Package 'safestats'

October 5, 2026

Type Package

Title Safe Anytime-Valid Inference

Version 0.8.9

Date 2026-10-02

Maintainer Alexander Ly <alexander.ly.nl@gmail.com>

Description Functions to design and apply tests that are anytime valid. The functions can be used to design hypothesis tests in the prospective/randomised control trial setting or in the observational/retrospective setting. The resulting tests remain valid under both optional stopping and optional continuation. The current version includes safe t-tests and safe tests of two proportions. For details on the theory of safe tests, see Ly, A, Boehm, Grunwald, Ramdas and van Ravenzwaaij (2024). ``Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values" (<doi:10.31234/osf.io/h5vae>) and Grunwald, de Heide and Koolen (2024) ``Safe Testing" (<doi:10.1093/jrsssb/qkae011>), for details on safe logrank tests see ter Schure, Perez-Ortiz, Ly and Grunwald (2024) ``The Anytime-Valid Logrank Test: Error Control under Continuous Monitoring with Unlimited Horizon" (<doi:10.51387/24-NEJSDS65>), and Turner, Ly and Grunwald (2024) ``Generic E-variables for exact sequential k-sample tests that allow for optional stopping" (<doi:10.1016/j.jspi.2023.106116>) for details on safe contingency table tests.

License LGPL (>= 3)

Encoding UTF-8

Depends R (>= 4.5)

Imports stats (>= 3.6), lamW (>= 2.2.1), hypergeo (>= 1.2-13), survival (>= 3.2-13), BiasedUrn (>= 1.07), boot (>= 1.3-28), dplyr (>= 1.0.6), purrr (>= 0.3.5), rlang (>= 1.0.6), Matrix (>= 1.7-3), graphics (>= 4.2.2)

Suggests testthat (>= 3.0.0), knitr (>= 1.49), rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Language en-GB

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Author Alexander Ly [cre, aut],
Rosanne Turner [aut],
Yongqi Wang [aut],
Muriel Felipe Perez-Ortiz [ctb],
Udo Boehm [ctb],
Judith ter Schure [ctb],
Peter Grunwald [ctb]

Repository CRAN

Date/Publication 2026-10-05 03:10:02 UTC

Contents

addCite . . . 5
ceil . . . 5
checkAndReturnEsMinParameterSide . . . 6
checkAndReturnNPlan . . . 7
checkAndReturnsEsMinParameterSide . . . 7
checkDoubleArgumentsDesignObject . . . 8
compute1F1AllVersions . . . 9
computeBetaBatchSafeZ . . . 10
computeBetaBootstrapper . . . 11
computeBetaSafeT . . . 12
computeBetaSafeZ . . . 14
computeBootObj . . . 16
computeConfidenceBoundForLogOddsTwoProportions . . . 17
computeConfidenceBoundsForDifferenceTwoProportions . . . 18
computeConfidenceInterval2x2LogOdds . . . 19
computeConfidenceInterval2x2PropDiff . . . 20
computeConfidenceIntervalT . . . 21
computeConfidenceIntervalZ . . . 22
computeConfidenceSequence2x2LogOdds . . . 24
computeConfidenceSequence2x2PropDiff . . . 25
computeConjugateCredibleIntervalTwoSampleT . . . 26
computeEsMinSavi2x2 . . . 27
computeLogrankBetaFrom . . . 29
computeLogrankNEvents . . . 31
computeLogrankZ . . . 33
computeMinEsBatchSafeZ . . . 34
computeMinEsBatchSaviT . . . 36
computeNEff . . . 37
computeNPlanBatchSafeZ . . . 38
computeNPlanBatchSaviT . . . 40
computeNPlanBootstrapper . . . 41
computeNPlanSafeT . . . 42

computeNPlanSafeZ . . . 44
computeNPlanSavi2x2 . . . 47
computePowerBootstrapper . . . 48
computePowerSavi2x2 . . . 49
computeStatsForLogrank . . . 50
computeZTSumStats . . . 51
conjugateBfTStat . . . 52
conjugateBfTStatOld . . . 54
constructSafeDesignObj . . . 55
constructSafeTestObj . . . 56
constructSampleStoppingTimesList . . . 57
defineTTestN . . . 57
designFreqT . . . 58
designFreqZ . . . 59
designSafe1aHelper . . . 61
designSafe2Helper . . . 62
designSafeLogrank . . . 63
designSafeLogrank2WantBeta . . . 66
designSafeT . . . 68
designSafeT1aWantNPlan . . . 72
designSafeTwoProportions . . . 74
designSafeZ . . . 77
designSafeZ1aWantNPlan . . . 81
designSafeZ2WantBeta . . . 83
designSafeZ3bWantParameter . . . 85
designSafeZ3WantEsMin . . . 87
designSavi2x2 . . . 88
designSaviT2WantPower . . . 92
designSaviT3bWantParameter . . . 94
designSaviT3WantEsMin . . . 95
extractNameFromArgs . . . 96
fnchLogPartition . . . 97
generateNormalData . . . 97
generateSurvData . . . 98
getArgs . . . 100
getNameAlternative . . . 100
getNameTestType . . . 101
isTryError . . . 101
lcosh . . . 102
logEValueVec2x2LogOddsEGauss . . . 102
logEValueVec2x2LogOddsGrow . . . 103
logEValueVec2x2PropDiffEBeta . . . 104
logEValueVec2x2PropDiffGrow . . . 105
logLikelihoodFNCH . . . 106
logrankSingleEExact . . . 107
logrankSingleZ . . . 108
makeRunningIntersection . . . 109
matchAlphaRelevanceWith . . . 110

matchEParameterWith . . . 111
matchEsMinWith . . . 112
matchPowerWith . . . 113
matchRelevanceParameterWith . . . 113
plot.safe2x2Sim . . . 114
plot.safeDesign . . . 115
plot.safeTest . . . 117
plotConfidenceSequenceTwoProportions . . . 120
plotHistogramDistributionStoppingTimes . . . 121
predictiveThetas2x2 . . . 122
print.safe2x2Sim . . . 123
print.safeDesign . . . 124
print.safeTest . . . 124
pValueFromZStat . . . 125
returnOne . . . 127
rLogrank . . . 127
safeLogrankTest . . . 128
safeTTestStat . . . 131
safeTwoProportionsTest . . . 135
safeZTest . . . 138
safeZTestStat . . . 141
sampleLogrankStoppingTimes . . . 143
sampleStoppingTimesSafeT . . . 145
sampleStoppingTimesSafeZ . . . 148
sampleStoppingTimesSavi2x2 . . . 150
savi2x2TestStat . . . 152
savi2x2TestStatUmp . . . 154
saviAnovaOneB . . . 155
saviAnovaStat . . . 156
saviTTestStat . . . 156
selectivelyContinueZOrTTestData . . . 160
setLowAndHighEsTrueZ . . . 162
setSafeStatsPlotOptionsAndReturnOldOnes . . . 163
simulateCoverageDifferenceTwoProportions . . . 163
simulateIncorrectStoppingTimesFisher . . . 164
simulateOptionalStoppingScenarioTwoProportions . . . 165
simulateTwoProportions . . . 166
solveRIPr2x2PropDiff . . . 168
solveUmpLogOdds . . . 169
solveWorstCaseTheta2x2PropDiff . . . 170
subjectiveBfZStat . . . 171
tryOrFailWithNA . . . 172
tTestWidthDerivative . . . 172
zMarg . . . 173

Index **174**

addCite *Add a savi reference*

Description

Keys are of the form firstAuthorYearFirstWord

Usage

```
addCite(..., breakLine = TRUE)
```

Arguments

...	further arguments to be passed to or from methods.
breakLine	logical, default TRUE to break up the references

Value

a character string

Examples

```
addCite(grunwald2024safe)
```

ceil *A ceiling function with a tolerance at the 13th digit*

Description

A ceiling function with a tolerance at the 13th digit

Usage

```
ceil(x, digits = 13)
```

Arguments

x	numeric, that needs
digits	integer, position of the digit to round of to

Value

integer

Examples

```
ceiling(27/21*21)
ceil(27/21*21)
```

checkAndReturnEsMinParameterSide *Checks consistency between the sided of the hypothesis and the minimal clinically relevant effect size or savi test defining parameter. Throws an error if the one-sided hypothesis is incongruent with the*

Description

Checks consistency between the sided of the hypothesis and the minimal clinically relevant effect size or savi test defining parameter. Throws an error if the one-sided hypothesis is incongruent with the

Usage

```
checkAndReturnEsMinParameterSide(
  paramToCheck,
  alternative = c("twoSided", "greater", "less"),
 esMinName = c("noName", "meanDiffMin", "meanDiffTrue", "phiS", "deltaMin", "deltaS",
   "hrMin", "thetaS", "deltaTrue", "g", "kappaG", "propDiffMin", "logOddsMin"),
  paramDomain = NULL
)
```

Arguments

paramToCheck	numeric. Either a named savi test defining parameter such as phiS, or thetaS, or a minimal clinically relevant effect size called with a non-null esMinName name
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
esMinName	provides the name of the effect size. Either "meanDiffMin" for the z-test, "deltaMin" for the t-test, "hrMin" for the logrank test, or "propDiffMin" and "logOddsMin" for the 2x2 test (both A minus B)
paramDomain	Domain of the paramToCheck, typically, positiveNumbers. Default NULL

Value

paramToCheck after checking, perhaps with a change in sign

checkAndReturnNPlan *Check consistency between nPlan and the testType for one and two-sample z and t-tests*

Description

Check consistency between nPlan and the testType for one and two-sample z and t-tests

Usage

```
checkAndReturnNPlan(
  nPlan,
  ratio = 1,
  testType = c("oneSample", "paired", "twoSample")
)
```

Arguments

nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
testType	either one of "oneSample", "paired", "twoSample".

Value

nPlan a vector of sample sizes of length 1 or 2

checkAndReturnsEsMinParameterSide *Checks consistency between the sided of the hypothesis and the minimal clinically relevant effect size or safe test defining parameter. Throws an error if the one-sided hypothesis is incongruent with the*

Description

Checks consistency between the sided of the hypothesis and the minimal clinically relevant effect size or safe test defining parameter. Throws an error if the one-sided hypothesis is incongruent with the

Usage

```
checkAndReturnsEsMinParameterSide(
  paramToCheck,
  alternative = c("twoSided", "greater", "less"),
 esMinName = c("noName", "meanDiffMin", "phiS", "deltaMin", "deltaS", "hrMin", "thetaS",
    "deltaTrue"),
  paramDomain = NULL
)
```

Arguments

paramToCheck	numeric. Either a named safe test defining parameter such as phiS, or thetaS, or a minimal clinically relevant effect size called with a non-null esMinName name
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
esMinName	provides the name of the effect size. Either "meanDiffMin" for the z-test, "deltaMin" for the t-test, or "hrMin" for the logrank test
paramDomain	Domain of the paramToCheck, typically, positiveNumbers. Default NULL

Value

paramToCheck after checking, perhaps with a change in sign

---

checkDoubleArgumentsDesignObject

Helper function to check whether arguments are specified in a function at a higher level and already provided in the design object.

Description

Helper function to check whether arguments are specified in a function at a higher level and already provided in the design object.

Usage

```
checkDoubleArgumentsDesignObject(designObj, ...)
```

Arguments

designObj	an object of class "safeDesign".
...	arguments that need checking.

Value

Returns nothing only used for its side-effects to produces warnings if needed.

Examples

```
designObj <- designSaviZ(0.4)

checkDoubleArgumentsDesignObject(designObj, "alpha"=NULL, alternative=NULL)
# Throws a warning
checkDoubleArgumentsDesignObject(designObj, "alpha"=0.4, alternative="d")
```

`compute1F1AllVersions` *Computes 6 equivalent forms of the confluent hypergeometric function 1f1*

Description

Repeated calls of genhypergeo, which provides further details.

Usage

```
compute1F1AllVersions(U, L, z, tol = 0, maxiter = 2000)
```

Arguments

`U`	Upper arguments respectively (real or complex)
`L`	Lower arguments respectively (real or complex)
`z`	Primary complex argument
`tol`	tolerance with default zero meaning to iterate until additional terms to not change the partial sum
`maxiter`	Maximum number of iterations to perform

Value

a vector

Examples

```
compute1F1AllVersions(U=-359, L=1/2, z=-0.1891234)
```

`computeBetaBatchSafeZ` *Helper function: Computes 1-power based on meanDiffTrue and nPlan*

Description

Helper function: Computes 1-power based on meanDiffTrue and nPlan

Usage

```
computeBetaBatchSaviZ(
  meanDiffTrue,
  nPlan,
  alpha = 0.05,
  sigma = 1,
  kappa = sigma,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow")
)
```

Arguments

`meanDiffTrue`	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
`nPlan`	optional numeric vector of length at most 2, see scenario 2 and 3 above.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`sigma`	numeric > 0 representing the assumed population standard deviation used to scale the data.
`kappa`	the true population standard deviation. Default kappa=sigma.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`testType`	either one of "oneSample", "paired", "twoSample".
`parameter`	numeric, an optional savi test defining parameter. Default set to `NULL`. and adapts to meanDiffMin and eType, see `matchEParameterWith` for details.
`eType`	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.

Value

numeric that represents the type II error

Functions

- `computeBetaBatchSafeZ()`: Deprecated version of computeBetaBatchSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(5), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

---

computeBetaBootstrapper

Helper function to compute uncertainty regarding nPlan estimates

---

Description

Helper function to compute uncertainty regarding nPlan estimates

Usage

```
computeBetaBootstrapper(samplingResult, parameter, nPlan, nBoot)
```

Arguments

`samplingResult`	output from sampling functions such as computeNPlanSaviZ and computeNPlanSaviT
`parameter`	numeric > 0, the savi test defining parameter.
`nPlan`	vector of max length 2 representing the planned sample sizes.
`nBoot`	integer > 0 representing the number of bootstrap samples to estimate the uncertainty of various estimates.

Value

list with bootstrap objects

Examples

```
samplingResult <- sampleStoppingTimesSaviT(0.7, nSim=10, nMax=20)
result <- computeNPlanBootstrapper(samplingResult, 0.7, 0.2, 20, nBoot=1e2)
```

computeBetaSafeT *Helper function: Computes the power of the saviTTest based on deltaMin and nPlan*

Description

Helper function: Computes the power of the saviTTest based on deltaMin and nPlan

Usage

```
computePowerSaviT(
  deltaTrue,
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  deltaMin = deltaTrue,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  wantSamplePaths = TRUE,
  nuMin = 2,
  wantSimData = TRUE,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  ...
)
```

Arguments

deltaTrue	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses

	point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi t test under continuous monitoring,or for the computation of the logarithm of the implied target.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.
deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.
wantSimData	logical, if TRUE then also output the simulated data
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.

Value

a list which contains at least power and an adapted bootObject of class `boot()`.

Functions

- `computeBetaSafeT()`: Deprecated version of computeBetaSaviT

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computePowerSaviT(deltaTrue=0.7, 27, nSim=10)
```

computeBetaSafeZ *Helper function: Computes the power based on meanDiffTrue and nPlan*

Description

Helper function: Computes the power based on meanDiffTrue and nPlan

Usage

```
computePowerSaviZ(
  meanDiffTrue,
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  sigma = 1,
  kappa = sigma,
  meanDiffMin = meanDiffTrue,
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  ...
)
```

Arguments

meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.

testType	either one of "oneSample", "paired", "twoSample".
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi z test under continuous monitoring,or for the computation of the logarithm of the implied target.
...	further arguments to be passed to or from methods.
meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
wantSimData	logical, if TRUE then also output the simulated data
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.

Value

a list which contains at least power and an adapted bootObject of class boot.

Functions

- computeBetaSafeZ(): Deprecated version of computeBetaSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computePowerSaviZ(meanDiffTrue=0.7, 20, nSim=10)
```

`computeBootObj` *Computes the bootObj for sequential sampling procedures regarding nPlan, beta, the implied target*

Description

Computes the bootObj for sequential sampling procedures regarding nPlan, beta, the implied target

Usage

```
computeBootObj(
  values,
  power = NULL,
  nPlan = NULL,
  nBoot = 1000L,
  alpha = NULL,
  beta = NULL,
 objType = c("nPlan", "nMean", "power", "beta", "betaFromEValues", "logImpliedTarget",
    "expectedStopTime")
)
```

Arguments

`values`	numeric vector. If objType equals "nPlan" or "beta" then values should be stopping times, if objType equals "logImpliedTarget" then values should be eValues.
`power`	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
`nPlan`	numeric vector of length at most 2 representing the planned sample size(s).
`nBoot`	integer > 0 representing the number of bootstrap samples to estimate the uncertainty of various estimates.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`beta`	numerical in (0,1). Old parameter now replaced by the power parameter
`objType`	character string either "nPlan", "nMean", "beta", "betaFromEValues", "expectedStopTime" or "logImpliedTarget".

