda_1,\eta\right\}^{-1}\xi^{-1}+E\left\{\delta|\lambda_0,\eta\right\}^{-1}(1-\xi)^{-1}}\end{aligned} \tag{2.2}$$

Lachin and Foulkes did not cover any cases other than equality under the null hypothesis; (*i.e.*, the assumed $h_0 \neq 1$ or, equivalently, $\lambda_{00} \neq \lambda_{01}$). Equation (2.2) generalizes in this case to

$$\sqrt{N}\ln\left(\frac{h_1}{h_0}\right) = Z_\alpha\sqrt{E\left\{\delta|\lambda_{01},\eta_1\right\}^{-1}\xi^{-1} + E\left\{\delta|\lambda_{00},\eta_0\right\}^{-1}(1-\xi)^{-1}} + Z_\beta\sqrt{E\left\{\delta|\lambda_{11},\eta_1\right\}^{-1}\xi^{-1} + E\left\{\delta|\lambda_{10},\eta_0\right\}^{-1}(1-\xi)^{-1}} \quad (2.3)$$

Note that the computations inside square root signs are variance approximations. When multiple strata are considered, we sum the variance approximations across strata; we will assume a common hazard ratio across strata - although it would appear that power for testing for a weighted average of the log hazard ratios across strata might be considered with the same approach. With version 3.x.x we changed the weighting across strata and the piecewise exponential failure windows to inverse-variance weighted sum. [IN DEVELOPMENT!!!]. To apply equation (2.3), we need specific rates under both the null and alternate hypotheses. For superiority above we started with alternate hypothesis rates and computed rates for the null hypothesis. This also seems appropriate when testing for non-inferiority. Since

$$\lambda_{01} = h_0\lambda_{00},$$

the average of the null hypothesis rates weighted by randomization ratio of experimental to control r is

$$\bar{\lambda} = \frac{\lambda_{00} + r\lambda_{01}}{1+r} = \frac{\lambda_{00}(1+h_0 r)}{1+r}.$$

We have the analogous equation under the alternate hypothesis and we will assume the same $\bar{\lambda}$:

$$\bar{\lambda} = \frac{\lambda_{10}(1+h_1 r)}{1+r}.$$

Given the fixed values for h_0, h_1, r and λ_{10} we can solve for

$$\lambda_{00} = \frac{1+h_1 r}{1+h_0 r}\lambda_{10}$$

and

$$\lambda_{10} = h_0\lambda_{00}.$$

2.4 A period naming utility

We will want to make it clear upon return from any sample size computation what enrollment periods and failure rate periods were assumed. A simple period naming function will make this possible. We define the function `nameperiod()` and provide an initial example:

```
# nameperiod function [sinew] ----
nameperiod <- function(R, digits = 2) {
  if (length(R) == 1) {
    return(paste("0-", round(R, digits), sep = ""))
  }
  R0 <- c(0, R[1:(length(R) - 1)])
  return(paste(round(R0, digits), "-", round(R, digits), sep = ""))
}
```

Now we consider enrollment period durations, convert these to cumulative durations and name them.

```
R <- c(1, 2, 3)
gamma <- matrix(c(4, 5, 6), nrow = 3)
rownames(gamma) <- nameperiod(cumsum(R))
gamma
```

```
     [,1]
0-1    4
1-3    5
3-6    6
```

2.5 Fixed enrollment and study duration

Equation (2.3) can be solved directly given the values of $E\{\delta|\lambda,\eta\}$ for the various values of λ and η. That is, these values do not change with proportional increases or decreases in enrollment rates over a fixed duration of time.

2.5.1 LFPWE source

Note that on output, the input enrollment duration `R` is altered to ensure `T - minfup = sum(R)`. This may eliminate some enrollment periods and/or alter the final returned value of `R`. The returned value of `gamma` is also checked so that the length of the returned value matches the length of the returned `R`.

```
# LFPWE function [sinew] ----
LFPWE <- function(alpha = .025, sided = 1, beta = .1,
                  lambdaC = log(2) / 6, hr = .5, hr0 = 1, etaC = 0, etaE = 0,
                  gamma = 1, ratio = 1, R = 18, S = NULL, T = 24, minfup = NULL) {
  # set up parameters
  zalpha <- -stats::qnorm(alpha / sided)
  zbeta <- -stats::qnorm(beta)
```

```
if (is.null(minfup)) minfup <- max(0, T - sum(R))
if (length(R) == 1) {
  R <- T - minfup
} else if (sum(R) != T - minfup) {
  cR <- cumsum(R)
  nR <- length(R)
  if (cR[length(cR)] < T - minfup) {
    cR[length(cR)] <- T - minfup
  } else {
    cR[cR > T - minfup] <- T - minfup
    cR <- unique(cR)
  }
  if (length(cR) > 1) {
    R <- cR - c(0, cR[1:(length(cR) - 1)])
  } else {
    R <- cR
  }
  if (nR != length(R)) {
    if (is.vector(gamma)) {
      gamma <- gamma[1:length(R)]
    } else {
      gamma <- gamma[1:length(R), ]
    }
  }
}
ngamma <- length(R)
if (is.null(S)) {
  nlambda <- 1
} else {
  nlambda <- length(S) + 1
}
Qe <- ratio / (1 + ratio)
Qc <- 1 - Qe

# compute H0 failure rates as average of control, experimental
if (length(ratio) == 1) {
  lambdaC0 <- (1 + hr * ratio) / (1 + hr0 * ratio) * lambdaC
  gammaC <- gamma * Qc
  gammaE <- gamma * Qe
} else {
  lambdaC0 <- lambdaC %*% diag((1 + hr * ratio) / (1 + hr0 * ratio))
  gammaC <- gamma %*% diag(Qc)
  gammaE <- gamma %*% diag(Qe)
}
```

```
  # do computations
  eDC0 <- eEvents(
    lambda = lambdaC0, eta = etaC, gamma = gammaC,
    R = R, S = S, T = T, minfup = minfup
  )$d
  eDE0 <- eEvents(
    lambda = lambdaC0 * hr0, eta = etaE, gamma = gammaE,
    R = R, S = S, T = T, minfup = minfup
  )$d
  eDC <- eEvents(
    lambda = lambdaC, eta = etaC, gamma = gammaC,
    R = R, S = S, T = T, minfup = minfup
  )
  eDE <- eEvents(
    lambda = lambdaC * hr, eta = etaE, gamma = gammaE,
    R = R, S = S, T = T, minfup = minfup
  )

  n <- ((zalpha * sqrt(1 / sum(eDC0) + 1 / sum(eDE0)) +
    zbeta * sqrt(1 / sum(eDC$d) + 1 / sum(eDE$d))
  ) / log(hr / hr0))^2

  mx <- sum(eDC$n + eDE$n)
  rval <- list(
    alpha = alpha, sided = sided, beta = beta, power = 1 - beta,
    lambdaC = lambdaC, etaC = etaC, etaE = etaE, gamma = n * gamma,
    ratio = ratio, R = R, S = S, T = T, minfup = minfup,
    hr = hr, hr0 = hr0, n = n * mx, d = n * sum(eDC$d + eDE$d),
    eDC = eDC$d * n, eDE = eDE$d * n, eDC0 = eDC0 * n, eDE0 = eDE0 * n,
    eNC = eDC$n * n, eNE = eDE$n * n, variable = "Accrual rate"
  )
  class(rval) <- "nSurv"
  return(rval)
}
```

```
# print.nSurv roxy [sinew] ----
#' @rdname nSurv
#' @export
# print.nSurv function [sinew] ----
print.nSurv <- function(x, digits = 4, ...) {
  if (!inherits(x, "nSurv")) {
    stop("Primary argument must have class nSurv")
  }
  x$digits <- digits
  x$sided <- 1
```

```
cat("Fixed design, two-arm trial with time-to-event\n")
cat("outcome (Lachin and Foulkes, 1986).\n")
cat("Solving for: ", x$variable, "\n")
cat("Hazard ratio                  H1/H0=",
  round(x$hr, digits),
  "/", round(x$hr0, digits), "\n",
  sep = ""
)
cat("Study duration:                    T=",
  round(x$T, digits), "\n",
  sep = ""
)
cat("Accrual duration:                    ",
  round(x$T - x$minfup, digits), "\n",
  sep = ""
)
cat("Min. end-of-study follow-up: minfup=",
  round(x$minfup, digits), "\n",
  sep = ""
)
cat("Expected events (total, H1):         ",
  round(x$d, digits), "\n",
  sep = ""
)
cat("Expected sample size (total):        ",
  round(x$n, digits), "\n",
  sep = ""
)
enrollper <- periods(x$S, x$T, x$minfup, x$digits)
cat("Accrual rates:\n")
print(round(x$gamma, digits))
cat("Control event rates (H1):\n")
print(round(x$lambda, digits))
if (max(abs(x$etaC - x$etaE)) == 0) {
  cat("Censoring rates:\n")
  print(round(x$etaC, digits))
} else {
  cat("Control censoring rates:\n")
  print(round(x$etaC, digits))
  cat("Experimental censoring rates:\n")
  print(round(x$etaE, digits))
}
cat("Power:                  100*(1-beta)=",
  round((1 - x$beta) * 100, digits), "%\n",
```

```
    sep = ""
  )
  cat("Type I error (", x$sided,
    "-sided):   100*alpha=",
    100 * x$alpha, "%\n",
    sep = ""
  )
  if (min(x$ratio == 1) == 1) {
    cat("Equal randomization:          ratio=1\n")
  } else {
    cat(
      "Randomization (Exp/Control):  ratio=",
      x$ratio, "\n"
    )
  }
}
```

2.5.2 Experimental version

```
# LFPWEnew function [sinew] ----
LFPWEnew <- function(alpha = .025, sided = 1, beta = .1,
                     lambdaC = log(2) / 6, hr = .5, hr0 = 1, etaC = 0, etaE = 0,
                     gamma = 1, ratio = 1, R = 18, S = NULL, T = 24, minfup = NULL) {
  # set up parameters
  zalpha <- -stats::qnorm(alpha / sided)
  if (is.null(minfup)) minfup <- max(0, T - sum(R))
  if (length(R) == 1) {
    R <- T - minfup
  } else if (sum(R) != T - minfup) {
    cR <- cumsum(R)
    nR <- length(R)
    if (cR[length(cR)] < T - minfup) {
      cR[length(cR)] <- T - minfup
    } else {
      cR[cR > T - minfup] <- T - minfup
      cR <- unique(cR)
    }
    if (length(cR) > 1) {
      R <- cR - c(0, cR[1:(length(cR) - 1)])
    } else {
      R <- cR
    }
```

```
  if (nR != length(R)) {
    if (is.vector(gamma)) {
      gamma <- gamma[1:length(R)]
    } else {
      gamma <- gamma[1:length(R), ]
    }
  }
}
ngamma <- length(R)
if (is.null(S)) {
  nlambda <- 1
} else {
  nlambda <- length(S) + 1
}
Qe <- ratio / (1 + ratio)
Qc <- 1 - Qe

# compute H0 failure rates as average of control, experimental
if (length(ratio) == 1) {
  lambdaC0 <- (1 + hr * ratio) / (1 + hr0 * ratio) * lambdaC
  gammaC <- gamma * Qc
  gammaE <- gamma * Qe
} else {
  lambdaC0 <- lambdaC %*% diag((1 + hr * ratio) / (1 + hr0 * ratio))
  gammaC <- gamma %*% diag(Qc)
  gammaE <- gamma %*% diag(Qe)
}
# do computations
eDC0 <- eEvents(
  lambda = lambdaC0, eta = etaC, gamma = gammaC,
  R = R, S = S, T = T, minfup = minfup
)$d
eDE0 <- eEvents(
  lambda = lambdaC0 * hr0, eta = etaE, gamma = gammaE,
  R = R, S = S, T = T, minfup = minfup
)$d
eDC <- eEvents(
  lambda = lambdaC, eta = etaC, gamma = gammaC,
  R = R, S = S, T = T, minfup = minfup
)
eDE <- eEvents(
  lambda = lambdaC * hr, eta = etaE, gamma = gammaE,
  R = R, S = S, T = T, minfup = minfup
)
```

```
  # sample size
  if (!is.null(beta)) {
    zbeta <- -stats::qnorm(beta)
    ##### OLD
    n_old <- ((zalpha * sqrt(1 / sum(eDC0) + 1 / sum(eDE0)) +
      zbeta * sqrt(1 / sum(eDC$d) + 1 / sum(eDE$d))
    ) / log(hr / hr0))^2
    cat("Control events (H0): ")
    cat(eDC0)
    cat("\nControl events (H1): ")
    cat(eDC$d)
    cat("\nExperimental events (H0): ")
    cat(eDE0)
    cat("\nExperimental events (H1): ")
    cat(eDE$d)
    cat("\nOld n: ")
    cat(n_old)
    cat("\n")
    #####
    ##### NEW 20230327; NO CHANGE FOR UNSTRATIFIED
    n <- ((zalpha * sqrt(1 / sum((1 / eDC0 + 1 / eDE0)^(-1))) +
      zbeta * sqrt(1 / sum((1 / eDC$d + 1 / eDE$d)^(-1)))
    ) / log(hr / hr0))^2
    cat("New n: ")
    cat(n)
    cat("\n")
  } else {
    # power
  }
  mx <- sum(eDC$n + eDE$n)
  rval <- list(
    alpha = alpha, sided = sided, beta = beta, power = 1 - beta,
    lambdaC = lambdaC, etaC = etaC, etaE = etaE, gamma = n * gamma,
    ratio = ratio, R = R, S = S, T = T, minfup = minfup,
    hr = hr, hr0 = hr0, n = n * mx, d = n * sum(eDC$d + eDE$d),
    eDC = eDC$d * n, eDE = eDE$d * n, eDC0 = eDC0 * n, eDE0 = eDE0 * n,
    eNC = eDC$n * n, eNE = eDE$n * n, variable = "Accrual rate"
  )
  class(rval) <- "nSurv"
  return(rval)
}
```

2.5.3 Examples

In the Merck (ELSTIC) guidance on time-to-event sample size, an example with $\alpha = .025$, $\beta = .1$, $\lambda_C = .2$, $\lambda_E = .1$, $\eta_C = \eta_E = .1$, enrollment period of 1/2 year and total study time of 2 years. The Lachin and Foulkes [1986] method is documented to produce a sample size of 430, which is the same as the following if rounded up to the nearest even number. This is reproduced here by both the `nSurvival()` routine, which has been validated, and by `LFPWE()`.

```
LFPWE(
  alpha = .025, sided = 1, beta = .1, lambdaC = .2, hr = 1 / 2,
  etaC = .1, etaE = .1, gamma = 1, ratio = 1, R = .5, S = NULL, T = 2
)
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual rate
  Hazard ratio                  H1/H0=0.5/1
  Study duration:                   T=2
  Accrual duration:                   0.5
  Min. end-of-study follow-up: minfup=1.5
  Expected events (total, H1):        90.0987
  Expected sample size (total):       429.6189
  Accrual rates:
  [1] 859.2377
  Control event rates (H1):
  [1] 0.2
  Censoring rates:
  [1] 0.1
  Power:                 100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:          ratio=1
```

```
x <- gsDesign:::nSurvival(lambda1 = .2, lambda2 = .1, eta = .1, Tr = .5, Ts = 2)
x$n
```

```
  [1] 429.6189
```

```
x$nEvents
```

```
  [1] 90.09875
```

```
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Study duration (fixed):          Ts=2
```

```
  Accrual duration (fixed):        Tr=0.5
  Uniform accrual:               entry="unif"
  Control median:      log(2)/lambda1=3.5
  Experimental median: log(2)/lambda2=6.9
  Censoring median:        log(2)/eta=6.9
  Control failure rate:       lambda1=0.2
  Experimental failure rate:  lambda2=0.1
  Censoring rate:                eta=0.1
  Power:                 100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:          ratio=1
  Sample size based on hazard ratio=0.5 (type="rr")
  Sample size (computed):           n=430
  Events required (computed): nEvents=91
```