Value

bootObj

Examples

```
computeBootObj(1:100, objType="nPlan", beta=0.3)
```

`computeConfidenceBoundForLogOddsTwoProportions`
Estimate an upper or lower bound for a savi confidence sequence on the logarithm of the odds ratio for two proportions.

Description

Estimate an upper or lower bound for a savi confidence sequence on the logarithm of the odds ratio for two proportions.

Usage

```
computeConfidenceBoundForLogOddsTwoProportions(
  ya,
  yb,
  saviDesign,
  bound = c("lower", "upper"),
  deltaStart,
  deltaStop,
  precision
)
```

Arguments

`ya`	positive observations/ events per data block in group a: a numeric with integer values between (and including) 0 and `na`, the number of observations in group a per block.
`yb`	positive observations/ events per data block in group b: a numeric with integer values between (and including) 0 and `nb`, the number of observations in group b per block.
`saviDesign`	a 'saviDesign' object obtained through `designSaviTwoProportions`
`bound`	type of bound to calculate; "lower" to get a lower bound on positive delta, "upper" to get an upper bound on negative delta.
`deltaStart`	starting value of the grid to search over for the bound on the confidence sequence (in practice: the interval). Numeric >0 when searching for a lower bound, numeric < 0 when searching for an upper bound.
`deltaStop`	end value of the grid to search over for the bound on the confidence sequence (in practice: the interval). Numeric >0 when searching for a lower bound, numeric < 0 when searching for an upper bound.
`precision`	precision of the grid between deltaStart and deltaStop.

Value

numeric: the established lower- or upper bound on the logarithm of the odds ratio between the groups

Examples

```
balancedSaviDesign <- designSaviTwoProportions(na = 1,
                                               nb = 1,
                                               nBlocksPlan = 10,
                                               alpha = 0.05)
#hypothesize OR < 1 (i.e., log OR < 0)
ya <- c(1,1,1,1,1,1,1,1,0,1)
yb <- c(0,0,0,0,1,0,0,0,0,0)
#one-sided CI for OR-, establish upper bound on log odds ratio
computeConfidenceBoundForLogOddsTwoProportions(ya = ya,
                                           yb = yb,
                                           saviDesign = balancedSaviDesign,
                                           bound = "upper",
                                           deltaStart = -0.01,
                                           deltaStop = -4,
                                           precision = 20)
```

---

computeConfidenceBoundsForDifferenceTwoProportions

Estimate Lower and Upper Bounds on the Confidence Sequence (Interval) for the Difference Divergence Measure for Two Proportions

---

Description

Estimate Lower and Upper Bounds on the Confidence Sequence (Interval) for the Difference Divergence Measure for Two Proportions

Usage

```
computeConfidenceBoundsForDifferenceTwoProportions(
  ya,
  yb,
  precision,
  saviDesign
)
```

Arguments

ya	positive observations/ events per data block in group a: a numeric with integer values between (and including) 0 and na, the number of observations in group a per block.

yb	positive observations/ events per data block in group b: a numeric with integer values between (and including) 0 and nb, the number of observations in group b per block.
precision	precision of the grid of differences to search over for the lower and upper bounds.
saviDesign	a 'saviDesign' object obtained through designSaviTwoProportions

Value

list with found lower and upper bound.

Examples

```
balancedSaviDesign <- designSaviTwoProportions(na = 1,
                                               nb = 1,
                                               nBlocksPlan = 10,
                                               alpha = 0.05)
ya <- c(1,1,1,1,1,1,1,1,0,1)
yb <- c(0,0,0,0,1,0,0,0,0,0)
computeConfidenceBoundsForDifferenceTwoProportions(ya = ya,
                                               yb = yb,
                                               precision = 20,
                                               saviDesign = balancedSaviDesign)
```

---

computeConfidenceInterval2x2LogOdds

Helper function: Computes the savi confidence interval for logOdds

---

Description

Inverts the eGauss numerator: the interval is the set of logOdds values at which the numerator over the conditional likelihood of logLikelihoodFNCH() stays below 1 / alpha, found by one minimum and two roots on domain.

Usage

```
computeConfidenceInterval2x2LogOdds(
  ya,
  yb,
  na,
  nb,
  logPTotal,
  alpha,
  domain = c(-40, 40)
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
logPTotal	numeric, the eGauss log numerator on all blocks: the plain cumulative log e-value plus the cumulative hypergeometric log likelihood.
alpha	numeric in (0, 1), one minus the confidence level.
domain	c(lower, upper), the candidates searched.

Value

numeric vector that contains the lower and upper bound of the savi confidence interval, named lowerBound, upperBound; NA with a warning for a bound that lies beyond domain, and the whole domain with a warning when the set is empty.

---

computeConfidenceInterval2x2PropDiff

Helper function: Computes the savi confidence interval for propDiff

---

Description

Inverts the eBeta e-process: the predictive Beta numerator of predictiveThetas2x2() against, per block, the reverse information projection of solveRIPr2x2PropDiff() onto each candidate propDiff. The interval is the set of candidates whose e-value stays below 1 / alpha, found by one minimum and two roots on domain.

Usage

```
computeConfidenceInterval2x2PropDiff(
  ya,
  yb,
  na,
  nb,
  betaParameter,
  alpha,
  domain = c(-1, 1)
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
betaParameter	list(betaA1, betaA2, betaB1, betaB2), the Beta prior shapes on thetaA and thetaB for "eBeta", each a single finite positive number; NULL means 1 / (2 * na) and 1 / (2 * nb), taking block 1's sizes with a warning when they vary by block. A given prior keeps its own block 1: the test does not replace it by the UMP e-value. For "grow" on propDiff, the test puts Beta(betaA1, betaA2) on thetaA rescaled to its feasible interval on the curve, B's shapes unused, and NULL means uniform; planning ignores it.
alpha	numeric in (0, 1), one minus the confidence level.
domain	c(lower, upper), the candidates searched.

Value

numeric vector that contains the lower and upper bound of the savi confidence interval, named lowerBound, upperBound; the domain edge when that edge is still inside, and c(-1, 1) with a warning when the set is empty.

---

computeConfidenceIntervalT

Helper function: Computes the savi confidence sequence for the mean in a T-test

---

Description

Helper function: Computes the savi confidence sequence for the mean in a T-test

Usage

```
computeConfidenceIntervalT(
  meanObs,
  sdObs,
  nEff,
  nu,
  parameter,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  alternative = c("twoSided", "greater", "less"),
  ciValue = 0.95,
```

```
  maxRoot = 11,
  nuMin = 2
)
```

Arguments

meanObs	numeric, the observed mean. For two sample tests this is difference of the means.
sdObs	numeric, the observed standard deviation. For a two-sample test this is the root of the pooled variance.
nEff	numeric > 0, the effective sample size. For one sample tests, this is just n.
nu	numeric > 0, the degrees of freedom.
parameter	numeric > 0, the savi test defining parameter, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha
maxRoot	Used to bound the candidate set of width of the confidence interval/
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.

Value

numeric vector that contains the upper and lower bound of the savi confidence sequence

Examples

```
computeConfidenceIntervalT(meanObs=0.3, sdObs=2, nEff=12, nu=11, parameter=0.4)
```

---

computeConfidenceIntervalZ

Helper function: Computes the savi confidence sequence for a Z-test

---

Description

Helper function: Computes the savi confidence sequence for a Z-test

Usage

```
computeConfidenceIntervalZ(
  nEff,
  meanObs,
  parameter,
  sigma = 1,
  ciValue = 0.95,
  alternative = "twoSided",
  a = NULL,
  g = NULL,
 eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "freq", "credibleInterval"),
  maxRoot = 20
)
```

Arguments

nEff	numeric > 0, the effective sample size.
meanObs	numeric, the observed mean. For two sample tests this is difference of the means.
parameter	numeric > 0, the savi test defining parameter, see matchEParameterWith for details.
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
a	numeric, only used for eType="credibleInterval". a specifies the centre of the Gaussian prior on population mean.
g	numeric > 0, only used for eType="credibleInterval". g specifies the variance of the Gaussian prior on population mean.
eType	character string, one of "eCauchy", "eGauss", "grow", "freq", "credibleInterval". This is somewhat a misnomer as "freq" and "credibleInterval" do not correspond to e-value tests. "eCauchy" yields an anytime-valid confidence interval based on a Cauchy mixture, whereas "eGauss" yields an anytime-valid confidence interval based on a Gaussian mixture. "grow" yields an anytime-valid confidence interval based on a mixture of point masses at the minimal clinically relevant mean difference. This confidence interval unfortunately does not shrink as the sample size tends to infinity. "freq" yields the standard unsafe frequentist confidence interval, which is not safe. "credibleInterval" yields the Bayesian credible interval based on a conjugate prior as is usual in Bayesian analysis. This interval is also not safe.
maxRoot	Used to bound the candidate set of width of the confidence interval, whenever eType="eCauchy"

Value

numeric vector that contains the upper and lower bound of the savi confidence sequence

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computeConfidenceIntervalZ(nEff=15, meanObs=0.3, parameter=0.2)
```

`computeConfidenceSequence2x2LogOdds`

Helper function: Computes the savi confidence sequence for logOdds

---

Description

Row `i` is the interval of `computeConfidenceInterval2x2LogOdds()` on blocks `1..i`. The eGauss numerator at block `i` is that of the test on blocks `1..i`, as every factor is fixed given its block's total or predictable, so the test's numerator vector serves all rows; the root finding is repeated per prefix.

Usage

```
computeConfidenceSequence2x2LogOdds(
  ya,
  yb,
  na,
  nb,
  logNumerator,
  alpha,
  runningIntersection
)
```

Arguments

`ya`	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and `na`, the number of observations in group A per block.
`yb`	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and `nb`, the number of observations in group B per block.
`na`	number of observations in group A per data block, one value or one per block.

nb	number of observations in group B per data block, one value or one per block.
logNumerator	numeric vector, the eGauss log numerator on blocks 1..i for element i: the plain cumulative log e-process, before the UMP replacement of block 1, plus the cumulative hypergeometric log likelihood.
alpha	numeric in (0, 1), one minus the confidence level.
runningIntersection	logical, if TRUE then intersect each row of the blockwise confidence sequence with the previous one; NULL keeps the constructor's FALSE.

Value

an nBlocks x 2 matrix that contains the lower and upper bound of the savi confidence sequence per block; an empty set is an NA row, and with runningIntersection the rows stay NA after the first empty row.

---

computeConfidenceSequence2x2PropDiff

Helper function: Computes the savi confidence sequence for propDiff

---

Description

Row i is the interval of computeConfidenceInterval2x2PropDiff() on blocks 1..i, found by advancing one e-process per grid candidate and reading off the run of candidates below 1 / alpha, each bound refined outward by a secant.

Usage

```
computeConfidenceSequence2x2PropDiff(
  ya,
  yb,
  na,
  nb,
  betaParameter,
  alpha,
  runningIntersection,
  nGrid = 2000L
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.

na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
betaParameter	list(betaA1, betaA2, betaB1, betaB2), the Beta prior shapes on thetaA and thetaB for "eBeta", each a single finite positive number; NULL means 1 / (2 * na) and 1 / (2 * nb), taking block 1's sizes with a warning when they vary by block. A given prior keeps its own block 1: the test does not replace it by the UMP e-value. For "grow" on propDiff, the test puts Beta(betaA1, betaA2) on thetaA rescaled to its feasible interval on the curve, B's shapes unused, and NULL means uniform; planning ignores it.
alpha	numeric in (0, 1), one minus the confidence level.
runningIntersection	logical, if TRUE then intersect each row of the blockwise confidence sequence with the previous one; NULL keeps the constructor's FALSE.
nGrid	integer > 0, the number of candidates strictly inside (-1, 1).

Value

an nBlocks x 2 matrix that contains the lower and upper bound of the savi confidence sequence per block; an empty set is an NA row, and with runningIntersection the rows are nested and stay NA after the first empty row.

`computeConjugateCredibleIntervalTwoSampleT` *Computes the credible interval of a two-sample t-test based on conjugate priors*

Description

Computes the credible interval of a two-sample t-test based on conjugate priors

Usage

```
computeConjugateCredibleIntervalTwoSampleT(
  x1,
  sdObs1,
  n1,
  x2,
  sdObs2,
  n2,
  a1 = 3.98,
  g1 = 0.03,
  a2 = 4.02,
  g2 = 0.05,
  aGamma = 2,
  bGamma = 1/2,
  ciValue = 0.95
)
```

Arguments

x1	numeric, sample mean of group 1
sdObs1	numeric, the observed standard deviation of the first group.
n1	integer sample size of group 1
x2	numeric, sample mean of group 2
sdObs2	numeric, the observed standard deviation of the second group.
n2	integer sample size of group 2
a1	numeric, prior mean of the population mean mu1 of group 1
g1	numeric > 0, conditional prior variance of the population mean mu1 of group 1 is given by g1*sigma^2
a2	numeric, prior mean of the population mean mu2 of group 2
g2	numeric > 0, conditional prior variance of the population mean mu2 of group 2 is given by g2*sigma^2
aGamma	numeric > 0, shape parameter of the prior on the standard deviation sigma
bGamma	numeric > 0, rate parameter of the prior on the standard deviation sigma
ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha

Value

a vector of length two representing the credible interval

References

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computeConjugateCredibleIntervalTwoSampleT(1, 1, 3, 1, 1, 3)
```

---

computeEsMinSavi2x2 *Computes the smallest detectable propDiffMin with power probability, for the provided number of blocks*

Description

The smallest propDiffMin at which the worst-case power of computePowerSavi2x2() at nBlocks = nBlocksPlan reaches power, found by stats::uniroot() on propDiffBounds. Every candidate is simulated with the same seed, so the target is a deterministic step function of the candidate. There is no planning on logOdds.

Usage

```
computeEsMinSavi2x2(
  na,
  nb,
  nBlocksPlan,
  power = 0.8,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  nSim = 1000L,
  seed = NULL,
  pb = TRUE,
  propDiffBounds = c(0.01, 0.9),
  tol = 1e-05
)
```

Arguments

na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
nBlocksPlan	planned number of data blocks collected, see scenario 2 and 3 above.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi 2x2 test under continuous monitoring.
seed	integer, seed number. Default seed=NULL yields seed=2026.
pb	logical, if TRUE, then show progress bar.
propDiffBounds	search interval for the magnitude of propDiffMin, strictly inside (0, 1).
tol	tolerance of the root on the propDiff scale.

Value

numeric that represents the minimal detectable difference in proportions, negative for "less", or NA when not found.

computeLogrankBetaFrom

Helper function: Computes the type II error under optional stopping based on the minimal clinically relevant hazard ratio and the maximum number of nEvents.

Description

Helper function: Computes the type II error under optional stopping based on the minimal clinically relevant hazard ratio and the maximum number of nEvents.

Usage

```
computeLogrankBetaFrom(
  hrMin,
  nEvents,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  m0 = 50000L,
  m1 = 50000L,
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  groupSizePerTimeFunction = returnOne,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  ...
)
```

Arguments

hrMin	numeric that defines the minimal relevant hazard ratio, the smallest hazard ratio that we want to detect.
nEvents	numeric > 0, targetted number of events.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis, which must be one of "twoSided" (default),"greater" or "less". The alternative is pitted against the null hypothesis of equality of the survival distributions. More specifically, let lambda1 be the hazard rate of group 1 (i.e., placebo), and lambda2 the hazard ratio of group 2 (i.e., treatment), then the null hypothesis states that the hazard

	ratio theta = lambda2/lambda1 = 1. If alternative = "less", the null hypothesis is compared to theta < 1, thus, lambda2 < lambda1, that is, the hazard of group 2 (i.e., treatment) is less than that of group 1 (i.e., placebo), hence, the treatment is beneficial. If alternative = "greater", then the null hypothesis is compared to theta > 1, thus, lambda2 > lambda1, hence, harm.
`m0`	Number of subjects in the control group 0/1 at the beginning of the trial, i.e., `nPlan[1]`.
`m1`	Number of subjects in the treatment group 1/2 at the beginning of the trial, i.e., `nPlan[2]`.
`testType`	either one of "oneSample", "paired", "twoSample".
`ratio`	numeric > 0 representing the randomisation ratio of condition 2 (Treatment) over condition 1 (Placebo), thus, m1/m0. Note that m1 and m0 are not used to specify ratio. Ratio is only used when `zApprox=TRUE`, which ignores m1 and m0.
`parameter`	Numeric > 0, represents the savi tests defining thetaS. Default NULL so it's decided by the algorithm, typically, this equals hrMin, which corresponds to the GROW choice.
`eType`	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
`wantSamplePaths`	logical, if `TRUE` then also outputs the sample paths.
`groupSizePerTimeFunction`	A function without parameters and integer output. This function provides the number of events at each time step. For instance, if `rpois(1, 7)` leads to a random number of events at each time step.
`pb`	logical, if `TRUE`, then show progress bar.
`seed`	integer, seed number.
`nSim`	integer > 0, the number of simulations needed to compute power or the number of events for the exact savi logrank test under continuous monitoring
`nBoot`	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximation of power or nEvents for the exact savi logrank test under continuous monitoring
`relevanceTest`	logical, if `TRUE` then impose rule to stop for minimal efficiency if e <= alphaRelevance. Default `FALSE`.
`...`	further arguments to be passed to or from methods.

Value

a list which contains at least beta and an adapted bootObject of class `boot`.

Author(s)

Muriel Felipe Perez-Ortiz and Alexander Ly

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
computeLogrankBetaFrom(hrMin=0.7, 300, nSim=10)
```

`computeLogrankNEvents` *Helper function: Computes the planned sample size based on the minimal clinical relevant hazard ratio, alpha and beta under optional stopping.*

Description

Helper function: Computes the planned sample size based on the minimal clinical relevant hazard ratio, alpha and beta under optional stopping.

Usage

```
computeLogrankNEvents(
  hrMin,
  beta,
  m0 = 50000,
  m1 = 50000,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  nSim = 1000L,
  nBoot = nSim,
  groupSizePerTimeFunction = returnOne,
  nMax = 1000L,
  parameter = NULL,
  digits = getOption("digits"),
  pb = TRUE
)
```

Arguments

`hrMin`	numeric that defines the minimal relevant hazard ratio, the smallest hazard ratio that we want to detect.
`beta`	numerical in (0,1). Old parameter now replaced by the power parameter
`m0`	Number of subjects in the control group 0/1 at the beginning of the trial, i.e., `nPlan[1]`.

m1	Number of subjects in the treatment group 1/2 at the beginning of the trial, i.e., nPlan[2].
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis, which must be one of "twoSided" (default),"greater" or "less". The alternative is pitted against the null hypothesis of equality of the survival distributions. More specifically, let lambda1 be the hazard rate of group 1 (i.e., placebo), and lambda2 the hazard ratio of group 2 (i.e., treatment), then the null hypothesis states that the hazard ratio theta = lambda2/lambda1 = 1. If alternative = "less", the null hypothesis is compared to theta < 1, thus, lambda2 < lambda1, that is, the hazard of group 2 (i.e., treatment) is less than that of group 1 (i.e., placebo), hence, the treatment is beneficial. If alternative = "greater", then the null hypothesis is compared to theta > 1, thus, lambda2 > lambda1, hence, harm.
nSim	integer > 0, the number of simulations needed to compute power or the number of events for the exact savi logrank test under continuous monitoring
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximation of power or nEvents for the exact savi logrank test under continuous monitoring
groupSizePerTimeFunction	A function without parameters and integer output. This function provides the number of events at each time step. For instance, if rpois(1, 7) leads to a random number of events at each time step.
nMax	An integer. Once nEvents hits nMax the experiment terminates, if it didn't stop due to threshold crossing crossing already. Default set to Inf.
parameter	Numeric > 0, represents the savi tests defining thetaS. Default NULL so it's decided by the algorithm, typically, this equals hrMin, which corresponds to the GROW choice.
digits	number of significant digits to be used.
pb	logical, if TRUE, then show progress bar.

Value

a list which contains at least nEvents and an adapted bootObject of class boot.

Author(s)

Muriel Felipe Perez-Ortiz and Alexander Ly

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
computeLogrankNEvents(0.7, 0.2, nSim=10)
```

`computeLogrankZ` *Helper function to computes the logrank statistic for 'Surv' objects of type "right" and "counting" with the hypergeometric variance.*

Description

This function was created to complement `survdiff` from the 'survival' package, which is restricted to 'Surv' objects of type "right". Most likely `survdiff` is much faster

Usage

```
computeLogrankZ(
  survObj,
  group,
  computeZ = TRUE,
  computeExactE = FALSE,
  theta0 = 1,
  thetaS = NULL,
  ...
)
```

Arguments

`survObj`	a Surv object that is either of type
`group`	a grouping factor with 2 levels
`computeZ`	logical. If `TRUE` computes the logrank z-statistic. Default is `TRUE`.
`computeExactE`	logical. If `TRUE` computes one-sided exact logrank e-value. Default is `FALSE`.
`theta0`	numeric > 0 used only for the e-value, i.e., if computeExactE is `TRUE`. Default is 1.
`thetaS`	numeric > 0 used only for the e-value, i.e., if computeExactE is `TRUE`. Default is NULL.
`...`	further arguments to be passed to or from methods.

Value

Returns a list containing at least the following components:

nEvents the number of events.

z the observed logrank statistic.

oMinEVector vector of observed minus expected.

varVector vector of hypergeometric variances.

stopTimeVector vector at which the events occurred.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Schoenfeld, D. (1981). The asymptotic properties of nonparametric tests for comparing survival distributions. *Biometrika*, **68**(*1*), 316-319, https://doi.org/10.2307/2335833.

Examples

```
data <- generateSurvData(nP = 5,
                         nT = 5,
                         lambdaP = 0.03943723,
                         lambdaT = 0.5*0.03943723,
                         endTime = 40,
                         seed = 2006)

survObj <- survival::Surv(data$time, data$status)

survObj <- survival::Surv(data$time, data$status)

result <- computeLogrankZ(survObj, data$group)
result$z
sqrt(survival::survdiff(survObj~data$group)$chisq)
```

---

computeMinEsBatchSafeZ

Computes the meanDiffMin that is detectable with probability "power" for given nPlan

---

Description

Computes the meanDiffMin that is detectable with probability "power" for given nPlan

Usage

```
computeMinEsBatchSaviZ(
  nPlan,
  alpha = 0.05,
  power = 0.8,
  sigma = 1,
  kappa = sigma,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  beta = NULL,
```

```
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  lowEsTrue = 0.01,
  highEsTrue = 0.002,
  ...
)
```

Arguments

nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
lowEsTrue	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
highEsTrue	numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
...	further arguments to be passed to or from methods.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.

Value

numeric > 0 that represents the minimal detectable mean difference

Functions

- computeMinEsBatchSafeZ(): Deprecated version of computeMinEsBatchSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computeMinEsBatchSaviZ(27)
```

`computeMinEsBatchSaviT` *Computes the smallest detectable deltaMin with power probability, for the provided sample size*

Description

Computes the smallest detectable deltaMin with power probability, for the provided sample size

Usage

```
computeMinEsBatchSaviT(
  nPlan,
  alpha = 0.05,
  power = 0.8,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  beta = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  lowEsTrue = 0.01,
  highEsTrue = 3,
  ...
)
```

Arguments

`nPlan`	optional numeric vector of length at most 2, see scenario 2 and 3 above.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`power`	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.

alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
lowEsTrue	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
highEsTrue	numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.

Value

numeric > 0 that represents the minimal detectable effect size

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computeMinEsBatchSaviT(27)
```

---

computeNEff — *Help function to compute the effective sample size based on a length 2 vector of samples*

Description

Help function to compute the effective sample size based on a length 2 vector of samples

Usage

```
computeNEff(n, testType = c("oneSample", "paired", "twoSample"), silent = TRUE)
```

Arguments

n	vector of length at most 2 representing the sample sizes of the first and second group
testType	either one of "oneSample", "paired", "twoSample".
silent	logical, if true, then turn off warnings

Value

a numeric that represents the effective sample size.

---

computeNPlanBatchSafeZ

Helper function: Computes nPlan based on meanDiffTrue, alpha and power.

Description

Helper function: Computes nPlan based on meanDiffTrue, alpha and power.

Usage

```
computeNPlanBatchSaviZ(
  meanDiffTrue,
  alpha = 0.05,
  power = 0.8,
  sigma = 1,
  kappa = sigma,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  parameter = NULL,
  meanDiffMin = NULL,
  beta = NULL,
  highN = 10000L,
  ratio = 1,
  ...
)
```

Arguments

meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
beta	numerical in (0,1). Old parameter now replaced by the power parameter

sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
highN	integer, largest possible sampling horizon. This might be the largest n that we are able to fund, which by default is set to 1e4L. Typically, highN is not used, as the function computeNPlanBatchSaviZ() tries to find the sampling horizon. If all fails, then use highN as the sampling horizon.
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
...	further arguments to be passed to or from methods.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).

Value

a list which contains at least nPlan and the savi test defining parameter.

Functions

- computeNPlanBatchSafeZ(): Deprecated version of computeNPlanBatchSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

`computeNPlanBatchSaviT` *Helper function: Computes the planned sample size for the savi T-test based deltaMin, alpha and power*

Description

Helper function: Computes the planned sample size for the savi T-test based deltaMin, alpha and power

Usage

```
computeNPlanBatchSaviT(
  deltaTrue,
  alpha = 0.05,
  power = 0.8,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  sigma = 1,
  sigma2 = 1,
  parameter = NULL,
  beta = NULL,
  ratio = 1,
  deltaMin = NULL,
  ...
)
```

Arguments

`deltaTrue`	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`power`	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`testType`	either one of "oneSample", "paired", "twoSample".
`eType`	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
`sigma`	numeric > 0 representing the population standard deviation used for the test.

sigma2	numeric > 0 representing the population standard deviation used for the test, for the second group in a two-sample t-test
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
...	further arguments to be passed to or from methods, but mainly to perform do.calls.

Value

a list which contains at least nPlan and the savi test defining parameter

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, **86**(5), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

---

computeNPlanBootstrapper

Helper function to compute uncertainty regarding nPlan estimates

---

Description

Helper function to compute uncertainty regarding nPlan estimates

Usage

```
computeNPlanBootstrapper(
  samplingResult,
  parameter,
  power,
  nPlanBatch,
  nBoot,
  beta = NULL
)
```

Arguments

samplingResult	output from sampling functions such as computeNPlanSaviZ and computeNPlanSaviT
parameter	numeric > 0, the savi test defining parameter.
power	numeric in (0, 1) that specifies the desirable power necessary to calculate both "n" and the minimum detectable effect size.
nPlanBatch	integer, the sample size needed in a batch design to reach the targeted power=1-beta with tolerable type I error alpha
nBoot	integer > 0 representing the number of bootstrap samples to estimate the uncertainty of various estimates.
beta	numerical in (0,1). Old parameter now replaced by the power parameter

Value

list with bootstrap objects

Examples

```
samplingResult <- sampleStoppingTimesSaviT(0.7, nSim=10, nMax=20)
result <- computeNPlanBootstrapper(samplingResult, 0.7, 0.2, 20, nBoot=1e2)
```

---

computeNPlanSafeT *Helper function: Computes nPlan based on deltaMin and power*

Description

Helper function: Computes nPlan based on deltaMin and power

Usage

```
computeNPlanSaviT(
  deltaTrue,
  power = 0.8,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  testType = c("oneSample", "paired", "twoSample"),
  deltaMin = NULL,
  beta = NULL,
  ratio = 1,
  parameter = NULL,
  nMax = 1e+08,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  nuMin = 2,
```

```
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  sigma = 1,
  sigma2 = 1,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  ...
)
```

Arguments

deltaTrue	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
nMax	integer > 0, maximum sample size of the (first) sample in each sample path.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi t test under continuous monitoring,or for the computation of the logarithm of the implied target.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.

deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
wantSimData	logical, if TRUE then also output the simulated data
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.
sigma	numeric > 0 representing the population standard deviation used for the test.
sigma2	numeric > 0 representing the population standard deviation used for the test, for the second group in a two-sample t-test
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.

Value

a list which contains at least nPlan and an adapted bootObject of class boot().

Functions

- computeNPlanSafeT(): Deprecated version of computeNPlanSaviT

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computeNPlanSaviT(0.7, 0.2, nSim=10)
```

---

computeNPlanSafeZ *Helper function: Computes nPlan based on meanDiffTrue, alpha and power*

Description

Helper function: Computes nPlan based on meanDiffTrue, alpha and power

Usage

```
computeNPlanSaviZ(
  meanDiffTrue,
  power = 0.8,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  testType = c("oneSample", "paired", "twoSample"),
  sigma = 1,
  kappa = sigma,
  meanDiffMin = NULL,
  beta = NULL,
  ratio = 1,
  parameter = NULL,
  nMax = 1e+08,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  highN = 10000L,
  ...
)
```

Arguments

meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
nMax	integer > 0, maximum sample size of the (first) sample in each sample path.

eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi z test under continuous monitoring,or for the computation of the logarithm of the implied target.
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
...	further arguments to be passed to or from methods.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
wantSimData	logical, if TRUE then also output the simulated data
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.
highN	integer, largest possible sampling horizon. This might be the largest n that we are able to fund, which by default is set to 1e4L. Typically, highN is not used, as the function computeNPlanBatchSaviZ() tries to find the sampling horizon. If all fails, then use highN as the sampling horizon.

Value

a list which contains at least nPlan and an adapted bootObject of class boot.