Same example with new version:

```
LFPWEnew(
  alpha = .025, sided = 1, beta = .1, lambdaC = .2, hr = 1 / 2,
  etaC = .1, etaE = .1, gamma = 1, ratio = 1, R = .5, S = NULL, T = 2
)
```

We return to the Bernstein and Lagakos [1978] example of a stratified trial with exponential failure times. Their methods incorporate only a variance estimate under the alternate hypothesis and yields a sample size of 172, slightly smaller than what we compute here.

```
x <- LFPWE(
  alpha = .05, beta = .2, hr = 2 / 3, ratio = 1,
  R = 2, S = NULL, T = 4,
  lambdaC = matrix(c(1, .8, .5), nrow = 1),
  etaC = matrix(0, nrow = 1, ncol = 3),
  etaE = matrix(0, nrow = 1, ncol = 3),
  gamma = matrix(c(.4, .4, .2), nrow = 1)
)
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual rate
  Hazard ratio                  H1/H0=0.6667/1
  Study duration:                   T=4
  Accrual duration:                   2
  Min. end-of-study follow-up: minfup=2
  Expected events (total, H1):        149.455
  Expected sample size (total):       178.7758
  Accrual rates:
```

```
        [,1]    [,2]    [,3]
[1,] 35.7552 35.7552 17.8776
Control event rates (H1):
     [,1] [,2] [,3]
[1,]    1  0.8  0.5
Censoring rates:
     [,1] [,2] [,3]
[1,]    0    0    0
Power:                  100*(1-beta)=80%
Type I error (1-sided):   100*alpha=5%
Equal randomization:          ratio=1
```

Next, we consider the piecewise exponential case

```
x <- LFPWE(
  alpha = .025, beta = .1,
  lambdaC = matrix(c(.05, .02, .01), ncol = 1), hr = .6,
  etaC = matrix(.01, nrow = 3, ncol = 1), etaE = matrix(.01, nrow = 3, ncol = 1),
  gamma = matrix(c(5, 10, 20), ncol = 1),
  R = c(2, 1, 2), S = c(1, 1), T = 20
)
x
```

```
Fixed design, two-arm trial with time-to-event
outcome (Lachin and Foulkes, 1986).
Solving for:  Accrual rate
Hazard ratio                  H1/H0=0.6/1
Study duration:                   T=20
Accrual duration:                   5
Min. end-of-study follow-up: minfup=15
Expected events (total, H1):        164.1408
Expected sample size (total):       1099.533
Accrual rates:
         [,1]
[1,]  91.6277
[2,] 183.2555
[3,] 366.5109
Control event rates (H1):
     [,1]
[1,] 0.05
[2,] 0.02
[3,] 0.01
Censoring rates:
     [,1]
[1,] 0.01
[2,] 0.01
```

```
  [3,] 0.01
  Power:                    100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:           ratio=1
```

Finally, we have all of: piecewise exponential failure, piecewise constant enrollment, and a stratified population with differing randomization ratio by stratum.

```
x <- LFPWE(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2),
  hr = .6, S = c(3, 6), ratio = 1:3, R = c(3, 3), minfup = 6, T = 30
)
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual rate
  Hazard ratio                    H1/H0=0.6/1
  Study duration:                     T=30
  Accrual duration:                     24
  Min. end-of-study follow-up: minfup=6
  Expected events (total, H1):        171.4585
  Expected sample size (total):       287.077
  Accrual rates:
         [,1]   [,2]   [,3]
  [1,] 1.4177 5.6707 4.2530
  [2,] 2.8353 2.1265 7.0883
  Control event rates (H1):
         [,1]   [,2]   [,3]
  [1,] 0.2310 0.1155 0.0770
  [2,] 0.1733 0.0866 0.0578
  [3,] 0.1386 0.0693 0.0462
  Censoring rates:
  [1] 0
  Power:                    100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Randomization (Exp/Control):  ratio= 1 2 3
```

2.6 Fixed enrollment rate methods

Now we assume fixed enrollment rates and adjust either the enrollment duration or follow-up duration to obtain a targeted power or number of events. As

this is following the strategy of Kim and Tsiatis [1990], we name the function `KT()`. `KT()` is primarily a root-finding program based on the routine `KTZ()` which is used to calculate power based on (eq:npwr) or the expected number of events under the alternate hypothesis (using `eEvents()`) given enrollment rates, duration of a study and duration of follow-up.

2.6.1 KTZ source code and discussion

`KTZ()` can compute the difference from a targeted number of events or targeted power. It can also compute power for a trial with given design characteristics. Depending on the structure of other input variables, a non-NULL input value of `x` may be used to set

- the minimum follow-up after a given enrollment duration (if `minfup` is input as `NULL`), or
- the duration of enrollment.

If `simple` is input as `TRUE`, then the routine is used to return a difference from a target value for

- a targeted total number of events in `n1Target` if `n1Target` is not `NULL`, or
- a targeted power (target is actually set to the standard normal inverse of `1 - beta`), if `beta` if both `n1Target` and `beta` are `NULL` on input.

If `simple` is input as `FALSE`, then an `nSurv` object is returned giving the power and type II error for the input parameter values for the trial.

```
# KTZ function [sinew] ----
KTZ <- function(x = NULL, minfup = NULL, n1Target = NULL,
                lambdaC = log(2) / 6, etaC = 0, etaE = 0,
                gamma = 1, ratio = 1, R = 18, S = NULL, beta = .1,
                alpha = .025, sided = 1, hr0 = 1, hr = .5, simple = TRUE) {
  zalpha <- -stats::qnorm(alpha / sided)
  Qc <- 1 / (1 + ratio)
  Qe <- 1 - Qc
  # set minimum follow-up to x if that is missing and x is given
  if (!is.null(x) && is.null(minfup)) {
    minfup <- x
    if (sum(R) == Inf) {
      stop("If minimum follow-up is sought, enrollment duration must be finite")
    }
    T <- sum(R) + minfup
    variable <- "Follow-up duration"
  } else if (!is.null(x) && !is.null(minfup)) { # otherwise, if x is given, set it to accru
    T <- x + minfup
```

```
  R[length(R)] <- Inf
  variable <- "Accrual duration"
} else { # otherwise, set follow-up time to accrual plus follow-up
  T <- sum(R) + minfup
  variable <- "Power"
}
# compute H0 failure rates as average of control, experimental
if (length(ratio) == 1) {
  lambdaC0 <- (1 + hr * ratio) / (1 + hr0 * ratio) * lambdaC
  gammaC <- gamma * Qc
  gammaE <- gamma * Qe
} else {
  lambdaC0 <- lambdaC %*% diag((1 + hr * ratio) / (1 + hr0 * ratio))
  gammaC <- gamma %*% diag(Qc)
  gammaE <- gamma %*% diag(Qe)
}

# do computations
eDC <- eEvents(
  lambda = lambdaC, eta = etaC, gamma = gammaC,
  R = R, S = S, T = T, minfup = minfup
)
eDE <- eEvents(
  lambda = lambdaC * hr, eta = etaE, gamma = gammaE,
  R = R, S = S, T = T, minfup = minfup
)
# if this is all that is needed, return difference
# from targeted number of events
if (simple && !is.null(n1Target)) {
  return(sum(eDC$d + eDE$d) - n1Target)
}
eDC0 <- eEvents(
  lambda = lambdaC0, eta = etaC, gamma = gammaC,
  R = R, S = S, T = T, minfup = minfup
)
eDE0 <- eEvents(
  lambda = lambdaC0 * hr0, eta = etaE, gamma = gammaE,
  R = R, S = S, T = T, minfup = minfup
)
# compute Z-value related to power from power equation
zb <- (log(hr0 / hr) -
  zalpha * sqrt(1 / sum(eDC0$d) + 1 / sum(eDE0$d))) /
  sqrt(1 / sum(eDC$d) + 1 / sum(eDE$d))
# if that is all that is needed, return difference from
```

```
  # targeted value
  if (simple) {
    if (!is.null(beta)) {
      return(zb + stats::qnorm(beta))
    } else {
      return(stats::pnorm(-zb))
    }
  }
  # compute power
  power <- stats::pnorm(zb)
  beta <- 1 - power
  # set accrual period durations
  if (sum(R) != T - minfup) {
    if (length(R) == 1) {
      R <- T - minfup
    } else {
      nR <- length(R)
      cR <- cumsum(R)
      cR[cR > T - minfup] <- T - minfup
      cR <- unique(cR)
      cR[length(R)] <- T - minfup
      if (length(cR) == 1) {
        R <- cR
      } else {
        R <- cR - c(0, cR[1:(length(cR) - 1)])
      }
      if (length(R) != nR) {
        gamma <- matrix(gamma[1:length(R), ], nrow = length(R))
        gdim <- dim(gamma)
      }
    }
  }
  rval <- list(
    alpha = alpha, sided = sided, beta = beta, power = power,
    lambdaC = lambdaC, etaC = etaC, etaE = etaE,
    gamma = gamma, ratio = ratio, R = R, S = S, T = T,
    minfup = minfup, hr = hr, hr0 = hr0, n = sum(eDC$n + eDE$n),
    d = sum(eDC$d + eDE$d), tol = NULL, eDC = eDC$d, eDE = eDE$d,
    eDC0 = eDC0$d, eDE0 = eDE0$d, eNC = eDC$n, eNE = eDE$n,
    variable = variable
  )
  class(rval) <- "nSurv"
  return(rval)
}
```

```
# input minimum follow-up in x and consider R fixed
KTZ(
  x = 2, lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2), n1Target = 0,
  hr = .6, S = c(3, 6), R = c(3, 15), minfup = NULL
)
```

```
[1] 148.8083
```

```
# set total time to x + minfup (R is variable)
KTZ(
  x = 10, lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2), n1Target = 0,
  hr = .6, S = c(3, 6), R = c(3, 15), minfup = 2
)
```

```
[1] 65.46827
```

```
# same as 2nd example with differing randomization ratio by stratum
KTZ(
  x = 10, lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2), n1Target = 0,
  hr = .6, S = c(3, 6), ratio = 1:3, R = c(3, 15), minfup = 2
)
```

```
[1] 61.67763
```

2.6.2 KT source code

```
# KT function [sinew] ----
KT <- function(alpha = .025, sided = 1, beta = .1,
               lambdaC = log(2) / 6, hr = .5, hr0 = 1, etaC = 0, etaE = 0,
               gamma = 1, ratio = 1, R = 18, S = NULL, minfup = NULL,
               n1Target = NULL, tol = .Machine$double.eps^0.25) {
  # set up parameters
  ngamma <- length(R)
  if (is.null(S)) {
    nlambda <- 1
  } else {
    nlambda <- length(S) + 1
  }
  Qe <- ratio / (1 + ratio)
  Qc <- 1 - Qe
  if (!is.matrix(lambdaC)) lambdaC <- matrix(lambdaC)
```

```
  ldim <- dim(lambdaC)
  nstrata <- ldim[2]
  nlambda <- ldim[1]
  etaC <- matrix(etaC, nrow = nlambda, ncol = nstrata)
  etaE <- matrix(etaE, nrow = nlambda, ncol = nstrata)
  if (!is.matrix(gamma)) gamma <- matrix(gamma)
  gdim <- dim(gamma)
  eCdim <- dim(etaC)
  eEdim <- dim(etaE)

  # search for trial duration needed to achieve desired power
  if (is.null(minfup)) {
    if (sum(R) == Inf) {
      stop("Enrollment duration must be specified as finite")
    }
    left <- KTZ(.01,
      lambdaC = lambdaC, n1Target = n1Target,
      etaC = etaC, etaE = etaE, gamma = gamma, ratio = ratio,
      R = R, S = S, beta = beta, alpha = alpha, sided = sided,
      hr0 = hr0, hr = hr
    )
    right <- KTZ(1000,
      lambdaC = lambdaC, n1Target = n1Target,
      etaC = etaC, etaE = etaE, gamma = gamma, ratio = ratio,
      R = R, S = S, beta = beta, alpha = alpha, sided = sided,
      hr0 = hr0, hr = hr
    )
    if (left > 0) stop("Enrollment duration over-powers trial")
    if (right < 0) stop("Enrollment duration insufficient to power trial")
    y <- stats::uniroot(
      f = KTZ, interval = c(.01, 10000), lambdaC = lambdaC,
      etaC = etaC, etaE = etaE, gamma = gamma, ratio = ratio,
      R = R, S = S, beta = beta, alpha = alpha, sided = sided,
      hr0 = hr0, hr = hr, tol = tol, n1Target = n1Target
    )
    minfup <- y$root
    xx <- KTZ(
      x = y$root, lambdaC = lambdaC,
      etaC = etaC, etaE = etaE, gamma = gamma, ratio = ratio,
      R = R, S = S, minfup = NULL, beta = beta, alpha = alpha,
      sided = sided, hr0 = hr0, hr = hr, simple = F
    )
    xx$tol <- tol
    return(xx)
```

```
  } else {
    y <- stats::uniroot(
      f = KTZ, interval = minfup + c(.01, 10000), lambdaC = lambdaC,
      n1Target = n1Target,
      etaC = etaC, etaE = etaE, gamma = gamma, ratio = ratio,
      R = R, S = S, minfup = minfup, beta = beta,
      alpha = alpha, sided = sided, hr0 = hr0, hr = hr, tol = tol
    )
    xx <- KTZ(
      x = y$root, lambdaC = lambdaC,
      etaC = etaC, etaE = etaE, gamma = gamma, ratio = ratio,
      R = R, S = S, minfup = minfup, beta = beta, alpha = alpha,
      sided = sided, hr0 = hr0, hr = hr, simple = F
    )
    xx$tol <- tol
    return(xx)
  }
}
```

```
# set minimum follow-up (allow R to vary to get power)
KT(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2), n1Target = 10,
  hr = .6, S = c(3, 6), ratio = 1:3, R = c(3, 15), minfup = 2
)
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual duration
  Hazard ratio                    H1/H0=0.6/1
  Study duration:                     T=4.6979
  Accrual duration:                     2.6979
  Min. end-of-study follow-up: minfup=2
  Expected events (total, H1):        10
  Expected sample size (total):       43.1662

  Warning in max(periods): no non-missing arguments to max; returning -Inf

  Accrual rates:
       [,1] [,2] [,3]
  [1,]    2    8    6
  [2,]    4    3   10
  Control event rates (H1):
         [,1]   [,2]   [,3]
  [1,] 0.2310 0.1155 0.0770
  [2,] 0.1733 0.0866 0.0578
```

```
  [3,] 0.1386 0.0693 0.0462
  Censoring rates:
       [,1] [,2] [,3]
  [1,]    0    0    0
  [2,]    0    0    0
  [3,]    0    0    0
  Power:                  100*(1-beta)=10.9406%
  Type I error (1-sided):   100*alpha=2.5%
  Randomization (Exp/Control):  ratio= 1 2 3
```

```
# allow minimum follow-up to vary to obtain power
KT(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2), n1Target = 100,
  hr = .6, S = c(3, 6), ratio = 1:3, R = c(3, 10), minfup = NULL
)
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Follow-up duration
  Hazard ratio                   H1/H0=0.6/1
  Study duration:                    T=16.7158
  Accrual duration:                    13
  Min. end-of-study follow-up: minfup=3.7158
  Expected events (total, H1):         100.0001
  Expected sample size (total):        218
  Accrual rates:
       [,1] [,2] [,3]
  [1,]    2    8    6
  [2,]    4    3   10
  Control event rates (H1):
         [,1]   [,2]   [,3]
  [1,] 0.2310 0.1155 0.0770
  [2,] 0.1733 0.0866 0.0578
  [3,] 0.1386 0.0693 0.0462
  Censoring rates:
       [,1] [,2] [,3]
  [1,]    0    0    0
  [2,]    0    0    0
  [3,]    0    0    0
  Power:                  100*(1-beta)=69.7639%
  Type I error (1-sided):   100*alpha=2.5%
  Randomization (Exp/Control):  ratio= 1 2 3
```