Functions

- computeNPlanSafeZ(): Deprecated version of computeNPlanSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
computeNPlanSaviZ(0.7, 0.2, nSim=10)
```

`computeNPlanSavi2x2` *Helper function: Computes nBlocksPlan based on propDiffMin, alpha and power*

Description

Runs `sampleStoppingTimesSavi2x2()` and reports its `nPlan`, the `power` quantile of the stopping time at the hardest baseline, with a bootstrap SE and the mean stopping time of paths capped at `nPlan`, both at that baseline.

Usage

```
computeNPlanSavi2x2(
  propDiffMin,
  na,
  nb,
  power = 0.8,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  nSim = 1000L,
  nBoot = nSim,
  nMax = 10000L,
  seed = NULL,
  wantSamplePaths = FALSE,
  pb = TRUE
)
```

Arguments

`propDiffMin`	numeric in (-1, 1) that defines the minimal relevant difference in proportions `thetaA - thetaB`, the smallest difference that we would like to detect (with sufficient power).
`na`	number of observations in group A per data block, one value or one per block.
`nb`	number of observations in group B per data block, one value or one per block.

power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi 2x2 test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, or the number of blocks for the savi 2x2 test under continuous monitoring.
nMax	integer > 0, maximum number of data blocks in each sample path.
seed	integer, seed number. Default seed=NULL yields seed=2026.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.

Value

a list which contains at least nPlan (the worst-case block count, Inf with a warning when the worst baseline crossed too rarely) and adapted bootObjects bootObjNPlan and bootObjNMean of class boot::boot() on the worst baseline (NULL when nPlan is Inf); further nPlanVec (the quantile per curve), nMean, worstCaseThetaA, worstCaseThetaB, and the sampler's thetaA, thetaB, stoppingTimes, breakVector, eValuesStopped, samplePaths, n1Vector.

---

computePowerBootstrapper

Helper function to compute uncertainty regarding nPlan estimates

---

Description

Helper function to compute uncertainty regarding nPlan estimates

Usage

```
computePowerBootstrapper(samplingResult, parameter, nPlan, nBoot)
```

Arguments

samplingResult	output from sampling functions such as computeNPlanSaviZ and computeNPlanSaviT
parameter	numeric > 0, the savi test defining parameter.
nPlan	vector of max length 2 representing the planned sample sizes.
nBoot	integer > 0 representing the number of bootstrap samples to estimate the uncertainty of various estimates.

Value

list with bootstrap objects

Examples

```
samplingResult <- sampleStoppingTimesSaviT(0.7, nSim=10, nMax=20)
result <- computeNPlanBootstrapper(samplingResult, 0.7, 0.2, 20, nBoot=1e2)
```

`computePowerSavi2x2` *Helper function: Computes the power of the savi 2x2 test based on propDiffMin and nBlocks*

Description

Runs `sampleStoppingTimesSavi2x2()` with `nMax = nBlocks` and reports, per curve, the fraction of paths that cross `1 / alpha` within `nBlocks`; the worst case is the smallest, since the test must meet its target whatever the baseline is.

Usage

```
computePowerSavi2x2(
  propDiffMin,
  na,
  nb,
  nBlocks,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  nSim = 1000L,
  nBoot = nSim,
  seed = NULL,
  wantSamplePaths = FALSE,
  pb = TRUE
)
```

Arguments

`propDiffMin`	numeric in (-1, 1) that defines the minimal relevant difference in proportions `thetaA - thetaB`, the smallest difference that we would like to detect (with sufficient power).
`na`	number of observations in group A per data block, one value or one per block.
`nb`	number of observations in group B per data block, one value or one per block.
`nBlocks`	integer > 0, the planned number of data blocks.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.

alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi 2x2 test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, or the number of blocks for the savi 2x2 test under continuous monitoring.
seed	integer, seed number. Default seed=NULL yields seed=2026.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.

Value

a list which contains at least power (the worst case over the curves) and an adapted bootObject bootObjPower of class boot::boot() on that baseline; further powerVec (one per curve), worstCaseThetaA, worstCaseThetaB, nBlocks, and the sampler's thetaA, thetaB, stoppingTimes, breakVector, eValuesStopped, samplePaths, n1Vector.

---

computeStatsForLogrank

Computes the sufficient statistics needed to compute 'logrankSingleZ'

Description

Computes the sufficient statistics needed to compute 'logrankSingleZ'

Usage

```
computeStatsForLogrank(
  survDataFrame,
  y0Index,
  y1Index,
  timeNow,
  timeBefore,
  survType = "right",
  ...
)
```

Arguments

survDataFrame	a 'Surv' object converted to a matrix, then to a data.frame
y0Index	vector of integers corresponding to the control group
y1Index	vector of integers corresponding to the treatment group

timeNow	numeric, current time
timeBefore	numeric, previous time
survType	character, either "right" or "counting" (left truncated, right censored)
...	further arguments to be passed to or from methods.

Value

Returns a list containing at least the following components:

obs0 number of observations in the control group.

obs1 number of observations in the treatment group.

y0 total number of participants in the control group.

y1 total number of participants in the treatment group.#'

Examples

```
data <- generateSurvData(nP = 5,
                         nT = 5,
                         lambdaP = 0.03943723,
                         lambdaT = 0.5*0.03943723,
                         endTime = 40,
                         seed = 2006)

survObj <- survival::Surv(data$time, data$status)

survDataFrame <- as.data.frame(as.matrix(survObj))
y0Index <- which(data$group=="P")
y1Index <- which(data$group=="T")

timeNow <- 4
timeBefore <- 0

computeStatsForLogrank(survDataFrame, y0Index, y1Index, timeNow, timeBefore)

timeNow <- 13
timeBefore <- 4

computeStatsForLogrank(survDataFrame, y0Index, y1Index, timeNow, timeBefore)
```

---

computeZTSumStats *Helper function to compute the relevant summary statistics in z and t-tests*

Description

Helper function to compute the relevant summary statistics in z and t-tests

Usage

```
computeZTSumStats(
  x,
  y = NULL,
  sequential = NULL,
  paired = FALSE,
  varEqual = TRUE,
  testType = NULL
)
```

Arguments

x	a (non-empty) numeric vector of data values.
y	an optional (non-empty) numeric vector of data values.
sequential	a logical indicating whether a sequential analysis should be performed.
paired	a logical, if TRUE then pair the data.
varEqual	a logical variable indicating whether to treat the two variances as being equal. Default varEqual=TRUE.
testType	either one of "oneSample", "paired", "twoSample".

Value

a list of summary statistics

Examples

```
n1 <- 23
n2 <- 39

x <- rnorm(n1, sd=7)
y <- rnorm(n2, sd=3)

computeZTSumStats(x=x, y=y)
```

---

conjugateBfTStat *A "subjective" Bayes factor for the two-sample T-test*

Description

Based on conjugate priors with a total of 8 hyperparameters.

Usage

```
conjugateBfTStat(
  x1,
  sdObs1,
  n1,
  x2,
  sdObs2,
  n2,
  a1 = 3.98,
  g1 = 0.03,
  a2 = 4.02,
  g2 = 0.05,
  a0 = 4,
  g0 = 2,
  aGamma = 2,
  bGamma = 1/2,
  log = FALSE
)
```

Arguments

x1	numeric, sample mean of group 1
sdObs1	numeric, the observed standard deviation of the first group.
n1	integer sample size of group 1
x2	numeric, sample mean of group 2
sdObs2	numeric, the observed standard deviation of the second group.
n2	integer sample size of group 2
a1	numeric, prior mean of the population mean mu1 of group 1
g1	numeric > 0, conditional prior variance of the population mean mu1 of group 1 is given by `g1*sigma^2`
a2	numeric, prior mean of the population mean mu2 of group 2
g2	numeric > 0, conditional prior variance of the population mean mu2 of group 2 is given by `g2*sigma^2`
a0	numeric, prior mean of the overall population mean mu0 of both groups
g0	numeric > 0, conditional prior variance of the population mean mu0 of both groups is given by `g1*sigma^2`
aGamma	numeric > 0, shape parameter of the prior on the standard deviation sigma
bGamma	numeric > 0, rate parameter of the prior on the standard deviation sigma
log	logical, default FALSE, if TRUE then return logarithm of the subjective Bayes factor outcome

Value

numeric > 0 representing the subjective Bayes factor outcome in favour of the alternative over the null

References

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
conjugateBfTStat(5.2, 2, 3, 3.4, 2, 12)
```

---

conjugateBfTStatOld *A "subjective" Bayes factor for the two-sample T-test*

Description

Based on conjugate priors with a total of 8 hyperparameters.

Usage

```
conjugateBfTStatOld(
  x1,
  sdObs1,
  n1,
  x2,
  sdObs2,
  n2,
  a1 = 3.98,
  g1 = 0.5,
  a2 = 4.02,
  g2 = 0.1,
  a0 = 4,
  g0 = 2,
  aGamma = 2,
  bGamma = 1/2,
  log = FALSE
)
```

Arguments

x1	numeric, sample mean of group 1
sdObs1	numeric, the observed standard deviation of the first group.
n1	integer sample size of group 1
x2	numeric, sample mean of group 2
sdObs2	numeric, the observed standard deviation of the second group.
n2	integer sample size of group 2

a1	numeric, prior mean of the population mean mu1 of group 1
g1	numeric > 0, conditional prior variance of the population mean mu1 of group 1 is given by g1*sigma^2
a2	numeric, prior mean of the population mean mu2 of group 2
g2	numeric > 0, conditional prior variance of the population mean mu2 of group 2 is given by g2*sigma^2
a0	numeric, prior mean of the overall population mean mu0 of both groups
g0	numeric > 0, conditional prior variance of the population mean mu0 of both groups is given by g1*sigma^2
aGamma	numeric > 0, shape parameter of the prior on the standard deviation sigma
bGamma	numeric > 0, rate parameter of the prior on the standard deviation sigma
log	logical, default FALSE, if TRUE then return logarithm of the subjective Bayes factor outcome

Value

numeric > 0 representing the subjective Bayes factor outcome in favour of the alternative over the null

References

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
conjugateBfTStat(5.2, 2, 3, 3.4, 2, 12)
```

---

constructSafeDesignObj

Construct a savi design object to be set in the design function

---

Description

Construct a savi design object to be set in the design function

Usage

```
constructSaviDesignObj(testName)
```

Arguments

testName	The name of the analysis that is performed such as "Z-Test", and "Test of Two Proportions".

Value

a savi design object

Functions

- `constructSafeDesignObj()`: Deprecated version of constructSaviDesignObj

Examples

```
obj <- constructSaviDesignObj("Z-Test")
```

`constructSafeTestObj` *Construct a savi test object to be set in the savi testing function*

Description

Construct a savi test object to be set in the savi testing function

Usage

```
constructSaviTestObj(testName)
```

Arguments

`testName`	The name of the analysis that is performed such as "Z-Test", and "Test of Two Proportions".

Value

a savi test object

Functions

- `constructSafeTestObj()`: Deprecated version of constructSaviTestObj

Examples

```
obj <- constructSaviTestObj("Z-Test")
```

`constructSampleStoppingTimesList` *Construct a list to be set in the sampleStoppingTimes... function*

Description

Construct a list to be set in the sampleStoppingTimes... function

Usage

```
constructSampleStoppingTimesList(
  nSim = 1000L,
  nMax = 1000L,
  wantEValuesAtNMax = FALSE,
  wantSamplePaths = TRUE
)
```

Arguments

`nSim`	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
`nMax`	integer > 0, maximum sample size of the (first) sample in each sample path.
`wantEValuesAtNMax`	logical. If `TRUE`, then compute eValues at nMax. Default `FALSE`.
`wantSamplePaths`	logical. If `TRUE`, then output the (stopped) sample paths. Default `TRUE`.

Value

a list with names

Examples

```
obj <- constructSampleStoppingTimesList()
```

`defineTTestN` *Computes a Sequence of (Effective) Sample Sizes of Z- and T-Tests*

Description

Helper function that outputs a sequence of sample sizes, effective sample sizes, and the degrees of freedom depending on the type of T-test. Also used for Z-tests.

Usage

```
defineTTestN(
  lowN = 3,
  highN = 100,
  ratio = 1,
  testType = c("oneSample", "paired", "twoSample")
)
```

Arguments

lowN	integer that defines the smallest n of our search space for n.
highN	integer largest sample size of the (first) sample. Default set to 100.
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
testType	either one of "oneSample", "paired", "twoSample".

Value

Returns the sample sizes and degrees of freedom.

---

designFreqT *Design a Frequentist T-Test*

Description

Computes the number of samples necessary to reach a tolerable type I and desired power for the frequentist T-test.

Usage

```
designFreqT(
  deltaMin,
  alpha = 0.05,
  power = 0.8,
  alternative = c("twoSided", "greater", "less"),
  h0 = 0,
  testType = c("oneSample", "paired", "twoSample"),
  ...
)
```

Arguments

deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).

alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
h0	numeric, representing the null value, default h0=0.
testType	either one of "oneSample", "paired", "twoSample".
...	further arguments to be passed to or from methods, but mainly to perform do.calls.

Value

Returns an object of class 'freqTDesign'. An object of class 'freqTDesign' is a list containing at least the following components:

nPlan the planned sample size(s).

esMin the minimal clinically relevant standardised effect size provided by the user.

alpha the tolerable type I error provided by the user.

power the desired power provided by the user.

lowN the smallest n of the search space for n provided by the user.

highN the largest n of the search space for n provided by the user.

testType any of "oneSample", "paired", "twoSample" provided by the user.

alternative any of "twoSided", "greater", "less" provided by the user.

Examples

```
designFreqT(0.5)
```

---

designFreqZ *Design a Frequentist Z-Test*

Description

Computes the number of samples necessary to reach a tolerable type I and desired power for the frequentist Z-test.

Usage

```
designFreqZ(
  meanDiffMin,
  alternative = c("twoSided", "greater", "less"),
  alpha = 0.05,
  power = 0.8,
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  sigma = 1,
  h0 = 0,
  kappa = sigma,
  lowN = 3L,
  highN = 100L,
  ...
)
```

Arguments

meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
h0	numeric, representing the null value, default h0=0.
kappa	the true population standard deviation. Default kappa=sigma.
lowN	integer that defines the smallest n of our search space for n.
highN	integer that defines the largest n of our search space for n. This might be the largest n that we are able to fund.
...	further arguments to be passed to or from methods.

Value

returns a 'freqZDesign' object.

Examples

```
freqDesign <- designFreqZ(meanDiffMin = 0.5, highN = 100)
freqDesign$nPlan
freqDesign2 <- designFreqZ(meanDiffMin = 0.2, lowN = 32, highN = 200)
freqDesign2$nPlan
```

---

designSafe1aHelper *Helper function that extract the results for the design scenario 1a: Target nPlan*

Description

Helper function that extract the results for the design scenario 1a: Target nPlan

Usage

```
designSavi1aHelper(
  samplingResult,
  esMin,
  power,
  ratio,
  beta = NULL,
  testType = c("oneSample", "paired", "twoSample")
)
```

Arguments

samplingResult	output from sampling functions such as computeNPlanSaviZ and computeNPlanSaviT
esMin	numeric that defines the minimal clinically relevant effect size, e.g. meanDiffMin for the z-test, or deltaMin for the t-test.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
testType	either one of "oneSample", "paired", "twoSample".
power	numeric in (0, 1) that specifies the desirable power necessary to calculate both "n" and the minimum detectable effect size.

Value

a list of partial results for the design scenario 1a

Functions

- `designSafe1aHelper()`: Deprecated version of designSavi1aHelper

Examples

```
samplingResult <- computeNPlanSaviZ(0.7, nSim=10, nMax=20)
result <- designSavi1aHelper(samplingResult, 0.7, 0.2, 1)
```

`designSafe2Helper` *Helper function that extract the results for the design scenario 1a: Target nPlan*

Description

Helper function that extract the results for the design scenario 1a: Target nPlan

Usage

```
designSavi2Helper(
  samplingResult,
  esMin,
  nPlan,
  ratio,
  testType = c("oneSample", "paired", "twoSample")
)
```

Arguments

`samplingResult`	output from sampling functions such as computeNPlanSaviZ and computeNPlanSaviT
`esMin`	numeric that defines the minimal clinically relevant effect size, e.g. meanDiffMin for the z-test, or deltaMin for the t-test.
`nPlan`	vector of max length 2 representing the planned sample sizes.
`ratio`	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
`testType`	either one of "oneSample", "paired", "twoSample".

Value

a list of partial results for the design scenario 1a

Functions

- `designSafe2Helper()`: Deprecated version of designSavi2Helper

Examples

```
samplingResult <- computeNPlanSaviZ(0.7, nSim=10, nMax=20)
result <- designSavi1aHelper(samplingResult, 0.7, 0.2, 1)
```

designSafeLogrank *Designs a Safe Anytime-Valid Logrank Test Experiment*

Description

A designed experiment requires (1) an anticipated number of events nEvents, or even better nPlan, the number of participants to be recruited in the study, and (2) the parameter of the savi test, i.e., thetaS. Provided with a clinically relevant minimal hazard ratio hrMin, this function outputs thetaS = hrMin as the savi test defining parameter in accordance to the GROW criterion. If a tolerable type II error beta is provided then nEvents can be sampled. The sampled nEvents is then the smallest nEvents for which hrMin is found with power of at least 1 - beta under optional stopping. If exact equal FALSE, then the computations exploit the local asymptotic normal approximation to sampling distribution of the logrank test derived by Schoenfeld (1981).

Usage

```
designSaviLogrank(
  hrMin = NULL,
  power = NULL,
  nEvents = NULL,
  alpha = 0.05,
  h0 = 1,
  alternative = c("twoSided", "greater", "less"),
  m0 = 50000L,
  m1 = 50000L,
  testType = c("exactLogrank", "gaussianLogrank"),
  ratio = 1,
  exact = TRUE,
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  groupSizePerTimeFunction = returnOne,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  beta = NULL,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  wantEValuesAtNMax = NULL,
  wantSimData = FALSE,
  ...
)
```

Arguments

hrMin	numeric that defines the minimal relevant hazard ratio, the smallest hazard ratio that we want to detect.

beta	numerical in (0,1). Old parameter now replaced by the power parameter
nEvents	numeric > 0, targetted number of events.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
h0	numeric > 0, represents the null hypothesis, default h0=1.
alternative	a character string specifying the alternative hypothesis, which must be one of "twoSided" (default),"greater" or "less". The alternative is pitted against the null hypothesis of equality of the survival distributions. More specifically, let lambda1 be the hazard rate of group 1 (i.e., placebo), and lambda2 the hazard ratio of group 2 (i.e., treatment), then the null hypothesis states that the hazard ratio theta = lambda2/lambda1 = 1. If alternative = "less", the null hypothesis is compared to theta < 1, thus, lambda2 < lambda1, that is, the hazard of group 2 (i.e., treatment) is less than that of group 1 (i.e., placebo), hence, the treatment is beneficial. If alternative = "greater", then the null hypothesis is compared to theta > 1, thus, lambda2 > lambda1, hence, harm.
m0	Number of subjects in the control group 0/1 at the beginning of the trial, i.e., nPlan[1].
m1	Number of subjects in the treatment group 1/2 at the beginning of the trial, i.e., nPlan[2].
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 (Treatment) over condition 1 (Placebo), thus, m1/m0. Note that m1 and m0 are not used to specify ratio. Ratio is only used when zApprox=TRUE, which ignores m1 and m0.
exact	a logical indicating whether the design should be based on the exact savi logrank test based on the hypergeometric likelihood. Default is TRUE, if FALSE then the design is based on a savi z-test.
parameter	Numeric > 0, represents the savi tests defining thetaS. Default NULL so it's decided by the algorithm, typically, this equals hrMin, which corresponds to the GROW choice.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
groupSizePerTimeFunction	A function without parameters and integer output. This function provides the number of events at each time step. For instance, if rpois(1, 7) leads to a random number of events at each time step.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of events for the exact savi logrank test under continuous monitoring

nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximation of power or nEvents for the exact savi logrank test under continuous monitoring
...	further arguments to be passed to or from methods.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop for the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. This overrides the "beta" argument
relevanceTest	logical, if TRUE then impose rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant standardised mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
wantEValuesAtNMax	logical. If TRUE then compute eValues at nMax. Default FALSE.
wantSimData	logical. If TRUE, then output the simulated data.

Value

Returns a saviDesign object that includes:

nEvents the anticipated number of events, either (1) specified by the user, or (2) computed based on beta and thetaMin.

parameter the parameter that defines the savi test. Here log(thetaS).

esMin the minimally clinically relevant hazard ratio specified by the user.

alpha the tolerable type I error provided by the user.

beta the tolerable type II error provided by the user.

alternative any of "twoSided", "greater", "less" provided by the user.

testType "logrank".

ratio default is 1. It defines the ratio between the planned randomisation of condition 2 over condition 1.

pilot FALSE to indicate that the design is not a pilot study.

call the expression with which this function is called.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Schoenfeld, D. (1981). The asymptotic properties of nonparametric tests for comparing survival distributions. *Biometrika*, **68**(*1*), 316-319, https://doi.org/10.2307/2335833.

Examples

```
designSaviLogrank(hrMin=0.7)
designSaviLogrank(hrMin=0.7, exact=FALSE)
designSaviLogrank(hrMin=0.7, beta=0.3, nSim=10)
designSaviLogrank(hrMin=0.7, nEvents=190, nSim=10)
```

`designSafeLogrank2WantBeta` *Helper function to design a savi logrank test (output beta)*

Description

Finds the parameter and beta when provided with only alpha, esMin, and nPlan

Usage

```
designSaviLogrank2WantBeta(
  hrMin,
  nEvents,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  m0 = 50000L,
  m1 = 50000L,
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  groupSizePerTimeFunction = returnOne,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  ...
)
```

Arguments

`hrMin`	numeric that defines the minimal relevant hazard ratio, the smallest hazard ratio that we want to detect.
`nEvents`	numeric > 0, targetted number of events.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`alternative`	a character string specifying the alternative hypothesis, which must be one of "twoSided" (default),"greater" or "less". The alternative is pitted against the null hypothesis of equality of the survival distributions. More specifically, let

lambda1 be the hazard rate of group 1 (i.e., placebo), and lambda2 the hazard ratio of group 2 (i.e., treatment), then the null hypothesis states that the hazard ratio theta = lambda2/lambda1 = 1. If alternative = "less", the null hypothesis is compared to theta < 1, thus, lambda2 < lambda1, that is, the hazard of group 2 (i.e., treatment) is less than that of group 1 (i.e., placebo), hence, the treatment is beneficial. If alternative = "greater", then the null hypothesis is compared to theta > 1, thus, lambda2 > lambda1, hence, harm.

`m0` Number of subjects in the control group 0/1 at the beginning of the trial, i.e., `nPlan[1]`.

`m1` Number of subjects in the treatment group 1/2 at the beginning of the trial, i.e., `nPlan[2]`.

`testType` either one of "oneSample", "paired", "twoSample".

`ratio` numeric > 0 representing the randomisation ratio of condition 2 (Treatment) over condition 1 (Placebo), thus, m1/m0. Note that m1 and m0 are not used to specify ratio. Ratio is only used when `zApprox=TRUE`, which ignores m1 and m0.

`parameter` Numeric > 0, represents the savi tests defining thetaS. Default NULL so it's decided by the algorithm, typically, this equals hrMin, which corresponds to the GROW choice.

`eType` character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.

`wantSamplePaths` logical, if `TRUE` then also outputs the sample paths.

`groupSizePerTimeFunction` A function without parameters and integer output. This function provides the number of events at each time step. For instance, if `rpois(1, 7)` leads to a random number of events at each time step.

`pb` logical, if `TRUE`, then show progress bar.

`seed` integer, seed number.

`nSim` integer > 0, the number of simulations needed to compute power or the number of events for the exact savi logrank test under continuous monitoring

`nBoot` integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximation of power or nEvents for the exact savi logrank test under continuous monitoring

`...` further arguments to be passed to or from methods.

Value

A list with the parameter and beta amongst other items

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Schoenfeld, D. (1981). The asymptotic properties of nonparametric tests for comparing survival distributions. *Biometrika*, **68**(*1*), 316-319, https://doi.org/10.2307/2335833.

Examples

```
designSaviLogrank2WantBeta(hrMin=0.9, nEvents=7, nSim=10)
```

---

designSafeT *Design a Safe Anytime-Valid Experiment to Test Means with a Z Test*

Description

A designed experiment requires (1) a sample size nPlan to plan for, and (2) a savi test defining parameter. The design involves alpha and the three quantities: (1) nPlan, (2) power, and (3) a minimal clinically relevant standarised mean difference deltaMin.

Scenario 1.a Goal: "nPlan" and optimal E-variable. Given: deltaMin and power.

Scenario 1.b Goal: an optimal E-variable. Given: deltaMin only.

Scenario 2 Goal: "power" and optimal E-variable. Given: deltaMin and nPlan.

Scenario 3.a Goal: "deltaMin" and optimal E-variable. Given: power and nPlan.

Scenario 3.b Goal: an optimal E-variable. Given: nPlan only.

Usage

```
designSaviT(
  deltaMin = NULL,
  power = NULL,
  nPlan = NULL,
  alpha = 0.05,
  h0 = 0,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  beta = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  deltaTrue = NULL,
```

```
  sigma = 1,
  sigma2 = sigma,
  lowEsTrue = 0.01,
  highEsTrue = 3,
  varEqual = TRUE,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  betaDefault = 0.2,
  highN = 10000L,
  wantSampling = TRUE,
  nuMin = 2,
  ...
)
```

Arguments

deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
beta	numerical in (0,1). Old parameter now replaced by the power parameter
nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
h0	numeric, representing the null value, default h0=0.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
lowEsTrue	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
highEsTrue	numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.

pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi t test under continuous monitoring,or for the computation of the logarithm of the implied target.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.
deltaTrue	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
wantSimData	logical, if TRUE then also output the simulated data
sigma	numeric > 0 representing the population standard deviation used for the test.
sigma2	numeric > 0 representing the population standard deviation used for the test, for the second group in a two-sample t-test
varEqual	a logical variable indicating whether to treat the two variances as being equal. Default varEqual=TRUE.
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.
betaDefault	numeric, defaulting value for 1-power and alphaRelevance
highN	integer, largest possible sampling horizon. This might be the largest n that we are able to fund, which by default is set to 1e4L. Typically, highN is not used, as the function computeNPlanBatchSaviT() tries to find the sampling horizon. If all fails, then use highN as the sampling horizon.
wantSampling	logical, default TRUE so sampling paths are drawn. For instance, if meanDiffMin and power, are given, then nPlan (scenario 1a) is derived by sampling. Set this to FALSE, whenever we want to run a minimal efficacy test without needing to know nPlan
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.

Details

Every scenario returns an E-variable adapted to the input. Scenario 1.a, for instance, outputs the parameter of the provided eType (default mom) savi test, see matchEParameterWith for details,

and nPlan. The nPlan is based on samples paths drawn under deltaTrue (if not specified, then deltaTrue=deltaMin by default). The resulting nPlan corresponds to the power (say 80%) quantile of the first-passage time distribution associated with E crossing threshold 1/alpha.

Value

Returns an object of class 'saviDesign'. An object of class 'saviDesign' is a list containing at least the following components:

parameter the savi test defining parameter, see `matchEParameterWith`.

alpha the tolerable type I error provided by the user.

pilot logical, specifying whether it's a pilot design, which occurs when saviTTest is called without a designObj.

testName "T-Test".

call the expression with which this function is called.

Functions

- `designSafeT()`: Deprecated version of designSaviT
- `designSafeT2WantBeta()`: Deprecated version of designSaviT2WantBeta
- `designSafeT3WantEsMin()`: Deprecated version of designSafeT3WantEsMin
- `designSafeT3bWantParameter()`: Deprecated version of designSaviT3bWantParameter
- `computeNPlanBatchSafeT()`: Deprecated version of computeNPlanBatchSaviT
- `computeMinEsBatchSafeT()`: Deprecated version of computeMinEsBatchSaviT

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
# Scenario 1.b: Goal: an E-variable
designObj <- designSaviT(deltaMin=0.8)

# Scenario 1.a: Goal: "nPlan" and optimal E-variable.
designObj <- designSaviT(deltaMin=0.8, power=0.6, alpha=0.2,
                         alternative="greater", nSim=100)

plot(designObj)

# Scenario 1a. with relevance testing, also stopping for practically null
designObj <- designSaviT(deltaMin=0.8, power=0.6, alpha=0.2,
                         alternative="greater", nSim=100,
                         relevanceTest=TRUE)
```

```
plot(designObj)

# Scenario 2: Goal: "power" and optimal E-variable
designObj <- designSaviT(deltaMin=0.8, nPlan=16, nSim=100)

# Scenario 3.a: Goal: "meanDiffMin" and optimal E-variable
designObj <- designSaviT(power=0.7, nPlan=16)

# Scenario 3.b: Goal: an optimal E-variable. Given: nPlan only.
designObj <- designSaviT(nPlan=16)
```

`designSafeT1aWantNPlan` *Helper function to designing a savi T-test (output nPlan)*

Description

Finds the parameter and power when provided with only alpha, esMin, and nPlan

Usage

```
designSaviT1aWantNPlan(
  deltaMin,
  power,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  deltaTrue = NULL,
  beta = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  sigma = 1,
  sigma2 = 1,
  alphaRelevance = NULL,
  nuMin = 2,
  ...
)
```

Arguments

deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
beta	numerical in (0,1). Old parameter now replaced by the power parameter
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi t test under continuous monitoring,or for the computation of the logarithm of the implied target.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
deltaTrue	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
wantSimData	logical, if TRUE then also output the simulated data
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
sigma	numeric > 0 representing the population standard deviation used for the test.

sigma2	numeric > 0 representing the population standard deviation used for the test, for the second group in a two-sample t-test
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.

Value

A list with the parameter and the targeted nPlan amongst other items

Functions

- designSafeT1aWantNPlan(): Deprecated version of designSaviT1aWantNPlan

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(5), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviT1aWantNPlan(deltaMin=0.9, power=0.7, nSim=10)
```

---

designSafeTwoProportions

Designs a Savi Experiment to Test Two Proportions in Stream Data

---

Description

The design requires the number of observations one expects to collect in each group in each data block. I.e., when one expects balanced data, one could choose na = nb = 1 and would be allowed to analyse the data stream each time a new observation in both groups has come in. The best results in terms of power are achieved when the data blocks are chosen as small as possible, as this allows for analysing and updating the savi test as often as possible, to fit the data best. Further, the design requires two out of the following three parameters to be known:

- the power one aims to achieve (1 - beta),
- the minimal relevant difference between the groups (delta)
- the number of blocks planned (nBlocksPlan),

where the unknown out of the three will be estimated. In the case of an exploratory "pilot" analysis, one can also only provide the number of blocks planned.

Usage

```
designSaviTwoProportions(
  na,
  nb,
  nBlocksPlan = NULL,
  beta = NULL,
  delta = NULL,
  alternativeRestriction = c("none", "difference", "logOddsRatio"),
  alpha = 0.05,
  pilot = "FALSE",
  hyperParameterValues = NULL,
  previousSaviTestResult = NULL,
  M = 1000,
  simThetaAMin = NULL,
  simThetaAMax = NULL
)
```

Arguments

na	number of observations in group a per data block
nb	number of observations in group b per data block
nBlocksPlan	planned number of data blocks collected
beta	numeric in (0, 1) that specifies the tolerable type II error control necessary to calculate both "nBlocksPlan" and "delta". Note that 1-beta defines the power.
delta	a priori minimal relevant divergence between group means b and a, either a numeric between -1 and 1 for no alternative restriction or a restriction on difference, or a real for a restriction on the log odds ratio.
alternativeRestriction	a character string specifying an optional restriction on the alternative hypothesis; must be one of "none" (default), "difference" (difference group mean b minus group b) or "logOddsRatio" (the log odds ratio between group means b and a).
alpha	numeric in (0, 1) that specifies the tolerable type I error control –independent on n– that the designed test has to adhere to. Note that it also defines the rejection rule e10 >= 1/alpha.
pilot	logical, specifying whether it's a pilot design.
hyperParameterValues	named list containing numeric values for hyperparameters betaA1, betaA2, betaB1 and betaB2, with betaA1 and betaB1 specifying the parameter equivalent to `shape1` in `stats::dbeta` for groups A and B, respectively, and betaA2 and betaB2 equivalent to `shape2`. By default chosen to optimize evidence collected over subsequent experiments (REGRET). Pass in the following format: `list(betaA1 = numeric1, betaA2 = numeric2, betaB1 = numeric3, betaB2 = numeric4)`.
M	number of simulations used to estimate power or nBlocksPlan. Default `1000`.

`simThetaAMin` minimal event rate in control group to simulate nPlan or power for. Can be specified when specifically interested in planning studies for specific event rates. Default `NULL`, then the entire parameter space (possibly restricted by delta) is used for simulation.

`simThetaAMax` maximal event rate in control group to simulate nPlan or power for. Default `NULL`.

`previousSaviTestResult`
optionally, a previous savi test result can be provided. The posterior of the hyperparameters of this test is then used for the hyperparameter settings. Default NULL.

Value

Returns a 'saviDesign' object that includes:

nPlan the sample size(s) to plan for. Computed based on beta and meanDiffMin, or provided by the user if known.

parameter the savi test defining parameter: here the hyperparameters.

esMin the minimally clinically relevant effect size provided by the user.

alpha the tolerable type I error provided by the user.

beta the tolerable type II error specified by the user.

alternative any of "twoSided", "greater", "less" based on the `alternativeRestriction` provided by the user.

testType here 2x2

pilot logical, specifying whether it's a pilot design.

call the expression with which this function is called.

Examples