2.6.3 Checks

We will check same sample size and power calculations using all different methods we have using a common example. We do not print the output of each check due to volume, but allow the reader the option of running the code to make comparisons. Note that `nSurvival()` which was validated versus external results in a previous release of the gsDesign package and that `LFPWE()` was previously checked versus `nSurvival()`. Thus, we begin by deriving a sample size using `LFPWE()`. The input parameters are specifically kept separate at the beginning of the code so that the user can easily check using whatever parameters interest them. Note that `nSurvival()` will only work with a single stratum and failure rate period.

```
lambdaC <- log(2) / 20
hr <- .5
hr0 <- 1
S <- NULL
T <- 30
R <- 20
minfup <- 10
gamma <- 8
alpha <- .025
sided <- 1
alpha <- alpha
beta <- .1
etaC <- 0
etaE <- 0
ratio <- 1

gsDesign::nSurvival(
  lambda1 = lambdaC, lambda2 = lambdaC * hr, Ts = T,
  Tr = R, eta = etaC, alpha = alpha, sided = sided, beta = beta
)
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Study duration (fixed):          Ts=30
  Accrual duration (fixed):        Tr=20
  Uniform accrual:              entry="unif"
  Control median:      log(2)/lambda1=20
  Experimental median: log(2)/lambda2=40
  Censoring only at study end (eta=0)
  Control failure rate:       lambda1=0.035
  Experimental failure rate:  lambda2=0.017
  Censoring rate:                eta=0
```

```
  Power:                  100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:           ratio=1
  Sample size based on hazard ratio=0.5 (type="rr")
  Sample size (computed):            n=228
  Events required (computed): nEvents=89
```

```
y <- LFPWE(
  gamma = gamma, hr = hr, lambdaC = lambdaC, R = R, T = T,
  minfup = minfup, alpha = alpha, sided = sided, beta = beta
)
```

Now we move on to derive the above result using `KTZ()` which computes power. With `simple = TRUE` (the default) and `beta = NULL`, the routine will should return the power to match the desired 90% input above. The first call derives `R` based on the input duration of accrual input in `x`. The second call sets the minimum follow-up duration to `minfup` to the input value of `x`.

```
KTZ(
  x = y$T - y$minfup, gamma = y$gamma, hr = y$hr,
  lambdaC = y$lambdaC, R = Inf, minfup = y$minfup
)
KTZ(
  x = y$minfup, gamma = y$gamma, hr = y$hr, lambdaC = y$lambdaC,
  minfup = NULL, R = y$T - y$minfup
)
```

Now we move on to deriving the sample size using `KT()`. The first call derives the duration of enrollment required and the second call the minimum follow-up required to obtain the desired power.

```
if (length(R) == 1) {
  temR <- Inf
} else {
  temR <- R
  temR[length(R)] <- Inf
}
KT(gamma = y, hr = hr, lambdaC = lambdaC, R = temR, minfup = minfup)
KT(gamma = y$gamma, hr = hr, lambdaC = lambdaC, R = R)
```

2.6.4 Examples

Using the output design as input, we compute the power both for hazard ratios of .6 and .75:

```
x <- KTZ(
  gamma = y$gamma, hr = .6, lambdaC = y$lambdaC,
  minfup = y$minfup, R = y$T - y$minfup, simple = F
)
x$power
```

```
[1] 0.69822
```

```
x <- KTZ(
  gamma = y$gamma, hr = .75, lambdaC = y$lambdaC,
  minfup = y$minfup, R = y$T - y$minfup, simple = F
)
x$power
```

```
[1] 0.3063416
```

2.7 nSurv: the general fixed design sample size and power function

The code for the general sample size and power routine `nSurv()` is short as it simply calls routines derived above. Following the code, we document the input and output for the routine.

2.7.1 Source code

```
# nSurv roxy [sinew] ----
#' Advanced time-to-event sample size calculation
#'
#' \code{nSurv()} is used to calculate the sample size for a clinical trial
#' with a time-to-event endpoint and an assumption of proportional hazards.
#' This set of routines is new with version 2.7 and will continue to be
#' modified and refined to improve input error checking and output format with
#' subsequent versions. It allows both the Lachin and Foulkes (1986) method
#' (fixed trial duration) as well as the Kim and Tsiatis(1990) method (fixed
#' enrollment rates and either fixed enrollment duration or fixed minimum
#' follow-up). Piecewise exponential survival is supported as well as piecewise
#' constant enrollment and dropout rates. The methods are for a 2-arm trial
#' with treatment groups referred to as experimental and control. A stratified
#' population is allowed as in Lachin and Foulkes (1986); this method has been
#' extended to derive non-inferiority as well as superiority trials.
#' Stratification also allows power calculation for meta-analyses.
```

```
#' \code{gsSurv()} combines \code{nSurv()} with \code{gsDesign()} to derive a
#' group sequential design for a study with a time-to-event endpoint.
#'
#' \code{print()}, \code{xtable()} and \code{summary()} methods are provided to
#' operate on the returned value from \code{gsSurv()}, an object of class
#' \code{gsSurv}. \code{print()} is also extended to \code{nSurv} objects. The
#' functions \code{\link{gsBoundSummary}} (data frame for tabular output),
#' \code{\link{xprint}} (application of \code{xtable} for tabular output) and
#' \code{summary.gsSurv} (textual summary of \code{gsDesign} or \code{gsSurv}
#' object) may be preferred summary functions; see example in vignettes. See
#' also \link{gsBoundSummary} for output
#' of tabular summaries of bounds for designs produced by \code{gsSurv()}.
#'
#' Both \code{nEventsIA} and \code{tEventsIA} require a group sequential design
#' for a time-to-event endpoint of class \code{gsSurv} as input.
#' \code{nEventsIA} calculates the expected number of events under the
#' alternate hypothesis at a given interim time. \code{tEventsIA} calculates
#' the time that the expected number of events under the alternate hypothesis
#' is a given proportion of the total events planned for the final analysis.
#'
#' \code{nSurv()} produces an object of class \code{nSurv} with the number of
#' subjects and events for a set of pre-specified trial parameters, such as
#' accrual duration and follow-up period. The underlying power calculation is
#' based on Lachin and Foulkes (1986) method for proportional hazards assuming
#' a fixed underlying hazard ratio between 2 treatment groups. The method has
#' been extended here to enable designs to test non-inferiority. Piecewise
#' constant enrollment and failure rates are assumed and a stratified
#' population is allowed. See also \code{\link{nSurvival}} for other Lachin and
#' Foulkes (1986) methods assuming a constant hazard difference or exponential
#' enrollment rate.
#'
#' When study duration (\code{T}) and follow-up duration (\code{minfup}) are
#' fixed, \code{nSurv} applies exactly the Lachin and Foulkes (1986) method of
#' computing sample size under the proportional hazards assumption when For
#' this computation, enrollment rates are altered proportionately to those
#' input in \code{gamma} to achieve the power of interest.
#'
#' Given the specified enrollment rate(s) input in \code{gamma}, \code{nSurv}
#' may also be used to derive enrollment duration required for a trial to have
#' defined power if \code{T} is input as \code{NULL}; in this case, both
#' \code{R} (enrollment duration for each specified enrollment rate) and
#' \code{T} (study duration) will be computed on output.
#'
#' Alternatively and also using the fixed enrollment rate(s) in \code{gamma},
```

```
#' if minimum follow-up \code{minfup} is specified as \code{NULL}, then the
#' enrollment duration(s) specified in \code{R} are considered fixed and
#' \code{minfup} and \code{T} are computed to derive the desired power. This
#' method will fail if the specified enrollment rates and durations either
#' over-powers the trial with no additional follow-up or underpowers the trial
#' with infinite follow-up. This method produces a corresponding error message
#' in such cases.
#'
#' The input to \code{gsSurv} is a combination of the input to \code{nSurv()}
#' and \code{gsDesign()}.
#'
#' \code{nEventsIA()} is provided to compute the expected number of events at a
#' given point in time given enrollment, event and censoring rates. The routine
#' is used with a root finding routine to approximate the approximate timing of
#' an interim analysis. It is also used to extend enrollment or follow-up of a
#' fixed design to obtain a sufficient number of events to power a group
#' sequential design.
#'
#' @aliases nSurv print.nSurv print.gsSurv xtable.gsSurv
#' @param x An object of class \code{nSurv} or \code{gsSurv}.
#' \code{print.nSurv()} is used for an object of class \code{nSurv} which will
#' generally be output from \code{nSurv()}. For \code{print.gsSurv()} is used
#' for an object of class \code{gsSurv} which will generally be output from
#' \code{gsSurv()}. \code{nEventsIA} and \code{tEventsIA} operate on both the
#' \code{nSurv} and \code{gsSurv} class.
#' @param digits Number of digits past the decimal place to print
#' (\code{print.gsSurv.}); also a pass through to generic \code{xtable()} from
#' \code{xtable.gsSurv()}.
#' @param lambdaC scalar, vector or matrix of event hazard rates for the
#' control group; rows represent time periods while columns represent strata; a
#' vector implies a single stratum.
#' @param hr hazard ratio (experimental/control) under the alternate hypothesis
#' (scalar).
#' @param hr0 hazard ratio (experimental/control) under the null hypothesis
#' (scalar).
#' @param eta scalar, vector or matrix of dropout hazard rates for the control
#' group; rows represent time periods while columns represent strata; if
#' entered as a scalar, rate is constant across strata and time periods; if
#' entered as a vector, rates are constant across strata.
#' @param etaE matrix dropout hazard rates for the experimental group specified
#' in like form as \code{eta}; if NULL, this is set equal to \code{eta}.
#' @param gamma a scalar, vector or matrix of rates of entry by time period
#' (rows) and strata (columns); if entered as a scalar, rate is constant
#' across strata and time periods; if entered as a vector, rates are constant
```

```
#' across strata.
#' @param R a scalar or vector of durations of time periods for recruitment
#' rates specified in rows of \code{gamma}. Length is the same as number of
#' rows in \code{gamma}. Note that when variable enrollment duration is
#' specified (input \code{T=NULL}), the final enrollment period is extended as
#' long as needed.
#' @param S a scalar or vector of durations of piecewise constant event rates
#' specified in rows of \code{lambda}, \code{eta} and \code{etaE}; this is NULL
#' if there is a single event rate per stratum (exponential failure) or length
#' of the number of rows in \code{lambda} minus 1, otherwise.
#' @param T study duration; if \code{T} is input as \code{NULL}, this will be
#' computed on output; see details.
#' @param minfup follow-up of last patient enrolled; if \code{minfup} is input
#' as \code{NULL}, this will be computed on output; see details.
#' @param ratio randomization ratio of experimental treatment divided by
#' control; normally a scalar, but may be a vector with length equal to number
#' of strata.
#' @param sided 1 for 1-sided testing, 2 for 2-sided testing.
#' @param alpha type I error rate. Default is 0.025 since 1-sided testing is
#' default.
#' @param beta type II error rate. Default is 0.10 (90\% power); NULL if power
#' is to be computed based on other input values.
#' @param tol for cases when \code{T} or \code{minfup} values are derived
#' through root finding (\code{T} or \code{minfup} input as \code{NULL}),
#' \code{tol} provides the level of error input to the \code{uniroot()}
#' root-finding function. The default is the same as for \code{\link{uniroot}}.
#' @param k Number of analyses planned, including interim and final.
#' @param test.type \code{1=}one-sided \cr \code{2=}two-sided symmetric \cr
#' \code{3=}two-sided, asymmetric, beta-spending with binding lower bound \cr
#' \code{4=}two-sided, asymmetric, beta-spending with non-binding lower bound
#' \cr \code{5=}two-sided, asymmetric, lower bound spending under the null
#' hypothesis with binding lower bound \cr \code{6=}two-sided, asymmetric,
#' lower bound spending under the null hypothesis with non-binding lower bound.
#' \cr See details, examples and manual.
#' @param astar Normally not specified. If \code{test.type=5} or \code{6},
#' \code{astar} specifies the total probability of crossing a lower bound at
#' all analyses combined.  This will be changed to \eqn{1 - }\code{alpha} when
#' default value of 0 is used.  Since this is the expected usage, normally
#' \code{astar} is not specified by the user.
#' @param timing Sets relative timing of interim analyses in \code{gsSurv}.
#' Default of 1 produces equally spaced analyses.  Otherwise, this is a vector
#' of length \code{k} or \code{k-1}.  The values should satisfy \code{0 <
#' timing[1] < timing[2] < ... < timing[k-1] < timing[k]=1}. For
#' \code{tEventsIA}, this is a scalar strictly between 0 and 1 that indicates
```

```
#' the targeted proportion of final planned events available at an interim
#' analysis.
#' @param sfu A spending function or a character string indicating a boundary
#' type (that is, \dQuote{WT} for Wang-Tsiatis bounds, \dQuote{OF} for
#' O'Brien-Fleming bounds and \dQuote{Pocock} for Pocock bounds).  For
#' one-sided and symmetric two-sided testing is used to completely specify
#' spending (\code{test.type=1, 2}), \code{sfu}.  The default value is
#' \code{sfHSD} which is a Hwang-Shih-DeCani spending function.  See details,
#' \link{Spending_Function_Overview}, manual and examples.
#' @param sfupar Real value, default is \eqn{-4} which is an
#' O'Brien-Fleming-like conservative bound when used with the default
#' Hwang-Shih-DeCani spending function. This is a real-vector for many spending
#' functions.  The parameter \code{sfupar} specifies any parameters needed for
#' the spending function specified by \code{sfu}; this will be ignored for
#' spending functions (\code{sfLDOF}, \code{sfLDPocock}) or bound types
#' (\dQuote{OF}, \dQuote{Pocock}) that do not require parameters.
#' @param sfl Specifies the spending function for lower boundary crossing
#' probabilities when asymmetric, two-sided testing is performed
#' (\code{test.type = 3}, \code{4}, \code{5}, or \code{6}).  Unlike the upper
#' bound, only spending functions are used to specify the lower bound.  The
#' default value is \code{sfHSD} which is a Hwang-Shih-DeCani spending
#' function.  The parameter \code{sfl} is ignored for one-sided testing
#' (\code{test.type=1}) or symmetric 2-sided testing (\code{test.type=2}).  See
#' details, spending functions, manual and examples.
#' @param sflpar Real value, default is \eqn{-2}, which, with the default
#' Hwang-Shih-DeCani spending function, specifies a less conservative spending
#' rate than the default for the upper bound.
#' @param r Integer value controlling grid for numerical integration as in
#' Jennison and Turnbull (2000); default is 18, range is 1 to 80. Larger values
#' provide larger number of grid points and greater accuracy.  Normally
#' \code{r} will not be changed by the user.
#' @param usTime Default is NULL in which case upper bound spending time is
#' determined by \code{timing}. Otherwise, this should be a vector of length
#' code{k} with the spending time at each analysis (see Details in help for \code{gsDesign}
#' @param lsTime Default is NULL in which case lower bound spending time is
#' determined by \code{timing}. Otherwise, this should be a vector of length
#' \code{k} with the spending time at each analysis (see Details in help for \code{gsDesign
#' @param tIA Timing of an interim analysis; should be between 0 and
#' \code{y$T}.
#' @param target The targeted proportion of events at an interim analysis. This
#' is used for root-finding will be 0 for normal use.
#' @param simple See output specification for \code{nEventsIA()}.
#' @param footnote footnote for xtable output; may be useful for describing
#' some of the design parameters.
```

```
#' @param fnwid a text string controlling the width of footnote text at the
#' bottom of the xtable output.
#' @param timename character string with plural of time units (e.g., "months")
#' @param caption passed through to generic \code{xtable()}.
#' @param label passed through to generic \code{xtable()}.
#' @param align passed through to generic \code{xtable()}.
#' @param display passed through to generic \code{xtable()}.
#' @param auto passed through to generic \code{xtable()}.
#' @param ... other arguments that may be passed to generic functions
#' underlying the methods here.
#' @return \code{nSurv()} returns an object of type \code{nSurv} with the
#' following components: \item{alpha}{As input.} \item{sided}{As input.}
#' \item{beta}{Type II error; if missing, this is computed.} \item{power}{Power
#' corresponding to input \code{beta} or computed if output \code{beta} is
#' computed.} \item{lambdaC}{As input.} \item{etaC}{As input.} \item{etaE}{As
#' input.} \item{gamma}{As input unless none of the following are \code{NULL}:
#' \code{T}, \code{minfup}, \code{beta}; otherwise, this is a constant times
#' the input value required to power the trial given the other input
#' variables.} \item{ratio}{As input.} \item{R}{As input unless \code{T} was
#' \code{NULL} on input.} \item{S}{As input.} \item{T}{As input.}
#' \item{minfup}{As input.} \item{hr}{As input.} \item{hr0}{As input.}
#' \item{n}{Total expected sample size corresponding to output accrual rates
#' and durations.} \item{d}{Total expected number of events under the alternate
#' hypothesis.} \item{tol}{As input, except when not used in computations in
#' which case this is returned as \code{NULL}.  This and the remaining output
#' below are not printed by the \code{print()} extension for the \code{nSurv}
#' class.} \item{eDC}{A vector of expected number of events by stratum in the
#' control group under the alternate hypothesis.} \item{eDE}{A vector of
#' expected number of events by stratum in the experimental group under the
#' alternate hypothesis.} \item{eDC0}{A vector of expected number of events by
#' stratum in the control group under the null hypothesis.} \item{eDE0}{A
#' vector of expected number of events by stratum in the experimental group
#' under the null hypothesis.} \item{eNC}{A vector of the expected accrual in
#' each stratum in the control group.} \item{eNE}{A vector of the expected
#' accrual in each stratum in the experimental group.} \item{variable}{A text
#' string equal to "Accrual rate" if a design was derived by varying the
#' accrual rate, "Accrual duration" if a design was derived by varying the
#' accrual duration, "Follow-up duration" if a design was derived by varying
#' follow-up duration, or "Power" if accrual rates and duration as well as
#' follow-up duration was specified and \code{beta=NULL} was input.}
#'
#' \code{gsSurv()} returns much of the above plus an object of class
#' \code{gsDesign} in a variable named \code{gs}; see \code{\link{gsDesign}}
#' for general documentation on what is returned in \code{gs}.  The value of
```

```
#' \code{gs$n.I} represents the number of endpoints required at each analysis
#' to adequately power the trial. Other items returned by \code{gsSurv()} are:
#' \item{gs}{A group sequential design (\code{gsDesign}) object.}
#' \item{lambdaC}{As input.} \item{etaC}{As input.} \item{etaE}{As input.}
#' \item{gamma}{As input unless none of the following are \code{NULL}:
#' \code{T}, \code{minfup}, \code{beta}; otherwise, this is a constant times
#' the input value required to power the trial given the other input
#' variables.} \item{ratio}{As input.} \item{R}{As input unless \code{T} was
#' \code{NULL} on input.} \item{S}{As input.} \item{T}{As input.}
#' \item{minfup}{As input.} \item{hr}{As input.} \item{hr0}{As input.}
#' \item{eNC}{Total expected sample size corresponding to output accrual rates
#' and durations.} \item{eNE}{Total expected sample size corresponding to
#' output accrual rates and durations.} \item{eDC}{Total expected number of
#' events under the alternate hypothesis.} \item{eDE}{Total expected number of
#' events under the alternate hypothesis.} \item{tol}{As input, except when not
#' used in computations in which case this is returned as \code{NULL}.  This
#' and the remaining output below are not printed by the \code{print()}
#' extension for the \code{nSurv} class.} \item{eDC}{A vector of expected
#' number of events by stratum in the control group under the alternate
#' hypothesis.} \item{eDE}{A vector of expected number of events by stratum in
#' the experimental group under the alternate hypothesis.} \item{eDC0}{A vector
#' of expected number of events by stratum in the control group under the null
#' hypothesis.} \item{eDE0}{A vector of expected number of events by stratum in
#' the experimental group under the null hypothesis.} \item{eNC}{A vector of
#' the expected accrual in each stratum in the control group.} \item{eNE}{A
#' vector of the expected accrual in each stratum in the experimental group.}
#' \item{variable}{A text string equal to "Accrual rate" if a design was
#' derived by varying the accrual rate, "Accrual duration" if a design was
#' derived by varying the accrual duration, "Follow-up duration" if a design
#' was derived by varying follow-up duration, or "Power" if accrual rates and
#' duration as well as follow-up duration was specified and \code{beta=NULL}
#' was input.}
#'
#' \code{nEventsIA()} returns the expected proportion of the final planned
#' events observed at the input analysis time minus \code{target} when
#' \code{simple=TRUE}. When \code{simple=FALSE}, \code{nEventsIA} returns a
#' list with following components: \item{T}{The input value \code{tIA}.}
#' \item{eDC}{The expected number of events in the control group at time the
#' output time \code{T}.} \item{eDE}{The expected number of events in the
#' experimental group at the output time \code{T}.} \item{eNC}{The expected
#' enrollment in the control group at the output time \code{T}.} \item{eNE}{The
#' expected enrollment in the experimental group at the output time \code{T}.}
#'
#' \code{tEventsIA()} returns
```

```
#' @author Keaven Anderson \email{keaven_anderson@@merck.com}
#' @seealso \code{\link{gsBoundSummary}}, \code{\link{xprint}},
#' \link{gsDesign package overview}, \link{plot.gsDesign},
#' \code{\link{gsDesign}}, \code{\link{gsHR}}, \code{\link{nSurvival}}
#' @references Kim KM and Tsiatis AA (1990), Study duration for clinical trials
#' with survival response and early stopping rule. \emph{Biometrics}, 46, 81-92
#'
#' Lachin JM and Foulkes MA (1986), Evaluation of Sample Size and Power for
#' Analyses of Survival with Allowance for Nonuniform Patient Entry, Losses to
#' Follow-Up, Noncompliance, and Stratification. \emph{Biometrics}, 42,
#' 507-519.
#'
#' Schoenfeld D (1981), The Asymptotic Properties of Nonparametric Tests for
#' Comparing Survival Distributions. \emph{Biometrika}, 68, 316-319.
#' @keywords design
#' @examples
#'
#' # vary accrual rate to obtain power
#' nSurv(lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = 1, T = 36, minfup = 12)
#'
#' # vary accrual duration to obtain power
#' nSurv(lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = 6, minfup = 12)
#'
#' # vary follow-up duration to obtain power
#' nSurv(lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = 6, R = 25)
#'
#' # piecewise constant enrollment rates (vary accrual duration)
#' nSurv(
#'   lambdaC = log(2) / 6, hr = .5, eta = log(2) / 40, gamma = c(1, 3, 6),
#'   R = c(3, 6, 9), minfup = 12
#' )
#'
#' # stratified population (vary accrual duration)
#' nSurv(
#'   lambdaC = matrix(log(2) / c(6, 12), ncol = 2), hr = .5, eta = log(2) / 40,
#'   gamma = matrix(c(2, 4), ncol = 2), minfup = 12
#' )
#'
#' # piecewise exponential failure rates (vary accrual duration)
#' nSurv(lambdaC = log(2) / c(6, 12), hr = .5, eta = log(2) / 40, S = 3, gamma = 6, minfup =
#'
#' # combine it all: 2 strata, 2 failure rate periods
#' nSurv(
#'   lambdaC = matrix(log(2) / c(6, 12, 18, 24), ncol = 2), hr = .5,
```

```
#'   eta = matrix(log(2) / c(40, 50, 45, 55), ncol = 2), S = 3,
#'   gamma = matrix(c(3, 6, 5, 7), ncol = 2), R = c(5, 10), minfup = 12
#' )
#'
#' # example where only 1 month of follow-up is desired
#' # set failure rate to 0 after 1 month using lambdaC and S
#' nSurv(lambdaC = c(.4, 0), hr = 2 / 3, S = 1, minfup = 1)
#'
#' # group sequential design (vary accrual rate to obtain power)
#' x <- gsSurv(
#'   k = 4, sfl = sfPower, sflpar = .5, lambdaC = log(2) / 6, hr = .5,
#'   eta = log(2) / 40, gamma = 1, T = 36, minfup = 12
#' )
#' x
#' print(xtable::xtable(x,
#'   footnote = "This is a footnote; note that it can be wide.",
#'   caption = "Caption example."
#' ))
#' # find expected number of events at time 12 in the above trial
#' nEventsIA(x = x, tIA = 10)
#'
#' # find time at which 1/4 of events are expected
#' tEventsIA(x = x, timing = .25)
#' @export
# nSurv function [sinew] ----
nSurv <- function(lambdaC = log(2) / 6, hr = .6, hr0 = 1, eta = 0, etaE = NULL,
                  gamma = 1, R = 12, S = NULL, T = NULL, minfup = NULL, ratio = 1,
                  alpha = 0.025, beta = 0.10, sided = 1, tol = .Machine$double.eps^0.25) {
  if (is.null(etaE)) etaE <- eta
  # set up rates as matrices with row and column names
  # default is 1 stratum if lambdaC not input as matrix
  if (is.vector(lambdaC)) lambdaC <- matrix(lambdaC)
  ldim <- dim(lambdaC)
  nstrata <- ldim[2]
  nlambda <- ldim[1]
  rownames(lambdaC) <- paste("Period", 1:nlambda)
  colnames(lambdaC) <- paste("Stratum", 1:nstrata)
  etaC <- matrix(eta, nrow = nlambda, ncol = nstrata)
  etaE <- matrix(etaE, nrow = nlambda, ncol = nstrata)
  if (!is.matrix(gamma)) gamma <- matrix(gamma)
  gdim <- dim(gamma)
  eCdim <- dim(etaC)
  eEdim <- dim(etaE)
```

```
  if (is.null(minfup) || is.null(T)) {
    xx <- KT(
      lambdaC = lambdaC, hr = hr, hr0 = hr0, etaC = etaC, etaE = etaE,
      gamma = gamma, R = R, S = S, minfup = minfup, ratio = ratio,
      alpha = alpha, sided = sided, beta = beta, tol = tol
    )
  } else if (is.null(beta)) {
    xx <- KTZ(
      lambdaC = lambdaC, hr = hr, hr0 = hr0, etaC = etaC, etaE = etaE,
      gamma = gamma, R = R, S = S, minfup = minfup, ratio = ratio,
      alpha = alpha, sided = sided, beta = beta, simple = F
    )
  } else {
    xx <- LFPWE(
      lambdaC = lambdaC, hr = hr, hr0 = hr0, etaC = etaC, etaE = etaE,
      gamma = gamma, R = R, S = S, T = T, minfup = minfup, ratio = ratio,
      alpha = alpha, sided = sided, beta = beta
    )
  }

  nameR <- nameperiod(cumsum(xx$R))
  stratnames <- paste("Stratum", 1:ncol(xx$lambdaC))
  if (is.null(xx$S)) {
    nameS <- "0-Inf"
  } else {
    nameS <- nameperiod(cumsum(c(xx$S, Inf)))
  }
  rownames(xx$lambdaC) <- nameS
  colnames(xx$lambdaC) <- stratnames
  rownames(xx$etaC) <- nameS
  colnames(xx$etaC) <- stratnames
  rownames(xx$etaE) <- nameS
  colnames(xx$etaE) <- stratnames
  rownames(xx$gamma) <- nameR
  colnames(xx$gamma) <- stratnames
  return(xx)
}
```

2.7.2 Input variables

The input variables for `nSurv()` are:

- `lambdaC`: scalar, vector or matrix of event hazard rates for the control group; rows represent time periods while columns represent strata; a vector implies a single stratum.
- `hr`: hazard ratio (experimental/control) under the alternate hypothesis (scalar).
- `hr0`: hazard ratio (experimental/control) under the null hypothesis (scalar).
- `eta`: scalar, vector or matrix of dropout hazard rates for the control group; rows represent time periods while columns represent strata; if entered as a scalar, rate is constant accross strata and time periods; if entered as a vector, rates are constant accross strata.
- `etaE`: matrix dropout hazard rates for the experimental group specified in like form as `eta`; if `NULL`, this is set equal to `eta`.
- `gamma`: a scalar, vector or matrix of rates of entry by time period (rows) and strata (columns); if entered as a scalar, rate is constant accross strata and time periods; if entered as a vector, rates are constant accross strata.
- `R`: a scalar or vector of durations of time periods for recruitment rates specified in rows of `gamma`. Length is the same as number of rows in `gamma`. Note that when variable enrollment duration is specified, final enrollment period is extended as long as needed.
- `S`: a scalar or vector of durations of piecewise constant event rates specified in rows of `lambda`, `eta` and `etaE`; this is `NULL` if there is a single event rate per stratum (exponential failure) or of length 1 less than the number of rows in `lambda` otherwise.
- `T`: study duration.
- `minfup`: follow-up of last patient enrolled.
- `ratio`: randomization ratio of experimental treatment divided by control; normally a scalar, but can be a vector with length equal to number of strata.
- `sided`: 1 for 1-sided testing, 2 for 2-sided testing.
- `alpha`: type I error rate. Default is 0.025 since 1-sided testing is default.
- `beta`: type II error rate. Default is 0.10 (90% power); `NULL` if power is to be computed based on other input values.
- `tol`: for cases when values are derived through route finding, the level of error input to the `uniroot()` root-finding function.