```
#plan for an experiment to detect minimal difference of 0.6 with a balanced design
set.seed(3152021)
designSaviTwoProportions(na = 1,
                         nb = 1,
                         alpha = 0.1,
                         beta = 0.20,
                         delta = 0.6,
                         alternativeRestriction = "none",
                         M = 75)

#savi analysis of a pilot: number of samples already known
designSaviTwoProportions(na = 1,
                          nb = 1,
                          nBlocksPlan = 20,
                          pilot = TRUE)

#specify own hyperparameters
hyperParameterValues <- list(betaA1 = 10, betaA2 = 1, betaB1 = 1, betaB2 = 10)
designSaviTwoProportions(na = 1,
```

```
                          nb = 1,
                          alpha = 0.1,
                          beta = 0.20,
                          delta = 0.6,
                          hyperParameterValues = hyperParameterValues,
                          alternativeRestriction = "none",
                          M = 75)

#restrict range of proportions for estimating nPlan in the control group
designSaviTwoProportions(na = 1,
                          nb = 1,
                          beta = 0.20,
                          delta = 0.3,
                          alternativeRestriction = "none",
                          M = 75,
                          simThetaAMin = 0.1, simThetaAMax = 0.2)
```

`designSafeZ` *Design a Safe Anytime-Valid Experiment to Test Means with a Z Test*

Description

A designed experiment requires (1) a sample size nPlan to plan for, and (2) a savi test defining parameter. The design involves alpha and the three quantities: (1) nPlan, (2) power, and (3) a minimal clinically relevant mean difference meanDiffMin.

Scenario 1.a Goal: "nPlan" and optimal E-variable. Given: meanDiffMin and power.

Scenario 1.b Goal: an optimal E-variable. Given: meanDiffMin only.

Scenario 2 Goal: "power" and optimal E-variable. Given: meanDiffMin and nPlan.

Scenario 3.a Goal: "meanDiffMin" and optimal E-variable. Given: power and nPlan.

Scenario 3.b Goal: an optimal E-variable. Given: nPlan only.

Usage

```
designSaviZ(
  meanDiffMin = NULL,
  power = NULL,
  nPlan = NULL,
  alpha = 0.05,
  h0 = 0,
  alternative = c("twoSided", "greater", "less"),
  sigma = 1,
  kappa = sigma,
  meanDiffTrue = NULL,
  beta = NULL,
  testType = c("oneSample", "paired", "twoSample"),
```

```
  ratio = 1,
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  lowEsTrue = 0.01,
  highEsTrue = 3,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  betaDefault = 0.2,
  highN = 10000L,
  wantSampling = TRUE,
  ...
)
```

Arguments

meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
beta	numerical in (0,1). Old parameter now replaced by the power parameter
nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
h0	numeric, representing the null value, default h0=0.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.

lowEsTrue	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
highEsTrue	numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi z test under continuous monitoring,or for the computation of the logarithm of the implied target.
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
...	further arguments to be passed to or from methods.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
wantSimData	logical, if TRUE then also output the simulated data
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.
betaDefault	numeric, defaulting value for 1-power and alphaRelevance
highN	integer, largest possible sampling horizon. This might be the largest n that we are able to fund, which by default is set to 1e4L. Typically, highN is not used, as the function computeNPlanBatchSaviZ() tries to find the sampling horizon. If all fails, then use highN as the sampling horizon.
wantSampling	logical, default TRUE so sampling paths are drawn. For instance, if meanDiffMin and power, are given, then nPlan (scenario 1a) is derived by sampling. Set this to FALSE, whenever we want to run a minimal efficacy test without needing to know nPlan

Details

Every scenario returns an E-variable adapted to the input. Scenario 1.a, for instance, outputs the parameter of the provided eType (default mom) savi test, see matchEParameterWith for details, and nPlan. The nPlan is based on samples paths drawn under meanDiffTrue (if not specified, then meanDiffTrue=meanDiffMin by default). The resulting nPlan corresponds to the power (say 80%) quantile of the first-passage time distribution associated with E crossing threshold 1/alpha.

Value

Returns a saviDesign object that includes:

parameter the savi test defining parameter, see `matchEParameterWith`.

alpha the tolerable type I error provided by the user.

pilot logical, specifying whether it's a pilot design, which occurs when saviZTest is called without a designObj.

testName "Z-Test".

call the expression with which this function is called.

Functions

- `designSafeZ()`: Deprecated version of designSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
# Scenario 1.b: Goal: an E-variable
designObj <- designSaviZ(meanDiffMin=0.8)

# Scenario 1.a: Goal: "nPlan" and optimal E-variable.
designObj <- designSaviZ(meanDiffMin=0.8, power=0.6, alpha=0.2,
                         alternative="greater", nSim=100)

plot(designObj)

# Scenario 1a. with relevance testing, also stopping for practically null
designObj <- designSaviZ(meanDiffMin=0.8, power=0.6, alpha=0.2,
                         alternative="greater", nSim=100,
                         relevanceTest=TRUE)

plot(designObj)

# Scenario 2: Goal: "power" and optimal E-variable
designObj <- designSaviZ(meanDiffMin=0.8, nPlan=16, nSim=100)

# Scenario 3.a: Goal: "meanDiffMin" and optimal E-variable
designObj <- designSaviZ(power=0.7, nPlan=16)

# Scenario 3.b: Goal: an optimal E-variable. Given: nPlan only.
designObj <- designSaviZ(nPlan=16)
```

designSafeZ1aWantNPlan *Helper function to designing a savi Z-test (output nPlan)*

Description

Finds the parameter and power when provided with only alpha, esMin, and nPlan

Usage

```
designSaviZ1aWantNPlan(
  meanDiffMin,
  power,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  sigma = 1,
  kappa = sigma,
  beta = NULL,
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  meanDiffTrue = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  highN = 10000L,
  ...
)
```

Arguments

meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
beta	numerical in (0,1). Old parameter now replaced by the power parameter
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".

sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi z test under continuous monitoring,or for the computation of the logarithm of the implied target.
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
...	further arguments to be passed to or from methods.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
wantSimData	logical, if TRUE then also output the simulated data
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.
highN	integer, largest possible sampling horizon. This might be the largest n that we are able to fund, which by default is set to 1e4L. Typically, highN is not used, as the function computeNPlanBatchSaviZ() tries to find the sampling horizon. If all fails, then use highN as the sampling horizon.

Value

A list with the parameter and the targeted nPlan amongst other items

Functions

- `designSafeZ1aWantNPlan()`: Deprecated version of designSaviZ1aWantNPlan

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviZ1aWantNPlan(meanDiffMin=0.9, power=0.7, nSim=10)
```

---

designSafeZ2WantBeta *Helper function to designing a Z-test (output power)*

Description

Finds the parameter and power when provided with only alpha, esMin, and nPlan

Usage

```
designSaviZ2WantPower(
  meanDiffTrue,
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  sigma = 1,
  kappa = sigma,
  meanDiffMin = meanDiffTrue,
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  ...
)
```

Arguments

meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi z test under continuous monitoring,or for the computation of the logarithm of the implied target.
...	further arguments to be passed to or from methods.
meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
wantSimData	logical, if TRUE then also output the simulated data
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.

Value

A list with the parameter and power amongst other items

Functions

- `designSafeZ2WantBeta()`: Deprecated version of designSaviZ2WantBeta

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviZ2WantPower(meanDiffTrue=0.9, nPlan=7, nSim=10)
```

---

designSafeZ3bWantParameter

Helper function to designing a Z-test (output meanDiffMin based on the shortest interval at nPlan)

Description

Finds the parameter and meanDiffMin when provided with only alpha, nPlan

Usage

```
designSaviZ3bWantParameter(
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  sigma = 1,
  kappa = sigma,
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  ...
)
```

Arguments

nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
testType	either one of "oneSample", "paired", "twoSample".
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
...	further arguments to be passed to or from methods.

Value

A list with the parameter and the parameter amongst other items

Functions

- designSafeZ3bWantParameter(): Deprecated version of designSaviZ3bWantParameter

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviZ3bWantParameter(nPlan=13)
```

`designSafeZ3WantEsMin` *Helper function to designing a Z-test (output esMin)*

Description

Finds the parameter and esMin when provided with only alpha, power, and nPlan

Usage

```
designSaviZ3WantEsMin(
  power,
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  sigma = 1,
  kappa = sigma,
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  beta = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  lowEsTrue = 0.01,
  highEsTrue = 3,
  ...
)
```

Arguments

`beta`	numerical in (0,1). Old parameter now replaced by the power parameter
`nPlan`	optional numeric vector of length at most 2, see scenario 2 and 3 above.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`sigma`	numeric > 0 representing the assumed population standard deviation used to scale the data.
`kappa`	the true population standard deviation. Default kappa=sigma.
`testType`	either one of "oneSample", "paired", "twoSample".
`parameter`	numeric, an optional savi test defining parameter. Default set to `NULL`. and adapts to meanDiffMin and eType, see `matchEParameterWith` for details.
`eType`	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
`lowEsTrue`	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.

highEsTrue — numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.

... — further arguments to be passed to or from methods.

power — numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.

Value

A list with the parameter and the targeted esMin amongst other items

Functions

- `designSafeZ3WantEsMin()`: Deprecated version of designSaviZ3WantEsMin

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviZ3WantEsMin(power=0.7, nPlan=10)
```

---

designSavi2x2 *Design a Safe Anytime-Valid Experiment to Test Two Proportions*

Description

A designed experiment requires (1) a number of data blocks nBlocksPlan to plan for, and (2) a savi test defining parameter: `eType` and, with it, the effect measure. Both effects are A minus B and anchored on `thetaB`: `propDiff = thetaA - thetaB` and `logOdds = logit(thetaA) - logit(thetaB)`, so `"greater"` means group A has the larger proportion, as `x - y > 0` does in `stats::t.test()`.

Usage

```
designSavi2x2(
  na,
  nb,
  nBlocksPlan = NULL,
  propDiffMin = NULL,
  logOddsMin = NULL,
```

```
  alpha = 0.05,
  power = NULL,
  h0 = 0,
  alternative = c("twoSided", "greater", "less"),
  eType = c("eBeta", "grow", "eGauss"),
  betaParameter = NULL,
  gaussParameter = NULL,
  runningIntersection = NULL,
  nSim = 1000L,
  nBoot = 1000L,
  nMax = 10000L,
  seed = NULL,
  wantSamplePaths = FALSE,
  pb = TRUE
)
```

Arguments

na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
nBlocksPlan	planned number of data blocks collected, see scenario 2 and 3 above.
propDiffMin	numeric in (-1, 1) that defines the minimal relevant difference in proportions thetaA - thetaB, the smallest difference that we would like to detect (with sufficient power).
logOddsMin	numeric that defines the minimal relevant log odds ratio logit(thetaA) - logit(thetaB).
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true.
h0	numeric, representing the null value, default h0=0. Only h0 = 0 is currently supported.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
eType	character one of "eBeta", "grow", and "eGauss". "eBeta" is default and uses a Beta prior on each proportion, "grow" uses a point prior at the minimal effect, "eGauss" a normal prior on the log odds ratio.
betaParameter	list(betaA1, betaA2, betaB1, betaB2), the Beta prior shapes on thetaA and thetaB for "eBeta", each a single finite positive number; NULL means 1 / (2 * na) and 1 / (2 * nb), taking block 1's sizes with a warning when they vary by block. A given prior keeps its own block 1: the test does not replace it by the UMP e-value. For "grow" on propDiff, the test puts Beta(betaA1, betaA2) on thetaA rescaled to its feasible interval on the curve, B's shapes unused, and NULL means uniform; planning ignores it.

gaussParameter list(mean, sd), the Normal prior on logOdds for "eGauss", restricted to the grid (-20, 20) and to the side of a one-sided alternative; NULL means list(mean = 0, sd = 1).

runningIntersection
logical, if TRUE then intersect each row of the blockwise confidence sequence with the previous one; NULL keeps the constructor's FALSE.

nSim integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi 2x2 test under continuous monitoring.

nBoot integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, or the number of blocks for the savi 2x2 test under continuous monitoring.

nMax integer > 0, maximum number of data blocks in each sample path.

seed integer, seed number. Default seed=NULL yields seed=2026.

wantSamplePaths
logical, if TRUE then also outputs the sample paths.

pb logical, if TRUE, then show progress bar.

Details

- "eBeta" (propDiff) and "eGauss" (logOdds) are unrestricted: a minimal effect is an error, a power is dropped with a warning, and nBlocksPlan is kept as the planned block count. Each reads one prior, betaParameter or gaussParameter; NULL keeps the default, which savi2x2TestStat() fills in. A one-sided alternative acts in the UMP first block, and for "eGauss" also restricts the prior to that side.
- "grow" plugs in exactly one of propDiffMin, logOddsMin as the fixed alternative and reads no prior. As in designSaviZ(), a value whose sign contradicts a one-sided alternative is flipped with a warning; "twoSided" uses the magnitude. Zero is an error.

For "grow" on propDiff the design involves alpha and the three quantities: (1) nBlocksPlan, (2) power, and (3) a minimal relevant difference in proportions propDiffMin, simulated at the worst-case baseline of sampleStoppingTimesSavi2x2().

Scenario 1a Goal: "nBlocksPlan" and optimal E-variable. Given: propDiffMin and power.

Scenario 2 Goal: "power" and optimal E-variable. Given: propDiffMin and nBlocksPlan.

Scenario 3 Goal: "propDiffMin" and optimal E-variable. Given: power and nBlocksPlan.

A minimal effect alone designs without simulation. Any other combination is an error, except that logOddsMin drops a power with a warning. Per-block size vectors supply nBlocksPlan when it is NULL. Other arguments are taken as given; only alpha and power in (0, 1), equal lengths of na and nb, at most one minimal effect, and the minimal effects' ranges are checked.

Value

Returns a saviDesign object that includes:

nPlan the planned sample size(s): list(na, nb), with nBlocksPlan when given or planned.

parameter for "eBeta" and "eGauss", the prior as one named string, or "default" when NULL; absent for "grow", whose minimal effect is esMin.

esMin the minimal relevant effect size provided by the user, or found in scenario 3, signed and named `propDiff` or `logOdds`; `NULL` for "eBeta" and "eGauss".

alpha the tolerable type I error provided by the user.

power the desired power provided by the user, or the worst-case power found in scenario 2.

alternative any of "twoSided", "greater", "less" provided by the user.

eType any of "eBeta", "grow", "eGauss" provided by the user.

h0 the null value, 0.

betaParameter for "eBeta" and "grow" on propDiff, the Beta prior given, `NULL` for the default.

gaussParameter for "eGauss", the Normal prior given, `NULL` for the default.

runningIntersection logical, as provided by the user.

designScenario "1a", "2" or "3" when planned.

nPlanTwoSe, bootObjNBlocksPlan, nMean, nMeanTwoSe, bootObjNMean scenario 1a: the bootstrap summaries of the planned block count and of the mean stopping time.

powerTwoSe, bootObjPower scenario 2: the bootstrap summaries of the power.

worstCaseThetaA, worstCaseThetaB, breakVector, samplePaths scenarios 1a and 2: the worst baseline and its simulated paths.

testType here 2x2

testName "Two Proportions".

call the expression with which this function is called.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Turner, R., Ly, A., & Grünwald, P. D. (2024). Generic e-variables for exact sequential k-sample tests that allow for optional stopping. *Journal of Statistical Planning and inference* **230**, 106116, https://doi.org/10.1016/j.jspi.2023.106116.