2.7.3 Output structure

The output of `nSurv()` is of class `nSurv`, which is a list of the following items:

- `alpha`: As input.
- `sided`: As input.
- `beta`: Type II error; if missing, this is computed.

- `power`: Power corresponding to input `beta` or computed if output `beta` is computed.
- `lambdaC`: As input.
- `etaC`: As input.
- `etaE`: As input.
- `gamma`: As input unless none of the following are `NULL`: `T`, `minfup`, `beta`; otherwise, this is a constant times the input value required to power the trial given the other input variables.
- `ratio`: As input.
- `R`: As input unless `T` was `NULL` on input.
- `S`: As input.
- `T`: As input.
- `minfup`: As input.
- `hr`: As input.
- `hr0`: As input.
- `n`: Total expected sample size corresponding to output accrual rates and durations.
- `d`: Total expected number of events under the alternate hypothesis.
- `tol`: As input. This and the remaining output below are not printed by the `print()` extension for the `nSurv` class.
- `eDC`: A vector of expected number of events by stratum in the control group under the alternate hypothesis.
- `eDE`: A vector of expected number of events by stratum in the experimental group under the alternate hypothesis.
- `eDC0`: A vector of expected number of events by stratum in the control group under the null hypothesis.
- `eDE0`: A vector of expected number of events by stratum in the experimental group under the null hypothesis.
- `eNC`: A vector of the expected accrual in each stratum in the control group.
- `eNE`: A vector of the expected accrual in each stratum in the experimental group.

2.8 Documentation and primary examples for fixed design sample size and power

We present four examples for the advanced time-to-event sample size routine `nSurv()` corresponding to the four types of calculations peformed. All assume an input hazard ratio, dropout rates, Type I error rate. The first three assume an input Type II error rate, while the fourth computes power. The types of calculations are:

1. Based on a fixed enrollment duration and follow-up period, derive the enrollment rates required to obtain adequate power [Lachin and Foulkes, 1986].
2. Based on fixed enrollment rates and a fixed minimum follow-up period, derive accrual duration required to obtain adequate power [Kim and Tsiatis, 1990].
3. Based on fixed enrollment rates and accrual duration, derive study duration required to obtain adequate power [Kim and Tsiatis, 1990].
4. Based on fixed accrual duration, accrual rates and follow-up, derive study power.

2.8.1 Derive accrual rates

We begin by specifying values for all three of 1) control event rates (`lambdaC`), 2) accrual duration (`R`), and 3) minimum follow-up (`minfup`). Relative accrual rates are specified for the time periods specified in `R` in `gamma`; these are modified on output to give absolute accrual rates required to achieve the desired power.

```
x <- nSurv(
  lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = etaC, etaE = etaE,
  gamma = gamma, R = R, S = S, T = T, minfup = minfup, ratio = ratio,
  alpha = alpha, beta = beta
)
```

2.8.2 Derive accrual and study duration

Now we set the input study duration (`T`) to `NULL` so that it can be adjusted to achieve desired power; enrollment duration is altered to total study duration minus the input minimum follow-up.

```
x <- nSurv(
  lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = etaC, etaE = etaE,
  gamma = gamma, R = R, S = S, T = NULL, minfup = minfup, ratio = ratio,
  alpha = alpha, beta = beta
)
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual duration
  Hazard ratio                  H1/H0=0.5/1
```

```
  Study duration:                    T=35.836
  Accrual duration:                    25.836
  Min. end-of-study follow-up: minfup=10
  Expected events (total, H1):         88.3566
  Expected sample size (total):        206.6883
  Accrual rates:
          Stratum 1
  0-25.84         8
  Control event rates (H1):
        Stratum 1
  0-Inf    0.0347
  Censoring rates:
        Stratum 1
  0-Inf         0
  Power:                  100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:           ratio=1
```

```
KTZ(
  x = x$T - sum(x$R), lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = etaC, etaE = etaE,
  gamma = gamma, R = x$R, S = x$S, minfup = NULL, ratio = ratio,
  alpha = alpha, beta = .1, simple = TRUE
)
```

```
  [1] 6.233063e-08
```

2.8.3 Derive follow-up duration

Next we set the input follow-up (`minfup`) to `NULL` so that it can be derived to achieve the desired power. Note that this routine will fail if accrual is too little or too much in the specified time period. Thus, it is often easier to allow study and accrual duration to vary as just shown.

```
x <- nSurv(
  lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = etaC, etaE = etaE,
  gamma = gamma, R = R, S = S, T = T, minfup = NULL, ratio = ratio,
  alpha = alpha, beta = beta
)
```

2.8.4 Derive power

Finally, we compute the power for a design by setting the input `beta` to `NULL`.

```
nSurv(
  lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = etaC, etaE = etaE,
  gamma = gamma, R = R, S = S, T = T, minfup = minfup, ratio = ratio,
  alpha = alpha, beta = NULL
)
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Power
  Hazard ratio                   H1/H0=0.5/1
  Study duration:                    T=30
  Accrual duration:                    20
  Min. end-of-study follow-up: minfup=10
  Expected events (total, H1):        62.3423
  Expected sample size (total):       160
  Accrual rates:
       Stratum 1
  0-20         8
  Control event rates (H1):
        Stratum 1
  0-Inf    0.0347
  Censoring rates:
        Stratum 1
  0-Inf         0
  Power:                  100*(1-beta)=77.9917%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:          ratio=1
```

2.8.5 Examples with multiple strata, enrollment rates and failure rates

Fixed accrual duration and minimum follow-up duration — solve for accrual rate.

```
x <- nSurv(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2),
  hr = .6, S = c(3, 6), R = c(3, 3), minfup = 6, T = 30
)
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual rate
```

```
  Hazard ratio                   H1/H0=0.6/1
  Study duration:                    T=30
  Accrual duration:                    24
  Min. end-of-study follow-up: minfup=6
  Expected events (total, H1):         160.4241
  Expected sample size (total):        255.7091
  Accrual rates:
       Stratum 1 Stratum 2 Stratum 3
  0-3     1.2628    5.0510    3.7883
  3-24    2.5255    1.8941    6.3138
  Control event rates (H1):
        Stratum 1 Stratum 2 Stratum 3
  0-3      0.2310    0.1155    0.0770
  3-9      0.1733    0.0866    0.0578
  9-Inf    0.1386    0.0693    0.0462
  Censoring rates:
        Stratum 1 Stratum 2 Stratum 3
  0-3           0         0         0
  3-9           0         0         0
  9-Inf         0         0         0
  Power:                  100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:           ratio=1
```

Fixed accrual rate and minimum follow-up duration — solve for accrual duration.

```
x <- nSurv(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2),
  hr = .6, S = c(3, 6), R = c(3, 3), minfup = 6, T = NULL
)
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Accrual duration
  Hazard ratio                   H1/H0=0.6/1
  Study duration:                    T=22.647
  Accrual duration:                    16.647
  Min. end-of-study follow-up: minfup=6
  Expected events (total, H1):         160.7145
  Expected sample size (total):        279.9995
  Accrual rates:
          Stratum 1 Stratum 2 Stratum 3
  0-3             2         8         6
```

```
  3-16.65         4         3        10
  Control event rates (H1):
        Stratum 1 Stratum 2 Stratum 3
  0-3      0.2310    0.1155    0.0770
  3-9      0.1733    0.0866    0.0578
  9-Inf    0.1386    0.0693    0.0462
  Censoring rates:
        Stratum 1 Stratum 2 Stratum 3
  0-3           0         0         0
  3-9           0         0         0
  9-Inf         0         0         0
  Power:                  100*(1-beta)=90%
  Type I error (1-sided):   100*alpha=2.5%
  Equal randomization:           ratio=1
```

Fixed accrual rate and accrual duration — solve for minimum follow-up. Note that this option is generally not recommended as the accrual rate or accrual duration are the more common variables to solve for.

```
x <- nSurv(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2),
  hr = .6, S = c(3, 6), R = c(3, 15), minfup = NULL, T = 15
)
x
```

```
  Fixed design, two-arm trial with time-to-event
  outcome (Lachin and Foulkes, 1986).
  Solving for:  Follow-up duration
  Hazard ratio                  H1/H0=0.6/1
  Study duration:                   T=21.5363
  Accrual duration:                   18
  Min. end-of-study follow-up: minfup=3.5363
  Expected events (total, H1):        160.8979
  Expected sample size (total):       303
  Accrual rates:
       Stratum 1 Stratum 2 Stratum 3
  0-3          2         8         6
  3-18         4         3        10
  Control event rates (H1):
        Stratum 1 Stratum 2 Stratum 3
  0-3      0.2310    0.1155    0.0770
  3-9      0.1733    0.0866    0.0578
  9-Inf    0.1386    0.0693    0.0462
  Censoring rates:
        Stratum 1 Stratum 2 Stratum 3
```

```
0-3           0         0         0
3-9           0         0         0
9-Inf         0         0         0
Power:                  100*(1-beta)=90%
Type I error (1-sided):   100*alpha=2.5%
Equal randomization:          ratio=1
```

Chapter 3

Sample size and power for a group sequential design

We extend the fixed design calculations performed using `nSurv()` to compute sample size and power for a group sequential design. Classically, the required number of events is increased by a factor depending on the group sequential bounds to obtain an inflated statistical information. Under the null hypothesis of equal failure distributions, this is proportional to statistical information. While this adds another layer of approximation, we will follow this practice here. Thus, we will increase either accrual rates, accrual duration or follow-up duration to achieve the desired expected number of events under the alternate hypothesis. There are two required computations we will apply in deriving a group sequential design: first, inflating the expected number of events and second, deriving the times at which the expected number of events for each interim analysis equal the desired number.

3.1 Inflating the expected number of events

There are three cases to consider corresponding to the three ways we solved the sample size for a fixed design. These involve solving for 1) accrual rate, 2) accrual duration or 3) minimum follow-up duration required to obtain a desired number of events. To increase the expected number of events by an inflation factor when the study duration and relative enrollments are fixed over time requires only to increase the enrollment rate(s) by the same proportion. This also increases the expected sample size by the same proportion. The other two computations are simple applications of the function `KTZ()`.

3.2 Source code for inflating the expected number of events

We consider fixed design and a targeted number of events greater than that for a fixed design.

```
gsnSurv <- function(x, nEvents) {
  if (x$variable == "Accrual rate") {
    Ifct <- nEvents / x$d
    rval <- list(
      lambdaC = x$lambdaC, etaC = x$etaC, etaE = x$etaE,
      gamma = x$gamma * Ifct, ratio = x$ratio, R = x$R, S = x$S, T = x$T,
      minfup = x$minfup, hr = x$hr, hr0 = x$hr0, n = x$n * Ifct, d = nEvents,
      eDC = x$eDC * Ifct, eDE = x$eDE * Ifct, eDC0 = x$eDC0 * Ifct,
      eDE0 = x$eDE0 * Ifct, eNC = x$eNC * Ifct, eNE = x$eNE * Ifct,
      variable = x$variable
    )
  } else if (x$variable == "Accrual duration") {
    y <- KT(
      n1Target = nEvents, minfup = x$minfup, lambdaC = x$lambdaC,
      etaC = x$etaC, etaE = x$etaE,
      R = x$R, S = x$S, hr = x$hr, hr0 = x$hr0, gamma = x$gamma,
      ratio = x$ratio, tol = x$tol
    )
    rval <- list(
      lambdaC = x$lambdaC, etaC = x$etaC, etaE = x$etaE,
      gamma = x$gamma, ratio = x$ratio, R = y$R, S = x$S, T = y$T,
      minfup = y$minfup, hr = x$hr, hr0 = x$hr0,
      n = y$n, d = nEvents,
      eDC = y$eDC, eDE = y$eDE, eDC0 = y$eDC0,
      eDE0 = y$eDE0, eNC = y$eNC, eNE = y$eNE, tol = x$tol,
      variable = x$variable
    )
  } else {
    y <- KT(
      n1Target = nEvents, minfup = NULL,
      lambdaC = x$lambdaC, etaC = x$etaC,
      etaE = x$etaE, R = x$R, S = x$S, hr = x$hr, hr0 = x$hr0,
      gamma = x$gamma, ratio = x$ratio, tol = x$tol
    )
    rval <- list(
      lambdaC = x$lambdaC, etaC = x$etaC, etaE = x$etaE,
      gamma = x$gamma, ratio = x$ratio, R = x$R, S = x$S, T = y$T,
      minfup = y$minfup, hr = x$hr, hr0 = x$hr0, n = y$n,
      d = nEvents, eDC = y$eDC, eDE = y$eDE, eDC0 = y$eDC0,
```

```
      eDE0 = y$eDE0, eNC = y$eNC, eNE = y$eNE, tol = x$tol,
      variable = x$variable
    )
  }
  class(rval) <- "gsSize"
  return(rval)
}
```

```
x <- nSurv(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2),
  hr = .6, S = c(3, 6), R = c(3, 15), minfup = NULL, T = 15
)
x
```

```
$alpha
[1] 0.025

$sided
[1] 1

$beta
[1] 0.09999999

$power
[1] 0.9

$lambdaC
       Stratum 1  Stratum 2  Stratum 3
0-3    0.2310491 0.11552453 0.07701635
3-9    0.1732868 0.08664340 0.05776227
9-Inf  0.1386294 0.06931472 0.04620981

$etaC
       Stratum 1 Stratum 2 Stratum 3
0-3            0         0         0
3-9            0         0         0
9-Inf          0         0         0

$etaE
       Stratum 1 Stratum 2 Stratum 3
0-3            0         0         0
3-9            0         0         0
9-Inf          0         0         0
```

```
$gamma
     Stratum 1 Stratum 2 Stratum 3
0-3          2         8         6
3-18         4         3        10

$ratio
[1] 1

$R
[1]  3 15

$S
[1] 3 6

$T
[1] 21.53628

$minfup
[1] 3.536276

$hr
[1] 0.6

$hr0
[1] 1

$n
[1] 303

$d
[1] 160.8979

$tol
[1] 0.0001220703

$eDC
[1] 27.79551 23.34026 41.29445

$eDE
[1] 22.71825 17.33116 28.41828

$eDC0
[1] 25.75003 20.70618 35.33690

$eDE0
```

```
[1] 25.75003 20.70618 35.33690

$eNC
[1] 33.0 34.5 84.0

$eNE
[1] 33.0 34.5 84.0

$variable
[1] "Follow-up duration"

attr(,"class")
[1] "nSurv"
```

```
gsnSurv(x, 170)
```

```
$lambdaC
       Stratum 1  Stratum 2  Stratum 3
0-3   0.2310491 0.11552453 0.07701635
3-9   0.1732868 0.08664340 0.05776227
9-Inf 0.1386294 0.06931472 0.04620981

$etaC
       Stratum 1 Stratum 2 Stratum 3
0-3            0         0         0
3-9            0         0         0
9-Inf          0         0         0

$etaE
       Stratum 1 Stratum 2 Stratum 3
0-3            0         0         0
3-9            0         0         0
9-Inf          0         0         0

$gamma
      Stratum 1 Stratum 2 Stratum 3
0-3           2         8         6
3-18          4         3        10

$ratio
[1] 1

$R
[1]  3 15

$S
```

```
[1] 3 6

$T
[1] 22.85288

$minfup
[1] 4.85288

$hr
[1] 0.6

$hr0
[1] 1

$n
[1] 303

$d
[1] 170

$eDC
[1] 28.77412 24.39549 44.04440

$eDE
[1] 23.89893 18.31081 30.57626

$eDC0
[1] 26.85077 21.75258 37.84991

$eDE0
[1] 26.85077 21.75258 37.84991

$eNC
[1] 33.0 34.5 84.0

$eNE
[1] 33.0 34.5 84.0

$tol
[1] 0.0001220703

$variable
[1] "Follow-up duration"

attr(,"class")
```

```
[1] "gsSize"
```

3.3 Enrollment at interim analysis

We begin with a simple function to compute enrollment given enrollment rates, enrollment periods and duration of enrollment. The input variable `y` should be an object of class `nSurv`, `gsSize` or `gsSurv`. This specifies enrollment and event rates. `nEventsIA()` computes the expected number of events at an interim analysis performed at a specific time specified in the variable `tIA`. `tEventsIA()` uses root-finding to derive the time at which the expected number of events is equal to a planned interim proportion (specified in the input `timing`) of the final planned events. The input variable `tol` will normally be left as the default by the user; this is used in the root-finding routine `uniroot()`. Note that `nEventsIA()` was updated in August, 2015 so that the `simple = TRUE` option works correctly for stratified populations; this impacted `tEventsIA()` as well.

```
# tEventsIA roxy [sinew] ----
#' @export
#' @rdname nSurv
#' @seealso
#'  \code{\link[stats]{uniroot}}
#' @importFrom stats uniroot
# tEventsIA function [sinew] ----
tEventsIA <- function(x, timing = .25,
                      tol = .Machine$double.eps^0.25) {
  T <- x$T[length(x$T)]
  z <- stats::uniroot(
    f = nEventsIA, interval = c(0.0001, T - .0001), x = x,
    target = timing, tol = tol, simple = TRUE
  )
  return(nEventsIA(tIA = z$root, x = x, simple = FALSE))
}
```

```
# nEventsIA roxy [sinew] ----
#' @export
#' @rdname nSurv
# nEventsIA function [sinew] ----
nEventsIA <- function(tIA = 5, x = NULL, target = 0,
                      simple = TRUE) {
  Qe <- x$ratio / (1 + x$ratio)
  eDC <- eEvents(
    lambda = x$lambdaC, eta = x$etaC,
    gamma = x$gamma * (1 - Qe), R = x$R, S = x$S, T = tIA,
```

```
    Tfinal = x$T[length(x$T)], minfup = x$minfup
  )
  eDE <- eEvents(
    lambda = x$lambdaC * x$hr, eta = x$etaC,
    gamma = x$gamma * Qe, R = x$R, S = x$S, T = tIA,
    Tfinal = x$T[length(x$T)], minfup = x$minfup
  )
  if (simple) {
    if (class(x)[1] == "gsSize") {
      d <- x$d
    } else if (!is.matrix(x$eDC)) {
      d <- sum(x$eDC[length(x$eDC)] + x$eDE[length(x$eDE)])
    } else {
      d <- sum(x$eDC[nrow(x$eDC), ] + x$eDE[nrow(x$eDE), ])
    }
    return(sum(eDC$d + eDE$d) - target * d)
  } else {
    return(list(
      T = tIA, eDC = eDC$d, eDE = eDE$d,
      eNC = eDC$n, eNE = eDE$n
    ))
  }
}
```

3.4 The general group sequential design approach

We provide two variations, one with event-based (information-based) timing of analyses (`gsSurv()`), the other with calendar-based timing (`gsSurvCalendar()`). `gsSurv()` is being updated in August, 2023 as it inadvertently did not return expected event counts under null hypothesis when originally written. `gsSurvCalendar()` is being added in August, 2023. In addition to adding calendar-based timing, `gsSurvCalendar()` can be used to compute power given enrollment, failure rate, and dropout rate assumptions. This capability is not currently available for `gsSurv()`.

```
# gsSurv roxy [sinew] ----
#' @export
# gsSurv function [sinew] ----
gsSurv <- function(k = 3, test.type = 4, alpha = 0.025, sided = 1,
                   beta = 0.1, astar = 0, timing = 1, sfu = gsDesign::sfHSD, sfupar = -4,
                   sfl = gsDesign::sfHSD, sflpar = -2, r = 18,
                   lambdaC = log(2) / 6, hr = .6, hr0 = 1, eta = 0, etaE = NULL,
                   gamma = 1, R = 12, S = NULL, T = NULL, minfup = NULL, ratio = 1,
```

```
                    tol = .Machine$double.eps^0.25,
                    usTime = NULL, lsTime = NULL) # KA: last 2 arguments added 10/8/2017
{
  x <- nSurv(
    lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = eta, etaE = etaE,
    gamma = gamma, R = R, S = S, T = T, minfup = minfup, ratio = ratio,
    alpha = alpha, beta = beta, sided = sided, tol = tol
  )
  y <- gsDesign::gsDesign(
    k = k, test.type = test.type, alpha = alpha / sided,
    beta = beta, astar = astar, n.fix = x$d, timing = timing,
    sfu = sfu, sfupar = sfupar, sfl = sfl, sflpar = sflpar, tol = tol,
    delta1 = log(hr), delta0 = log(hr0),
    usTime = usTime, lsTime = lsTime
  ) # KA: last 2 arguments added 10/8/2017

  z <- gsnSurv(x, y$n.I[k])
  eDC <- NULL
  eDE <- NULL
  eDC0 <- NULL
  eDE0 <- NULL
  eNC <- NULL
  eNE <- NULL
  T <- NULL
  for (i in 1:(k - 1)) {
    xx <- tEventsIA(z, y$timing[i], tol)
    T <- c(T, xx$T)
    eDC <- rbind(eDC, xx$eDC)
    eDE <- rbind(eDE, xx$eDE)
    # Following a placeholder
    #    eDC0 <- rbind(eDC0, xx$eDC0) # Added 6/15/2023
    #    eDE0 <- rbind(eDE0, xx$eDE0) # Added 6/15/2023
    eNC <- rbind(eNC, xx$eNC)
    eNE <- rbind(eNE, xx$eNE)
  }
  y$T <- c(T, z$T)
  y$eDC <- rbind(eDC, z$eDC)
  y$eDE <- rbind(eDE, z$eDE)
  # Following is a placeholder
  #  y$eDC0 <- rbind(eDC0, z$eDC0) # Added 6/15/2023
  #  y$eDE0 <- rbind(eDE0, z$eDE0) # Added 6/15/2023
  y$eNC <- rbind(eNC, z$eNC)
  y$eNE <- rbind(eNE, z$eNE)
  y$hr <- hr
```

```
  y$hr0 <- hr0
  y$R <- z$R
  y$S <- z$S
  y$minfup <- z$minfup
  y$gamma <- z$gamma
  y$ratio <- ratio
  y$lambdaC <- z$lambdaC
  y$etaC <- z$etaC
  y$etaE <- z$etaE
  y$variable <- x$variable
  y$tol <- tol
  class(y) <- c("gsSurv", "gsDesign")

  nameR <- nameperiod(cumsum(y$R))
  stratnames <- paste("Stratum", 1:ncol(y$lambdaC))
  if (is.null(y$S)) {
    nameS <- "0-Inf"
  } else {
    nameS <- nameperiod(cumsum(c(y$S, Inf)))
  }
  rownames(y$lambdaC) <- nameS
  colnames(y$lambdaC) <- stratnames
  rownames(y$etaC) <- nameS
  colnames(y$etaC) <- stratnames
  rownames(y$etaE) <- nameS
  colnames(y$etaE) <- stratnames
  rownames(y$gamma) <- nameR
  colnames(y$gamma) <- stratnames
  return(y)
}
```

```
# print.gsSurv roxy [sinew] ----
#' @rdname nSurv
#' @export
# print.gsSurv function [sinew] ----
print.gsSurv <- function(x, digits = 2, ...) {
  cat("Time to event group sequential design with HR=", x$hr, "\n")
  if (x$hr0 != 1) cat("Non-inferiority design with null HR=", x$hr0, "\n")
  if (min(x$ratio == 1) == 1) {
    cat("Equal randomization:          ratio=1\n")
  } else {
    cat(
      "Randomization (Exp/Control):  ratio=",
      x$ratio, "\n"
    )
```

```
    if (length(x$ratio) > 1) cat("(randomization ratios shown by strata)\n")
  }
  gsDesign:::print.gsDesign(x)
  if (x$test.type != 1) {
    y <- cbind(
      x$T, (x$eNC + x$eNE) %*% rep(1, ncol(x$eNE)),
      (x$eDC + x$eDE) %*% rep(1, ncol(x$eNE)),
      round(gsDesign::zn2hr(x$lower$bound, x$n.I, x$ratio, hr0 = x$hr0, hr1 = x$hr), 3),
      round(gsDesign::zn2hr(x$upper$bound, x$n.I, x$ratio, hr0 = x$hr0, hr1 = x$hr), 3)
    )
    colnames(y) <- c("T", "n", "Events", "HR futility", "HR efficacy")
  } else {
    y <- cbind(
      x$T, (x$eNC + x$eNE) %*% rep(1, ncol(x$eNE)),
      (x$eDC + x$eDE) %*% rep(1, ncol(x$eNE)),
      round(gsDesign::zn2hr(x$upper$bound, x$n.I, x$ratio, hr0 = x$hr0, hr1 = x$hr), 3)
    )
    colnames(y) <- c("T", "n", "Events", "HR efficacy")
  }
  rnames <- paste("IA", 1:(x$k))
  rnames[length(rnames)] <- "Final"
  rownames(y) <- rnames
  print(y)
  cat("Accrual rates:\n")
  print(round(x$gamma, digits))
  cat("Control event rates (H1):\n")
  print(round(x$lambda, digits))
  if (max(abs(x$etaC - x$etaE)) == 0) {
    cat("Censoring rates:\n")
    print(round(x$etaC, digits))
  } else {
    cat("Control censoring rates:\n")
    print(round(x$etaC, digits))
    cat("Experimental censoring rates:\n")
    print(round(x$etaE, digits))
  }
}
```

We also provide an xtable function for gsSurv object types

```
# xtable.gsSurv roxy [sinew] ----
#' @seealso
#'  \code{\link[stats]{Normal}}
#'  \code{\link[xtable]{xtable}}
#' @importFrom stats pnorm
```

```
#' @rdname nSurv
#' @export
# xtable.gsSurv function [sinew] ----
xtable.gsSurv <- function(x, caption = NULL, label = NULL, align = NULL, digits = NULL,
                          display = NULL, auto = FALSE, footnote = NULL, fnwid = "9cm", time
  k <- x$k
  stat <- c(
    "Z-value", "HR", "p (1-sided)", paste("P\\{Cross\\} if HR=", x$hr0, sep = ""),
    paste("P\\{Cross\\} if HR=", x$hr, sep = "")
  )
  st <- stat
  for (i in 2:k) stat <- c(stat, st)
  an <- rep(" ", 5 * k)
  tim <- an
  enrol <- an
  fut <- an
  eff <- an
  an[5 * (0:(k - 1)) + 1] <- c(paste("IA ", as.character(1:(k - 1)), ": ",
    as.character(round(100 * x$timing[1:(k - 1)], 1)), "\\%",
    sep = ""
  ), "Final analysis")
  an[5 * (1:(k - 1)) + 1] <- paste("\\hline", an[5 * (1:(k - 1)) + 1])
  an[5 * (0:(k - 1)) + 2] <- paste("N:", ceiling(rowSums(x$eNC)) + ceiling(rowSums(x$eNE)))
  an[5 * (0:(k - 1)) + 3] <- paste("Events:", ceiling(rowSums(x$eDC + x$eDE)))
  an[5 * (0:(k - 1)) + 4] <- paste(round(x$T, 1), timename, sep = " ")
  if (x$test.type != 1) fut[5 * (0:(k - 1)) + 1] <- as.character(round(x$lower$bound, 2))
  eff[5 * (0:(k - 1)) + 1] <- as.character(round(x$upper$bound, 2))
  if (x$test.type != 1) fut[5 * (0:(k - 1)) + 2] <- as.character(round(gsHR(z = x$lower$bou
  eff[5 * (0:(k - 1)) + 2] <- as.character(round(gsHR(z = x$upper$bound, i = 1:k, x, ratio =
  asp <- as.character(round(stats::pnorm(-x$upper$bound), 4))
  asp[asp == "0"] <- "$< 0.0001$"
  eff[5 * (0:(k - 1)) + 3] <- asp
  asp <- as.character(round(cumsum(x$upper$prob[, 1]), 4))
  asp[asp == "0"] <- "$< 0.0001$"
  eff[5 * (0:(k - 1)) + 4] <- asp
  asp <- as.character(round(cumsum(x$upper$prob[, 2]), 4))
  asp[asp == "0"] <- "$< 0.0001$"
  eff[5 * (0:(k - 1)) + 5] <- asp
  if (x$test.type != 1) {
    bsp <- as.character(round(stats::pnorm(-x$lower$bound), 4))
    bsp[bsp == "0"] <- " $< 0.0001$"
    fut[5 * (0:(k - 1)) + 3] <- bsp
    bsp <- as.character(round(cumsum(x$lower$prob[, 1]), 4))
    bsp[bsp == "0"] <- "$< 0.0001$"
```

```
    fut[5 * (0:(k - 1)) + 4] <- bsp
    bsp <- as.character(round(cumsum(x$lower$prob[, 2]), 4))
    bsp[bsp == "0"] <- "$< 0.0001$"
    fut[5 * (0:(k - 1)) + 5] <- bsp
  }
  neff <- length(eff)
  if (!is.null(footnote)) {
    eff[neff] <-
      paste(eff[neff], "\\\\ \\hline \\multicolumn{4}{p{", fnwid, "}}{\\footnotesize", footn
  }
  if (x$test.type != 1) {
    xxtab <- data.frame(cbind(an, stat, fut, eff))
    colnames(xxtab) <- c("Analysis", "Value", "Futility", "Efficacy")
  } else {
    xxtab <- data.frame(cbind(an, stat, eff))
    colnames(xxtab) <- c("Analysis", "Value", "Efficacy")
  }
  return(xtable::xtable(xxtab,
    caption = caption, label = label, align = align, digits = digits,
    display = display, auto = auto, ...
  ))
}
```