Examples

```
# Unrestricted test on propDiff with the default Beta prior
designSavi2x2(na = 10, nb = 10, eType = "eBeta")

# Grow on a minimal difference, no planning
designSavi2x2(na = 10, nb = 10, propDiffMin = 0.2, eType = "grow")

# Scenario 2: worst-case power at a planned block count
designSavi2x2(na = 10, nb = 10, propDiffMin = 0.3, nBlocksPlan = 8,
              eType = "grow", nSim = 50, pb = FALSE)
```

`designSaviT2WantPower` *Helper function to designing a savi T-test (output power)*

Description

Finds the parameter and power when provided with only alpha, esMin, and nPlan

Usage

```
designSaviT2WantPower(
  deltaTrue,
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  deltaMin = deltaTrue,
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  alphaRelevance = NULL,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  nBoot = nSim,
  nuMin = 2,
  ...
)
```

Arguments

`deltaTrue`	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
`nPlan`	optional numeric vector of length at most 2, see scenario 2 and 3 above.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`deltaMin`	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
`testType`	either one of "oneSample", "paired", "twoSample".
`ratio`	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.

parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
wantSimData	logical, if TRUE then also output the simulated data
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.
nBoot	integer > 0 representing the number of bootstrap samples to assess the accuracy of the approximations of the power, the number of samples for the savi t test under continuous monitoring,or for the computation of the logarithm of the implied target.
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.

Value

A list with the parameter and beta amongst other items

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviT2WantPower(deltaTrue=0.9, nPlan=7, nSim=10)
```

`designSaviT3bWantParameter` *Helper function to designing a savi T-test (output deltaMin based on the shortest interval at nPlan)*

Description

Finds the parameter and deltaMin when provided with only alpha and nPlan

Usage

```
designSaviT3bWantParameter(
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  ...
)
```

Arguments

`nPlan`	optional numeric vector of length at most 2, see scenario 2 and 3 above.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`testType`	either one of "oneSample", "paired", "twoSample".
`parameter`	numeric, an optional savi test defining parameter. Default set to `NULL`. and adapts to meanDiffMin and eType, see `matchEParameterWith` for details.
`eType`	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
`...`	further arguments to be passed to or from methods, but mainly to perform do.calls.

Value

A list with the parameter and the parameter amongst other items

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviT3bWantParameter(nPlan=20)
```

`designSaviT3WantEsMin` *Helper function to designing a Savi T-test (output esMin)*

Description

Finds the parameter and esMin when provided with only alpha, power, and nPlan

Usage

```
designSaviT3WantEsMin(
  power,
  nPlan,
  alpha = 0.05,
  alternative = c("twoSided", "greater", "less"),
  testType = c("oneSample", "paired", "twoSample"),
  parameter = NULL,
  beta = NULL,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  lowEsTrue = 0.01,
  highEsTrue = 3,
  ...
)
```

Arguments

`power`	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
`nPlan`	optional numeric vector of length at most 2, see scenario 2 and 3 above.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".

testType	either one of "oneSample", "paired", "twoSample".
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
lowEsTrue	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
highEsTrue	numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.

Value

A list with the parameter and the targeted esMin amongst other items

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
designSaviT3WantEsMin(power=0.7, nPlan=10)
```

---

extractNameFromArgs *Helper function: Get all names as entered by the user*

Description

Helper function: Get all names as entered by the user

Usage

```
extractNameFromArgs(list, name)
```

Arguments

list	list from which the element needs retrieving
name	character string, name of the item that need retrieving

Value

returns a character string

`fnchLogPartition` *Log partition function of Fisher's noncentral hypergeometric distribution*

Description

Log partition function of Fisher's noncentral hypergeometric distribution

Usage

```
fnchLogPartition(na, nb, totalSuccesses, logOdds)
```

Arguments

`na`	number of observations in group A per data block, one value or one per block.
`nb`	number of observations in group B per data block, one value or one per block.
`totalSuccesses`	nonnegative integer, the total successes `ya + yb` in the table, at most `na + nb`.
`logOdds`	numeric, one log odds ratio `logit(thetaA) - logit(thetaB)`.

Value

A numeric, the log of the sum over the feasible `k` of `choose(na, k) * choose(nb, totalSuccesses - k) * exp(logOdds * k)`.

`generateNormalData` *Generates Normally Distributed Data Depending on the Design*

Description

The designs supported are "oneSample", "paired", "twoSample".

Usage

```
generateNormalData(
  nPlan,
  nSim = 1000L,
  deltaTrue = NULL,
  muGlobal = 0,
  sigma = 1,
  sigma2 = 1,
  paired = FALSE,
  seed = NULL,
  meanDiffTrue = NULL
)
```

Arguments

nPlan	optional numeric vector of length at most 2, see scenario 2 and 3 above.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.
deltaTrue	numeric, the value of the true standardised effect size (test-relevant parameter). This argument is used by designSaviT() with deltaTrue <- deltaMin
muGlobal	numeric, population grand mean
sigma	numeric > 0, population standard deviation
sigma2	numeric > 0 representing the population standard deviation used for the test, for the second group in a two-sample t-test
paired	a logical, if TRUE then pair the data.
seed	integer, seed number.
meanDiffTrue	numeric, data governing parameter value

Value

Returns a list of two data matrices contains at least the following components:

dataGroup1 a matrix of data dimension nSim by nPlan[1].

dataGroup2 a matrix of data dimension nSim by nPlan[2].

Examples

```
generateNormalData(20, 15, deltaTrue=0.3)
```

---

generateSurvData *Generate Survival Data which Can Be Analysed With the* survival *Package*

Description

Generate Survival Data which Can Be Analysed With the survival Package

Usage

```
generateSurvData(
  nP,
  nT,
  alpha = 1,
  lambdaP,
  lambdaT,
  seed = NULL,
  nDigits = 0,
  startTime = 1,
```

```
  endTime = 180,
  orderTime = TRUE,
  competeRatio = 0
)
```

Arguments

nP	integer > 0 representing the number of of patients in the placebo group.
nT	integer > 0 representing the number of of patients in the treatment group.
alpha	numeric > 0, representing the shape parameter of the Weibull distribution. If alpha=1, then data are generated from the exponential, i.e., constant hazard. For alpha > 1 the hazard increases, if alpha < 1, the hazard decreases.
lambdaP	The (relative) hazard of the placebo group.
lambdaT	The (relative) hazard of the treatment group.
seed	A seed number.
nDigits	numeric, the number of digits to round of the random time to
startTime	numeric, adds this to the random times. Default 1, so the startTime is not 0, which is the start time of rweibull.
endTime	The endtime of the experiment.
orderTime	logical, if TRUE then put the data set in increasing order
competeRatio	The ratio of the data that is due to competing risk.

Value

A data set with time, status and group.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
generateSurvData(800, 800, alpha=1, lambdaP=0.008, lambdaT=0.008/2)
```

getArgs *Helper function: Get all arguments as entered by the user*

Description

Helper function: Get all arguments as entered by the user

Usage

```
getArgs()
```

Value

a list of variable names of class "call" that can be changed into names

getNameAlternative *Gets the Label of the Alternative Hypothesis*

Description

Helper function that outputs the alternative hypothesis of the analysis.

Usage

```
getNameAlternative(
  alternative = c("twoSided", "greater", "less"),
  testType,
  h0 = 0
)
```

Arguments

alternative	A character string. "twoSided", "greater", "less".
testType	A character string either "oneSample", "paired", "twoSample", "gLogrank", or "eLogrank".
h0	the value of the null hypothesis

Value

Returns a character string with the name of the analysis.

getNameTestType *Gets the Label of the Test*

Description

Helper function that outputs the name of the analysis.

Usage

```
getNameTestType(testType, testName, relevanceTest = FALSE)
```

Arguments

testType	A character string. For the t-tests: "oneSample", "paired", "twoSample".
testName	The name of the analysis that is performed such as "Z-Test", and "Test of Two Proportions".
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.

Value

Returns a character string with the name of the analysis.

isTryError *Checks Whether a Vector of Object Inherits from the Class 'try-error'*

Description

Checks whether any of the provided objects contains a try error.

Usage

```
isTryError(...)
```

Arguments

...	objects that need testing.

Value

Returns TRUE if there's some object that's a try-error, FALSE when all objects are not try-errors.

Examples

```
x <- 1
y <- "a"
z <- try(integrate(exp, -Inf, Inf))
isTryError(x, y)
isTryError(x, y, z)
```

`lcosh` *Logarithmic hyperbolic cosine*

Description

Logarithmic hyperbolic cosine

Usage

```
lcosh(x)
```

Arguments

`x`	numeric

Value

numeric value

Examples

```
lcosh(7)
```

`logEValueVec2x2LogOddsEGauss`
Plain eGauss e-process for logOdds

Description

The conditional likelihood of Fisher's noncentral hypergeometric distribution on blocks `1..i`, mixed under the Normal prior `gaussParameter` on `logOddsGrid` restricted to the side of a one-sided `alternative`, over the hypergeometric null. The mixture for block `i` includes block `i`, which is valid because each factor conditions on its block's total.

Usage

```
logEValueVec2x2LogOddsEGauss(
  ya,
  yb,
  na,
  nb,
  gaussParameter = NULL,
  alternative = c("twoSided", "greater", "less"),
  logOddsGrid = seq(-20, 20, length.out = 2000)
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
gaussParameter	list(mean, sd), the Normal prior on logOdds for "eGauss", restricted to the grid (-20, 20) and to the side of a one-sided alternative; NULL means list(mean = 0, sd = 1).
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
logOddsGrid	numeric vector, the grid the prior is normalised on.

Value

A numeric vector, element i the log e-value on blocks 1..i.

---

logEValueVec2x2LogOddsGrow

Plain grow e-process for logOdds

Description

The cumulative conditional log likelihood ratio of logLikelihoodFNCH() at logOddsMin, signed by alternative, over the hypergeometric null. "twoSided" runs both signs of the magnitude and averages their cumulative e-processes.

Usage

```
logEValueVec2x2LogOddsGrow(
  ya,
  yb,
  na,
  nb,
  logOddsMin,
  alternative = c("twoSided", "greater", "less")
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
logOddsMin	numeric that defines the minimal relevant log odds ratio logit(thetaA) - logit(thetaB).
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.

Value

A numeric vector, element i the log e-value on blocks 1..i.

---

logEValueVec2x2PropDiffEBeta

Plain eBeta e-process for propDiff

Description

The cumulative log likelihood ratio of the predictive Beta means of predictiveThetas2x2() over the pooled null mean (na * thetaA + nb * thetaB) / (na + nb), block 1 included.

Usage

```
logEValueVec2x2PropDiffEBeta(ya, yb, na, nb, betaParameter)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
betaParameter	list(betaA1, betaA2, betaB1, betaB2), the Beta prior shapes on thetaA and thetaB for "eBeta", each a single finite positive number; NULL means 1 / (2 * na) and 1 / (2 * nb), taking block 1's sizes with a warning when they vary by block. A given prior keeps its own block 1: the test does not replace it by the UMP e-value. For "grow" on propDiff, the test puts Beta(betaA1, betaA2) on thetaA rescaled to its feasible interval on the curve, B's shapes unused, and NULL means uniform; planning ignores it.

Value

A numeric vector, element i the log e-value on blocks 1..i.

---

logEValueVec2x2PropDiffGrow

Plain grow e-process for propDiff

Description

The alternative is restricted to the curve thetaA = thetaB + propDiffMin, signed by alternative, which leaves one free parameter; its posterior on a grid of nWeight points under a Beta prior, uniform by default, gives the predictable plug-in for each block, against the pooled null mean. "twoSided" runs both signs of the magnitude and averages their cumulative e-processes.

Usage

```
logEValueVec2x2PropDiffGrow(
  ya,
  yb,
  na,
  nb,
  propDiffMin,
  alpha,
  alternative = c("twoSided", "greater", "less"),
  betaParameter = NULL,
  earlyStopping = FALSE,
  nWeight = 1000L
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
propDiffMin	numeric in (-1, 1) that defines the minimal relevant difference in proportions thetaA - thetaB, the smallest difference that we would like to detect (with sufficient power).
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
betaParameter	NULL or a list with betaA1, betaA2: the prior Beta(betaA1, betaA2) on thetaA rescaled to its feasible interval on the curve; B's shapes are unused. NULL means Beta(1, 1), uniform.
earlyStopping	logical, if TRUE cut the vector at the first crossing of 1 / alpha.
nWeight	integer > 0, the number of grid points.

Value

A numeric vector, element i the log e-value on blocks 1..i.

---

logLikelihoodFNCH *Conditional log likelihood of 2x2 tables at one logOdds*

Description

Given its total ya + yb, ya is Fisher's noncentral hypergeometric with odds exp(logOdds) on group A; logOdds = 0 is the hypergeometric null. Evaluated on the log scale, binomial coefficients plus logOdds times the feasible count normalised by log-sum-exp, so extreme logOdds give finite values where the plain density underflows to 0.

Usage

```
logLikelihoodFNCH(ya, yb, na, nb, logOdds)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
logOdds	numeric, one log odds ratio `logit(thetaA) - logit(thetaB)`.

Value

A numeric vector, one log density per block.

---

logrankSingleEExact *Helper function computes single component of the exact logrank e-value*

Description

Helper function computes single component of the exact logrank e-value

Usage

```
logrankSingleEExact(obs0, obs1, y0, y1, thetaS, theta0 = 1, ...)
```

Arguments

obs0	integer, number of observations in the control group.
obs1	integer, number of observations in the treatment group.
y0	integer, total number of participants in the control group.
y1	integer, total number of participants in the treatment group.
thetaS	numeric > 0 represents the savi test defining (GROW) alternative hypothesis obtained from `designSaviLogrank()`.
theta0	numeric > 0 represents the null hypothesis. Default theta0=1.
...	further arguments to be passed to or from methods.

Value

Returns a list containing at least the following components:

logP0 Log likelihood of Fisher's hypergeometric at the null

logEValueLess Log likelihood of Fisher's hypergeometric at the alternative

logEValueGreater Log likelihood of Fisher's hypergeometric at 1/alternative

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
#'
y0Vector <- c(5, 4, 3, 3, 2, 1)
y1Vector <- c(5, 5, 4, 2, 2, 0)
obs0Vector <- c(1, 1, 0, 1, 0, 1)
obs1Vector <- c(0, 0, 1, 0, 1, 0)

logEValueGreater <- logEValueLess <- vector("numeric", length(y0Vector))

for (i in seq_along(y0Vector)) {
  tempResult <- logrankSingleEExact(obs0=obs0Vector[i], obs1=obs1Vector[i],
                                    y0=y0Vector[i], y1=y1Vector[i],
                                    thetaS=0.7, theta0=1)
  logEValueLess[i] <- tempResult[["logEValueLess"]]
  logEValueGreater[i] <- tempResult[["logEValueGreater"]]
}

eValueLess <- exp(sum(logEValueLess))
eValueLess #1.116161
eValueGreater <- exp(sum(logEValueGreater))
eValueGreater # 0.7665818
eValue <- 1/2*eValueLess + 1/2*eValueGreater
eValue # 0.9413714
```

`logrankSingleZ` *Helper function computes single component of the logrank statistic*

Description

Helper function computes single component of the logrank statistic

Usage

```
logrankSingleZ(obs0, obs1, y0, y1, ...)
```

Arguments

`obs0`	integer, number of observations in the control group
`obs1`	integer, number of observations in the treatment group

y0	integer, total number of participants in the control group
y1	integer, total number of participants in the treatment group
...	further arguments to be passed to or from methods.