3.5 Examples

3.5.1 Simple

Fixed accrual duration and follow-up duration, variable accrual rate.

```
w <- LFPWE(lambdaC = log(2) / 3, hr = .6, minfup = 6, T = 36)
y <- nSurv(lambdaC = log(2) / 3, hr = .6, minfup = 6, T = 36, R = Inf)
x <- gsSurv(lambdaC = log(2) / 3, hr = .6, minfup = 6, T = 36)
```

Fixed accrual rate and follow-up duration, variable accrual duration.

```
x <- gsSurv(lambdaC = log(2) / 6, hr = .6, gamma = 8, minfup = 6)
x
```

```
  Time to event group sequential design with HR= 0.6
  Equal randomization:          ratio=1
  Asymmetric two-sided group sequential design with
  90 % power and 2.5 % Type I Error.
  Upper bound spending computations assume
  trial continues if lower bound is crossed.
```

```
                ----Lower bounds----  ----Upper bounds-----
  Analysis  N    Z   Nominal p Spend+  Z   Nominal p Spend++
         1  58 -0.24    0.4056 0.0148 3.01    0.0013  0.0013
         2 115  0.94    0.8267 0.0289 2.55    0.0054  0.0049
         3 172  2.00    0.9772 0.0563 2.00    0.0228  0.0188
     Total                     0.1000                 0.0250
+ lower bound beta spending (under H1):
 Hwang-Shih-DeCani spending function with gamma = -2.
++ alpha spending:
 Hwang-Shih-DeCani spending function with gamma = -4.

Boundary crossing probabilities and expected sample size
assume any cross stops the trial

Upper boundary (power or Type I Error)
          Analysis
   Theta      1      2      3  Total  E{N}
  0.0000 0.0013 0.0049 0.0171 0.0233  99.9
  0.2564 0.1412 0.4403 0.3185 0.9000 126.5

Lower boundary (futility or Type II Error)
          Analysis
   Theta      1      2      3  Total
  0.0000 0.4056 0.4290 0.1420 0.9767
  0.2564 0.0148 0.0289 0.0563 0.1000
            T        n    Events HR futility HR efficacy
IA 1  15.47476 123.7981  57.00202       1.065       0.450
IA 2  24.17700 193.4160 114.00405       0.838       0.621
Final 33.50127 220.0102 171.00607       0.737       0.737
Accrual rates:
       Stratum 1
0-27.5         8
Control event rates (H1):
      Stratum 1
0-Inf      0.12
Censoring rates:
      Stratum 1
0-Inf         0
```

Fixed accrual rate and accrual duration, variable follow-up duration.

```
x <- gsSurv(lambdaC = log(2) / 6, hr = .6, gamma = 8, R = 25, minfup = NULL)
```

3.5.2 Piecewise enrollment and failure rates

```
x <- gsSurv(
  lambdaC = log(2) / c(6, 8, 10), hr = .6, gamma = c(2, 4),
  S = c(3, 6), R = c(3, 3), minfup = 6, T = 20
)
x
```

```
Time to event group sequential design with HR= 0.6
Equal randomization:          ratio=1
Asymmetric two-sided group sequential design with
90 % power and 2.5 % Type I Error.
Upper bound spending computations assume
trial continues if lower bound is crossed.

                ----Lower bounds----  ----Upper bounds-----
  Analysis  N    Z   Nominal p Spend+  Z   Nominal p Spend++
         1  58 -0.24    0.4056 0.0148 3.01    0.0013  0.0013
         2 115  0.94    0.8267 0.0289 2.55    0.0054  0.0049
         3 173  2.00    0.9772 0.0563 2.00    0.0228  0.0188
     Total                     0.1000                 0.0250
+ lower bound beta spending (under H1):
 Hwang-Shih-DeCani spending function with gamma = -2.
++ alpha spending:
 Hwang-Shih-DeCani spending function with gamma = -4.

Boundary crossing probabilities and expected sample size
assume any cross stops the trial

Upper boundary (power or Type I Error)
          Analysis
   Theta      1      2      3  Total  E{N}
  0.0000 0.0013 0.0049 0.0171 0.0233 100.6
  0.2554 0.1412 0.4403 0.3185 0.9000 127.4

Lower boundary (futility or Type II Error)
          Analysis
   Theta      1      2      3  Total
  0.0000 0.4056 0.4290 0.1420 0.9767
  0.2554 0.0148 0.0289 0.0563 0.1000
              T        n    Events HR futility HR efficacy
IA 1   9.827039 203.8729  57.42358       1.065       0.452
IA 2  14.277264 306.0405 114.84716       0.839       0.622
Final 20.000000 306.0405 172.27073       0.737       0.737
Accrual rates:
```

```
      Stratum 1
0-3       12.24
3-14      24.48
Control event rates (H1):
      Stratum 1
0-3         0.12
3-9         0.09
9-Inf       0.07
Censoring rates:
      Stratum 1
0-3           0
3-9           0
9-Inf         0
```

3.5.3 Stratified population

We begin with the example of Bernstein and Lagakos [1978] using the `LFPWE()` routine and compare to the `LFPWEnew()` routine. Their results with a single (H1) variance formulation at a sample size of 172 and 150 expected events. With `gsDesign2::gs_designAHR()` with a 2-variance formulation comparable to here, the results are N=182 and 151 events.

```
LFPWE(
  alpha = .05, beta = .2, hr = 2 / 3, ratio = 1,
  R = 2, S = NULL, T = 4,
  lambdaC = matrix(c(1, .8, .5), nrow = 1),
  etaC = matrix(0, nrow = 1, ncol = 3),
  etaE = matrix(0, nrow = 1, ncol = 3),
  gamma = matrix(c(.4, .4, .2), nrow = 1)
)
```

```
Fixed design, two-arm trial with time-to-event
outcome (Lachin and Foulkes, 1986).
Solving for:  Accrual rate
Hazard ratio                  H1/H0=0.6667/1
Study duration:                   T=4
Accrual duration:                   2
Min. end-of-study follow-up: minfup=2
Expected events (total, H1):        149.455
Expected sample size (total):       178.7758
Accrual rates:
        [,1]    [,2]    [,3]
[1,] 35.7552 35.7552 17.8776
Control event rates (H1):
```

```
     [,1] [,2] [,3]
[1,]    1  0.8  0.5
Censoring rates:
     [,1] [,2] [,3]
[1,]    0    0    0
Power:                  100*(1-beta)=80%
Type I error (1-sided):   100*alpha=5%
Equal randomization:           ratio=1
```

Now compare to new version.

```
# NOT READY
LFPWEnew(
  alpha = .05, beta = .2, hr = 2 / 3, ratio = 1,
  R = 2, S = NULL, T = 4,
  lambdaC = matrix(c(1, .8, .5), nrow = 1),
  etaC = matrix(0, nrow = 1, ncol = 3),
  etaE = matrix(0, nrow = 1, ncol = 3),
  gamma = matrix(c(.4, .4, .2), nrow = 1)
)
```

```
x <- gsSurv(
  alpha = .05, beta = .2, lambda = matrix(c(1, .8, .5), nrow = 1),
  hr = 2 / 3, gamma = matrix(c(.4, .4, .2), nrow = 1), minfup = 2, S = NULL, T = 4, ratio =
)
x
```

```
Time to event group sequential design with HR= 0.6666667
Randomization (Exp/Control):  ratio= 1 2 3
(randomization ratios shown by strata)
Asymmetric two-sided group sequential design with
80 % power and 5 % Type I Error.
Upper bound spending computations assume
trial continues if lower bound is crossed.

                ----Lower bounds----  ----Upper bounds-----
  Analysis  N    Z   Nominal p Spend+  Z   Nominal p Spend++
         1  56 -0.40    0.3454 0.0297 2.79    0.0026  0.0026
         2 112  0.67    0.7473 0.0578 2.29    0.0110  0.0099
         3 167  1.68    0.9535 0.1125 1.68    0.0465  0.0375
     Total                     0.2000                 0.0500
+ lower bound beta spending (under H1):
 Hwang-Shih-DeCani spending function with gamma = -2.
++ alpha spending:
 Hwang-Shih-DeCani spending function with gamma = -4.

Boundary crossing probabilities and expected sample size
```

```
assume any cross stops the trial

Upper boundary (power or Type I Error)
          Analysis
   Theta      1      2      3  Total  E{N}
  0.0000 0.0026 0.0099 0.0343 0.0468 104.6
  0.1995 0.0958 0.3382 0.3660 0.8000 130.9

Lower boundary (futility or Type II Error)
          Analysis
   Theta      1      2      3  Total
  0.0000 0.3454 0.4138 0.1940 0.9532
  0.1995 0.0297 0.0578 0.1125 0.2000
             T        n    Events HR futility HR efficacy
IA 1  1.523440 154.7076  55.61556       1.113       0.473
IA 2  2.372963 203.1029 111.23112       0.875       0.631
Final 4.000000 203.1029 166.84668       0.741       0.741
Accrual rates:
    Stratum 1 Stratum 2 Stratum 3
0-2     40.62     40.62     20.31
Control event rates (H1):
      Stratum 1 Stratum 2 Stratum 3
0-Inf         1       0.8       0.5
Censoring rates:
      Stratum 1 Stratum 2 Stratum 3
0-Inf         0         0         0
```

3.5.4 Stratified population, piecewise enrollment and failure

Here we have three strata (columns of `lambdaC` and `gamma`), two enrollment rate periods (≤ 3 months and > 3 months in `R`; rates in rows of `gamma`) and three failure rate periods (0–3, 3–6, > 6 months); rates in rows of `lambdaC`). We also have the default of 3 analyses (`k = 3`) and default spending functions. Here are the failure rate assumptions. Rows represent time periods and columns represent strata.

```
lambdaC_stratified <- log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3)
lambdaC_stratified
```

```
          [,1]       [,2]       [,3]
[1,] 0.2310491 0.11552453 0.07701635
[2,] 0.1732868 0.08664340 0.05776227
[3,] 0.1386294 0.06931472 0.04620981
```

Associated with these event rates are durations of time periods for each rate. Again, rows represent time periods and columns represent strata. Note that this is one less row here as the final event rate above issumed to continue indefinitely. Notice also that the durations of rates can, as here, be different for different strata.

```
gamma_stratified <- matrix(c(2, 4, 8, 3, 6, 10), nrow = 2, byrow = TRUE)
gamma_stratified
```

```
     [,1] [,2] [,3]
[1,]    2    4    8
[2,]    3    6   10
```

We begin with the case where accrual duration and minimum follow-up duration are fixed and we solve for accrual rates.

```
x <- gsSurv(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2, byrow = TRUE),
  hr = .6, S = c(3, 6), R = c(3, 3), minfup = 6, T = 30, ratio = 2, hr0 = 1.1
)
x
```

```
Time to event group sequential design with HR= 0.6
Non-inferiority design with null HR= 1.1
Randomization (Exp/Control):  ratio= 2
Asymmetric two-sided group sequential design with
90 % power and 2.5 % Type I Error.
Upper bound spending computations assume
trial continues if lower bound is crossed.

                ----Lower bounds----  ----Upper bounds-----
  Analysis  N    Z   Nominal p Spend+  Z   Nominal p Spend++
         1  45 -0.24    0.4056 0.0148 3.01    0.0013  0.0013
         2  90  0.94    0.8267 0.0289 2.55    0.0054  0.0049
         3 134  2.00    0.9772 0.0563 2.00    0.0228  0.0188
     Total                     0.1000                 0.0250
+ lower bound beta spending (under H1):
 Hwang-Shih-DeCani spending function with gamma = -2.
++ alpha spending:
 Hwang-Shih-DeCani spending function with gamma = -4.

Boundary crossing probabilities and expected sample size
assume any cross stops the trial

Upper boundary (power or Type I Error)
          Analysis
```

```
    Theta      1      2      3  Total E{N}
   0.0000 0.0013 0.0049 0.0171 0.0233 78.2
   0.2897 0.1412 0.4403 0.3185 0.9000 99.1

  Lower boundary (futility or Type II Error)
           Analysis
    Theta      1      2      3  Total
   0.0000 0.4056 0.4290 0.1420 0.9767
   0.2897 0.0148 0.0289 0.0563 0.1000
              T        n    Events HR futility HR efficacy
  IA 1  14.30629 132.9678  44.64569       1.187       0.423
  IA 2  21.89133 207.5835  89.29137       0.891       0.621
  Final 30.00000 228.3269 133.93706       0.763       0.763
  Accrual rates:
       Stratum 1 Stratum 2 Stratum 3
  0-3       1.04      2.07      4.14
  3-24      1.55      3.11      5.18
  Control event rates (H1):
        Stratum 1 Stratum 2 Stratum 3
  0-3        0.23      0.12      0.08
  3-9        0.17      0.09      0.06
  9-Inf      0.14      0.07      0.05
  Censoring rates:
        Stratum 1 Stratum 2 Stratum 3
  0-3           0         0         0
  3-9           0         0         0
  9-Inf         0         0         0
```

The total number of enrollment and events at each analysis is printed above. Following we see the enrollment and expected number of events by stratum in rows of the following matrices.

```
x$eNC + x$eNE
```

```
           [,1]     [,2]      [,3]
  [1,] 20.66792 41.33584  70.96406
  [2,] 32.44934 64.89867 110.23544
  [3,] 35.72462 71.44923 121.15304
```

```
x$eDC + x$eDE
```

```
           [,1]     [,2]     [,3]
  [1,] 11.39809 14.59480 18.65280
  [2,] 21.48863 29.53394 38.26873
  [3,] 29.89590 44.55811 59.48305
```

We proceed to the case where accrual rate and minimum follow-up duration are fixed and solve for accrual duration.

```
x <- gsSurv(
  lambdaC = log(2) / matrix(c(3, 4, 5, 6, 8, 10, 9, 12, 15), nrow = 3),
  gamma = matrix(c(2, 4, 8, 3, 6, 10), nrow = 2, byrow = TRUE),
  hr = .6, S = c(3, 6), R = c(3, 3), minfup = 6, T = NULL
)
x
```

```
  Time to event group sequential design with HR= 0.6
  Equal randomization:          ratio=1
  Asymmetric two-sided group sequential design with
  90 % power and 2.5 % Type I Error.
  Upper bound spending computations assume
  trial continues if lower bound is crossed.

                  ----Lower bounds----  ----Upper bounds-----
    Analysis  N    Z   Nominal p Spend+  Z   Nominal p Spend++
           1  58 -0.24    0.4056 0.0148 3.01    0.0013  0.0013
           2 115  0.94    0.8267 0.0289 2.55    0.0054  0.0049
           3 173  2.00    0.9772 0.0563 2.00    0.0228  0.0188
       Total                     0.1000                 0.0250
  + lower bound beta spending (under H1):
   Hwang-Shih-DeCani spending function with gamma = -2.
  ++ alpha spending:
   Hwang-Shih-DeCani spending function with gamma = -4.

  Boundary crossing probabilities and expected sample size
  assume any cross stops the trial

  Upper boundary (power or Type I Error)
            Analysis
     Theta      1      2      3  Total  E{N}
    0.0000 0.0013 0.0049 0.0171 0.0233 100.5
    0.2556 0.1412 0.4403 0.3185 0.9000 127.3

  Lower boundary (futility or Type II Error)
            Analysis
     Theta      1      2      3  Total
    0.0000 0.4056 0.4290 0.1420 0.9767
    0.2556 0.0148 0.0289 0.0563 0.1000
               T        n    Events HR futility HR efficacy
  IA 1  10.87225 191.5728  57.37729       1.065       0.452
  IA 2  16.39266 296.4606 114.75457       0.839       0.622
  Final 22.95897 307.2204 172.13186       0.737       0.737
  Accrual rates:
          Stratum 1 Stratum 2 Stratum 3
```

```
0-3             2         4          8
3-16.96         3         6         10
Control event rates (H1):
      Stratum 1 Stratum 2 Stratum 3
0-3        0.23      0.12      0.08
3-9        0.17      0.09      0.06
9-Inf      0.14      0.07      0.05
Censoring rates:
      Stratum 1 Stratum 2 Stratum 3
0-3           0         0         0
3-9           0         0         0
9-Inf         0         0         0
```

Chapter 4

Calendar-based design

We create the function `gsSurvCalendar()` as an extension to `gsSurv()` to set interim analysis timing at time of design to calendar times. We demonstrate how this can be used with either information-based or calendar-based spending. The coding is quite simple in that input enrollment rates, failure rates and dropout rates are used to compute expected events over time. Calendar timing of all analyses is specified by the user.

Now for `gsSurvCalendar()` which is actually a bit simpler and more flexible than `gsSurv()`. We swap in the argument `calendarTime` for the `timing` and `T` arguments above in `gsSurv()`; here the `timing` (information fraction) will be computed, while in `gsSurv()` the calendar times are computed. The parameter `hrbeta` is used when β-spending should be based on a different hazard ratio than specified in `hr` (only used when `test.type` is 3 or 4).

```
#' @param test.type
#'
#' @param alpha Type I error rate. Default is 0.025 since 1-sided testing is default.
#' @param sided 1 for 1-sided testing, 2 for 2-sided testing.
#' @param beta Type II error rate. Default is 0.10 (90% power); NULL if power is to be compu
#' @param astar Normally not specified. If \code{test.type=5} or \code{6}, \code{astar} spe
#' @param sfu A spending function or a character string indicating a boundary type (that is
#' @param sfupar Real value, default is -4 which is an O'Brien-Fleming-like conservative bou
#' @param sfl Specifies the spending function for lower boundary crossing probabilities whe
#' @param sflpar Real value, default is -2, which, with the default Hwang-Shih-DeCani spendi
#' @param calendarTime Vector of increasing positive numbers with calendar times of analyses
#' @param spending Select between calendar-based spending and information-based spending.
#' @param lambdaC scalar, vector or matrix of event hazard rates for the control group; rows
#' @param hr hazard ratio (experimental/control) under the alternate hypothesis (scalar).
#' @param hr0 hazard ratio (experimental/control) under the null hypothesis (scalar).
#' @param eta scalar, vector or matrix of dropout hazard rates for the control group; rows i
#' @param etaE matrix dropout hazard rates for the experimental group specified in like form
```

```
#' @param gamma a scalar, vector or matrix of rates of entry by time period (rows) and stra
#' @param R a scalar or vector of durations of time periods for recruitment rates specified
#' @param S a scalar or vector of durations of piecewise constant event rates specified in
#' @param minfup A non-negative scalar less than the maximum value in \code{calendarTime}.
#' @param ratio randomization ratio of experimental treatment divided by control; normally
#' @param r Integer value controlling grid for numerical integration as in Jennison and Tur
#'
#' @export
gsSurvCalendar <- function(
    test.type = 4, alpha = 0.025, sided = 1, beta = 0.1, astar = 0,
    sfu = gsDesign::sfHSD, sfupar = -4,
    sfl = gsDesign::sfHSD, sflpar = -2,
    calendarTime = c(12, 24, 36),
    spending = c("information", "calendar"),
    lambdaC = log(2) / 6, hr = .6, hr0 = 1, eta = 0, etaE = NULL,
    gamma = 1, R = 12, S = NULL, minfup = 18, ratio = 1,
    r = 18 # , tol = .Machine$double.eps^0.25
    ) {
  x <- nSurv(
    lambdaC = lambdaC, hr = hr, hr0 = hr0, eta = eta, etaE = etaE,
    gamma = gamma, R = R, S = S, T = max(calendarTime),
    minfup = minfup, ratio = ratio,
    alpha = alpha, beta = beta, sided = sided # , tol = tol
  )

  # Get interim expected event counts and sample size based on
  # input gamma, eta, lambdaC, R, S, minfup

  eDC <- NULL
  eDE <- NULL
  eNC <- NULL
  eNE <- NULL
  T <- NULL
  k <- length(calendarTime)
  for (i in 1:k) {
    xx <- nEventsIA(tIA = calendarTime[i], x = x, simple = FALSE)
    eDC <- rbind(eDC, xx$eDC)
    eDE <- rbind(eDE, xx$eDE)
    eNC <- rbind(eNC, xx$eNC)
    eNE <- rbind(eNE, xx$eNE)
  }
  timing <- rowSums(eDC) + rowSums(eNC)
  timing <- timing / max(timing)
  # if calendar spending, set usTime, lsTime
```

```
if (spending[1] == "calendar") {
  lsTime <- calendarTime / max(calendarTime)
} else {
  lsTime <- NULL
}
usTime <- lsTime

# Now inflate events to get targeted power
y <- gsDesign::gsDesign(
  k = k, test.type = test.type, alpha = alpha / sided,
  beta = beta, astar = astar, n.fix = x$d, timing = timing,
  sfu = sfu, sfupar = sfupar, sfl = sfl, sflpar = sflpar, # tol = tol,
  delta1 = log(hr), delta0 = log(hr0),
  usTime = usTime, lsTime = lsTime
)
y$hr <- hr
y$hr0 <- hr0
y$R <- x$R
y$S <- x$S
y$minfup <- x$minfup
# Inflate fixed design enrollment to get targeted events
inflate <- max(y$n.I) / x$d
y$gamma <- x$gamma * inflate
y$eDC <- inflate * eDC
y$eDE <- inflate * eDE
y$eNC <- inflate * eNC
y$eNE <- inflate * eNE
y$ratio <- ratio
y$lambdaC <- x$lambdaC
y$etaC <- x$etaC
y$etaE <- x$etaE
y$variable <- x$variable
# y$tol <- tol
y$T <- calendarTime
class(y) <- c("gsSurv", "gsDesign")

nameR <- nameperiod(cumsum(y$R))
stratnames <- paste("Stratum", 1:ncol(y$lambdaC))
if (is.null(y$S)) {
  nameS <- "0-Inf"
} else {
  nameS <- nameperiod(cumsum(c(y$S, Inf)))
}
rownames(y$lambdaC) <- nameS
```

```
  colnames(y$lambdaC) <- stratnames
  rownames(y$etaC) <- nameS
  colnames(y$etaC) <- stratnames
  rownames(y$etaE) <- nameS
  colnames(y$etaE) <- stratnames
  rownames(y$gamma) <- nameR
  colnames(y$gamma) <- stratnames
  return(y)
}
```

Example: default arguments

```
xxx <- gsSurvCalendar()
gsDesign::gsBoundSummary(xxx)
```

```
      Analysis              Value Efficacy Futility
     IA 1: 50%                  Z   2.7508   0.4687
        N: 132       p (1-sided)   0.0030   0.3197
    Events: 88      ~HR at bound   0.5551   0.9046
     Month: 12  P(Cross) if HR=1   0.0030   0.6803
           P(Cross) if HR=0.6   0.3621   0.0268
     IA 2: 92%                  Z   2.1282   1.8018
        N: 198       p (1-sided)   0.0167   0.0358
   Events: 161      ~HR at bound   0.7148   0.7526
     Month: 24  P(Cross) if HR=1   0.0175   0.9654
           P(Cross) if HR=0.6   0.8661   0.0827
        Final                  Z   2.0453   2.0453
        N: 198       p (1-sided)   0.0204   0.0204
   Events: 175      ~HR at bound   0.7339   0.7339
     Month: 36  P(Cross) if HR=1   0.0222   0.9778
           P(Cross) if HR=0.6   0.9000   0.1000
```

We check this with a call to `gsSurv()` that should give an identical design.

```
yyy <- gsSurv(
  k = 3, test.type = 4, alpha = 0.025, sided = 1, beta = 0.1, astar = 0,
  sfu = gsDesign::sfHSD, sfupar = -4,
  sfl = gsDesign::sfHSD, sflpar = -2,
  timing = xxx$timing,
  T = max(xxx$T),
  lambdaC = log(2) / 6, hr = .6, hr0 = 1, eta = 0, etaE = NULL,
  gamma = 1, R = 12, S = NULL, minfup = 18, ratio = 1,
  r = 18, tol = .Machine$double.eps^0.25
)
gsDesign::gsBoundSummary(yyy)
```

```
      Analysis              Value Efficacy Futility
```

```
    IA 1: 50%                 Z   2.7508   0.4686
       N: 182       p (1-sided)   0.0030   0.3197
  Events: 88      ~HR at bound   0.5550   0.9046
   Month: 17  P(Cross) if HR=1   0.0030   0.6803
          P(Cross) if HR=0.6   0.3621   0.0268
    IA 2: 92%                 Z   2.1282   1.8017
       N: 198       p (1-sided)   0.0167   0.0358
  Events: 161      ~HR at bound   0.7148   0.7526
   Month: 30  P(Cross) if HR=1   0.0175   0.9653
          P(Cross) if HR=0.6   0.8661   0.0827
       Final                  Z   2.0453   2.0453
       N: 198       p (1-sided)   0.0204   0.0204
  Events: 175      ~HR at bound   0.7339   0.7339
   Month: 36  P(Cross) if HR=1   0.0222   0.9778
          P(Cross) if HR=0.6   0.9000   0.1000
```

Example: default arguments, but information fraction

Note that calendar fraction spending below gives much more conservative interim bounds than information fraction spending above. This behavior is due to event accumulation slowing down as the trial proceeds; this is not always the case.

```
gsDesign::gsBoundSummary(gsSurvCalendar(spending = "calendar"))
```

```
    Analysis              Value Efficacy Futility
    IA 1: 50%                 Z   3.0107   0.1378
       N: 122       p (1-sided)   0.0013   0.4452
  Events: 82      ~HR at bound   0.5126   0.9699
   Month: 12  P(Cross) if HR=1   0.0013   0.5548
          P(Cross) if HR=0.6   0.2425   0.0148
    IA 2: 92%                 Z   2.5422   1.3488
       N: 184       p (1-sided)   0.0055   0.0887
  Events: 150      ~HR at bound   0.6597   0.8020
   Month: 24  P(Cross) if HR=1   0.0062   0.9156
          P(Cross) if HR=0.6   0.7278   0.0437
       Final                  Z   1.9682   1.9682
       N: 184       p (1-sided)   0.0245   0.0245
  Events: 163      ~HR at bound   0.7344   0.7344
   Month: 36  P(Cross) if HR=1   0.0244   0.9756
          P(Cross) if HR=0.6   0.9000   0.1000
```

Example: computing power rather than sample size

Here we set `beta = NULL` so that `gsSurvCalendar()` will compuute power rather than sample size. This needs additional testing. Right now it does not work as `gsDesign()` will not compute power with the given call. This

can possibly be fixed by using the `maxn.IPlan` argument when in the `gsSurvCalendar()` code.

```
gsDesign::gsBoundSummary(gsSurvCalendar(beta = NULL))
```

Example: stratified design

Stratified designs still need fixing of weights for strata. However, `gsSurvCalendar()` will generate an answer with inappropriate weights until that time.

```
gsSurvCalendar(
  alpha = .05, beta = .2, hr = 2 / 3, ratio = 1,
  R = 2, S = NULL, calendarTime = c(2, 4),
  minfup = 2,
  lambdaC = matrix(c(1, .8, .5), nrow = 1),
  eta = matrix(0, nrow = 1, ncol = 3),
  etaE = matrix(0, nrow = 1, ncol = 3),
  gamma = matrix(c(.4, .4, .2), nrow = 1)
)
```

```
Time to event group sequential design with HR= 0.6666667
Equal randomization:          ratio=1
Asymmetric two-sided group sequential design with
80 % power and 5 % Type I Error.
Upper bound spending computations assume
trial continues if lower bound is crossed.

               ----Lower bounds----  ----Upper bounds-----
  Analysis  N   Z   Nominal p Spend+  Z   Nominal p Spend++
         1 128 1.13    0.8703  0.122 2.02    0.0214  0.0214
         2 160 1.69    0.9544  0.078 1.69    0.0456  0.0286
     Total                     0.2000                0.0500
+ lower bound beta spending (under H1):
 Hwang-Shih-DeCani spending function with gamma = -2.
++ alpha spending:
 Hwang-Shih-DeCani spending function with gamma = -4.

Boundary crossing probabilities and expected sample size
assume any cross stops the trial

Upper boundary (power or Type I Error)
          Analysis
   Theta      1      2  Total  E{N}
  0.0000 0.0214 0.0248 0.0462 130.6
  0.2034 0.6056 0.1944 0.8000 136.0
```

```
Lower boundary (futility or Type II Error)
          Analysis
   Theta      1      2  Total
  0.0000 0.8703 0.0835 0.9538
  0.2034 0.1220 0.0780 0.2000
      T        n   Events HR futility HR efficacy
IA 1  2 191.3497 127.0654       0.819       0.698
Final 4 191.3497 159.9666       0.766       0.766
Accrual rates:
    Stratum 1 Stratum 2 Stratum 3
0-2     38.27     38.27     19.13
Control event rates (H1):
      Stratum 1 Stratum 2 Stratum 3
0-Inf         1       0.8       0.5
Censoring rates:
      Stratum 1 Stratum 2 Stratum 3
0-Inf         0         0         0
```

References

D. Bernstein and S. Lagakos. Sample size and power determination for stratified clinical trials. *Journal of Statistical Computation and Simulation*, 8: 65–73, 1978.

Kyungmann Kim and Anastasios A. Tsiatis. Study duration for clinical trials with survival response and early stopping rule. *Biometrics*, 46:81–92, 1990.

John M. Lachin and Mary A. Foulkes. Evaluation of sample size and power for analyses of survival with allowance for nonuniform patient entry, losses to follow-up, noncompliance, and stratification. *Biometrics*, 42:507–519, 1986.