Value

Returns a list containing at least the following components:

oMinE observed minus expected.

v hypergeometric variance.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
y0Vector <- c(6, 4, 4, 1, 0)
y1Vector <- c(6, 6, 5, 2, 2)
obs0Vector <- c(1, 0, 2, 1, 0)
obs1Vector <- c(0, 1, 1, 0, 1)

varVector <- oMinEVector <-y0Vector

for (i in seq_along(y0Vector)) {
  tempResult <- logrankSingleZ(obs0=obs0Vector[i], obs1=obs1Vector[i],
                               y0=y0Vector[i], y1=y1Vector[i])
  oMinEVector[i] <- tempResult[["oMinE"]]
  varVector[i] <- tempResult[["v"]]
}

sum(oMinEVector)/sqrt(sum(varVector))
```

---

makeRunningIntersection

Helper function to create running intersections

Description

Helper function to create running intersections

Usage

```
makeRunningIntersection(x, upper = TRUE)
```

Arguments

x	vector of numeric, representing a sequence of upper or lower bounds of a confidence sequence
upper	logic, by default TRUE to construct a running intersection for the upper bound of a sequence with the minimum function. If FALSE, then use the maximum function for the lower bound.

Value

a sequence of numerics representing the running minimum or maximum

Examples

```
makeRunningIntersection(c(6, -1, 3, 12))
```

---

matchAlphaRelevanceWith

Checks and outputs a threshold for a minimal efficacy analysis

---

Description

Checks and outputs a threshold for a minimal efficacy analysis

Usage

```
matchAlphaRelevanceWith(
  alphaRelevance,
  alpha = NULL,
  power = NULL,
  beta = NULL,
  betaDefault = 0.2
)
```

Arguments

alphaRelevance	numeric > 0 and < 1, used to set the threshold of a minimal efficacy procedure
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
beta	numeric > 0 and < 1, a tolerable type II error, used to set the threshold if alphaRelevance is not given
betaDefault	numeric > 0 and < 1 a default value (0.2) to run a minimal efficacy procedure

Value

a alphaRelevance threshold

Examples

```
matchAlphaRelevanceWith(0.3)
```

`matchEParameterWith` *Match the parameter of a savi z or t-test*

Description

Based on the minimal clinically relevant effect size esMin, sigma (for z-tests), alternative and eType

Usage

```
matchEParameterWith(
  esMin,
  analysisType = c("z", "t", "logRank"),
  sigma = 1,
  alternative = c("twoSided", "greater", "less"),
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  parameter = NULL
)
```

Arguments

`esMin`	numeric: meanDiffMin for z-tests, or deltaMin for t-tests
`analysisType`	character. Either "z", "t", or "logRank", currently.
`sigma`	numeric > 0 representing the assumed population standard deviation used to scale the data.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`eType`	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
`parameter`	numeric, an optional savi test defining parameter. Default set to `NULL`. and adapts to meanDiffMin and eType, see `matchEParameterWith` for details.

Value

the parameter, a numeric value

Examples

```
matchEParameterWith(0.4)
```

matchEsMinWith *Match the meanDiffMin of a savi z-test*

Description

Based on the parameter, sigma, alternative and eType

Usage

```
matchEsMinWith(
  parameter,
  analysisType = c("z", "t"),
  sigma = 1,
  alternative = c("twoSided", "greater", "less"),
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow")
)
```

Arguments

parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
analysisType	character. Either "z", "t", or "logRank", currently.
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.

Value

the parameter, a numeric value

Examples

```
matchEsMinWith(parameter=0.4)
```

matchPowerWith *Helper function to check whether the power or beta (redundant now) argument is given*

Description

Helper function to check whether the power or beta (redundant now) argument is given

Usage

```
matchPowerWith(power, beta = NULL)
```

Arguments

power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
beta	numerical in (0,1). Old parameter now replaced by the power parameter

Value

numeric representing power

Examples

```
matchPowerWith(0.8)
```

matchRelevanceParameterWith *Match the parameter of a minimal efficacy savi z-test*

Description

Based on the relevanceSize, meanDiffMin, alternative and eType

Usage

```
matchRelevanceParameterWith(relevanceSize, esMin, esTrue)
```

Arguments

relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
esMin	numeric: meanDiffMin for z-tests, or deltaMin for t-tests
esTrue	numeric. meanDiffTrue for z-tests, or deltaTrue for t-tests

Value

the parameter, a numeric value

Examples

```
matchRelevanceParameterWith(0.4)
```

`plot.safe2x2Sim` *Plots Results of Simulations for Comparing Hyperparameters for Savi Tests of Two Proportions*

Description

Plots Results of Simulations for Comparing Hyperparameters for Savi Tests of Two Proportions

Usage

```
## S3 method for class 'savi2x2Sim'
plot(x, ...)
```

Arguments

`x`	a result object obtained through `simulateTwoProportions()`.
`...`	further arguments to be passed to or from methods.

Value

Plot data, mainly called for side effects, the plot of simulation results.

Examples

```
priorList1 <- list(betaA1 = 10, betaA2 = 1, betaB1 = 1, betaB2 = 10)
priorList2 <- list(betaA1 = 0.18, betaA2 = 0.18, betaB1 = 0.18, betaB2 = 0.18)
priorList3 <- list(betaA1 = 1, betaA2 = 1, betaB1 = 1, betaB2 = 1)

simResult <- simulateTwoProportions(
  hyperparameterList = list(priorList1, priorList2, priorList3),
  alternativeRestriction = "none",
  alpha = 0.1, beta = 0.2, na = 1, nb = 1,
  deltamax = -0.4, deltamin = -0.9, deltaGridSize = 3,
  M = 10
  )

plot(simResult)
```

plot.safeDesign *Plots the saviDesign object for designs with sample paths*

Description

Plots the saviDesign object for designs with sample paths

Usage

```
## S3 method for class 'saviDesign'
plot(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  maxNBins = 40,
  numSamplePaths = 100,
  wantStepLines = FALSE,
  wantQuantiles = NULL,
  breaks = NULL,
  lwd = 2,
  pch = 15,
  colQuant = "#AA0000",
  overColour = "#8FC6E3",
  overColourBorder = "#1F78B4E6",
  underColour = "#E7A72199",
  underColourBorder = "#BD8E17FF",
  continueColour = "#8CA252E6",
  continueColourBorder = "#637939E6",
  histInnerColour = col,
  col = overColour,
  border = overColourBorder,
  cex = 1.3,
  yLabPAdj = -1,
  wantNotStoppedHist = FALSE,
  wantNotStoppedSamplePaths = TRUE,
  wantLegend = TRUE,
  legendAdjRelevance = c(-0.1, 0.5, 1),
  legendAdj = c(0.2, 0.8),
  legendCexFactor = 0.85,
  histFewestAtTop = FALSE,
  density = NULL,
  wantTitle = TRUE,
  ...
)
```

Arguments

x	designObj
main	character string for the title of plot
xlab	character string for the x-axis
ylab	character string for the y-axis
xlim	vector of length 2 specifying the range of the x-axis
ylim	vector of length 2 specifying the range of the y-axis
maxNBins	integer, maximum number of bins of the histogram
numSamplePaths	integer, number of sample paths to plot
wantStepLines	logical, if TRUE, then plot the sample paths as step functions (realistic).
wantQuantiles	a vector of numerics between zero and one representing the quantile levels
border	the color of the border around the bars. The default is to use blue
breaks	Break points of the histogram see hist()
lwd	The line width, a positive number, defaulting to 2.
pch	An integer specifying a symbol.
histInnerColour	A colour to be used to fill the bars.
col	colour of the lines
colQuant	colour of the quantiles
cex	size of the labels and the quantile text
...	further arguments to be passed to or from methods.
overColour	colour of an e-value sequence that cross the upper boundary of 1/alpha
overColourBorder	colour for the border of a rectangle in the histogram corresponding to e-values that cross the upper boundary of 1/alpha
underColour	colour of an e-value sequence that cross the lower boundary of alphaRelevance
underColourBorder	colour for the border of a rectangle in the histogram corresponding to e-values that cross the lower boundary of alphaRelevance
continueColour	colour for lines of e-values that did not (yet) cross a threshold
continueColourBorder	colour for the border of a rectangle in the histogram corresponding to e-values that did not (yet) cross a threshold
yLabPAdj	numeric to adjust the position of the y-axis label
wantNotStoppedHist	logical, if TRUE then show histogram of not stopped paths
wantNotStoppedSamplePaths	logical, if TRUE then show stopped sample paths of e-values
wantLegend	logical, if TRUE then show information of the percentage of stopped sample paths

`legendAdjRelevance`
: vector of 3 numerics that allow for the adjustment of the percentages shown at the bottom of the plot, when relevance testing is on

`legendAdj`
: vector of 2 numerics that allow for the adjustment of the percentages shown at the bottom of the plot

`legendCexFactor`
: numeric, additional factor to control the text size of the additional information such as the percentages at the bottom of the plot

`histFewestAtTop`
: logical, if TRUE, then fewest counts (of against the null or against minimal efficacy) at the top of the histogram

`density`
: the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. Non-positive values of density also inhibit the drawing of shading lines.

`wantTitle`
: logical, if `TRUE` then add title

Value

Nothing it only plots

Functions

- `plot(safeDesign)`: Deprecated version of plot.saviDesign

---

plot.safeTest *Plots the saviTest object for sequential analyses*

Description

Plots the saviTest object for sequential analyses

Usage

```
## S3 method for class 'saviTest'
plot(
  x,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  lwd = 3,
  cex = 1.3,
  fillPlot = NULL,
  switchNFill = 10000,
  logScale = NULL,
```

```
  switchNLog = 30,
  h0Colour = "darkgrey",
  overColour = "#8FC6E3",
  overColourBorder = "#1F78B4E6",
  underColour = "#E7A72199",
  underColourBorder = "#BD8E17FF",
  continueColour = "#8CA252E6",
  continueColourBorder = "#637939E6",
  col = overColour,
  border = overColourBorder,
  wantConfSeqPlot = FALSE,
  add = FALSE,
  density = NULL,
  angle = 45,
  xaxt = NULL,
  yaxt = NULL,
  fillOddEven = FALSE,
  runningIntersection = NULL,
  wantRelevance = NULL,
  ...
)
```

Arguments

x	designObj
main	character string for the title of plot
xlab	character string for the x-axis
ylab	character string for the y-axis
xlim	vector of length 2 specifying the range of the x-axis
ylim	vector of length 2 specifying the range of the y-axis
lwd	The line width, a positive number, defaulting to 2.
cex	size of the labels and the quantile text
fillPlot	logical, if TRUE then plot the confidence sequence with a background colour
switchNFill	integer, if `is.null(fillPlot)`, then fill if the number of samples is smaller than switchNFill
logScale	logical, if TRUE then plot on the logscale
switchNLog	integer, if `is.null(logScale)`, then plot on the log scale if the number of samples is larger than switchNLog
h0Colour	Colour to indicate the null hypothesis.
col	The colour for filling the anytime-valid confidence interval.
border	The colour to draw the border.
wantConfSeqPlot	logical, if TRUE then plot the confidence sequence instead of the e-value progression

add	logical, default FALSE so a new plot is made. If TRUE and wantConfSeqPlot==FALSE then adds the e-value progression line to an existing plot. If TRUE and wantConfSeqPlot==TRUE then adds another anytime-valid confidence sequence.
density	the density of shading lines, in lines per inch. The default value of NULL means that no shading lines are drawn. A zero value of density means no shading nor filling whereas negative values and NA suppress shading (and so allow colour filling).
angle	the slope of shading lines, given as an angle in degrees (anti-clockwise).
fillOddEven	logical controlling the polygon shading mode: see polygon() for details. Default FALSE.
...	further arguments to be passed to or from methods.
overColour	colour of an e-value sequence that cross the upper boundary of 1/alpha
overColourBorder	colour for the border of a rectangle in the histogram corresponding to e-values that cross the upper boundary of 1/alpha
underColour	colour of an e-value sequence that cross the lower boundary of alphaRelevance
underColourBorder	colour for the border of a rectangle in the histogram corresponding to e-values that cross the lower boundary of alphaRelevance
continueColour	colour for lines of e-values that did not (yet) cross a threshold
continueColourBorder	colour for the border of a rectangle in the histogram corresponding to e-values that did not (yet) cross a threshold
xaxt	default NULL. If "n" then suppresses plotting of the x-axis.
yaxt	default NULL. If "n" then suppresses plotting of the y-axis.
runningIntersection	logical, if TRUE then plot the running minimum of the upper confidence sequence across time, and the running maximum of the lower confidence sequence across time
wantRelevance	logical, if FALSE, then don't show the e-values for relevanceTest. Default wantRelevance==NULL, if designObj[["relevanceTest"]]==TRUE then relevance e-values tests are shown automatically.

Value

Returns nothing just plots

Functions

- plot(safeTest): Deprecated version of plot.saviTest

`plotConfidenceSequenceTwoProportions` *Plot bounds of a savi confidence sequence of the difference or log odds ratio for two proportions against the number of data blocks in two data streams ya and yb.*

Description

Plot bounds of a savi confidence sequence of the difference or log odds ratio for two proportions against the number of data blocks in two data streams ya and yb.

Usage

```
plotConfidenceSequenceTwoProportions(
  ya,
  yb,
  saviDesign,
  differenceMeasure = c("difference", "odds"),
  precision = 100,
  deltaStart = 0.001,
  deltaStop = 3,
  trueDifference = NA
)
```

Arguments

`ya`	positive observations/ events per data block in group a: a numeric with integer values between (and including) 0 and `na`, the number of observations in group a per block.
`yb`	positive observations/ events per data block in group b: a numeric with integer values between (and including) 0 and `nb`, the number of observations in group b per block.
`saviDesign`	a savi test design for two proportions retrieved through `designSaviTwoProportions()`.
`differenceMeasure`	the difference measure to construct the confidence interval for: one of "difference" and "odds".
`precision`	precision of the grid to search over for the confidence sequence bounds.
`deltaStart`	for the odds difference measure: the (absolute value of the) smallest log odds ratio to assess for in- or exclusion in the confidence sequence. Default 0.001.
`deltaStop`	for the odds difference measure: the (absolute value of the) highest log odds ratio to assess for in- or exclusion in the confidence sequence. Default 3.
`trueDifference`	true difference or log odds ratio in groups A and B: added to the plot.

Value

no return value; called for its side effects, a plot of the confidence sequence.

Examples

```
set.seed(39413)
ya <- rbinom(n = 30, size = 1, prob = 0.1)
yb <- rbinom(n = 30, size = 1, prob = 0.8)
balancedSaviDesign <- designSaviTwoProportions(na = 1,
                                               nb = 1,
                                               nBlocksPlan = 30)
plotConfidenceSequenceTwoProportions(ya = ya,
                                     yb = yb,
                                     saviDesign = balancedSaviDesign,
                                     differenceMeasure = "difference",
                                     precision = 15,
                                     trueDifference = 0.7)

#log odds ratio difference measure
plotConfidenceSequenceTwoProportions(ya = ya,
                                     yb = yb,
                                     saviDesign = balancedSaviDesign,
                                     differenceMeasure = "odds",
                                     precision = 15,
                                     deltaStop = 5,
                                     trueDifference = log(36))

#switch ya and yb: observe negative log odds ratio in the data, plot mirrored in x-axis
plotConfidenceSequenceTwoProportions(ya = yb,
                                     yb = ya,
                                     saviDesign = balancedSaviDesign,
                                     differenceMeasure = "odds",
                                     precision = 15,
                                     deltaStop = 5,
                                     trueDifference = -log(36))
```

---

plotHistogramDistributionStoppingTimes

Plots the Histogram of Stopping Times

Description

Helper function to display the histogram of stopping times.

Usage

```
plotHistogramDistributionStoppingTimes(
  saviSim,
  nPlan,
  deltaTrue,
  showOnlyNRejected = FALSE,
  nBin = 25L,
```

```
  ...
)
```

Arguments

saviSim	A saviSim object
nPlan	numeric > 0, the planned sample size(s).
deltaTrue	numeric, that represents the true underlying standardised effect size delta.
showOnlyNRejected	logical, when TRUE discards the cases that did not reject.
nBin	numeric > 0, the minimum number of bins in the histogram.
...	further arguments to be passed to or from methods.

Value

a histogram object, and called for its side-effect to plot the histogram.

---

predictiveThetas2x2 *Predictive Beta posterior means of thetaA and thetaB*

Description

Independent Beta posteriors on thetaA and thetaB with prior shapes betaParameter, updated with the blocks before each block: block i uses blocks 1..i-1 only, and block 1 the prior.

Usage

```
predictiveThetas2x2(ya, yb, na, nb, betaParameter)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
betaParameter	list(betaA1, betaA2, betaB1, betaB2), the Beta prior shapes on thetaA and thetaB for "eBeta", each a single finite positive number; NULL means 1 / (2 * na) and 1 / (2 * nb), taking block 1's sizes with a warning when they vary by block. A given prior keeps its own block 1: the test does not replace it by the UMP e-value. For "grow" on propDiff, the test puts Beta(betaA1, betaA2) on thetaA rescaled to its feasible interval on the curve, B's shapes unused, and NULL means uniform; planning ignores it.

Value

A list of numeric vectors `thetaA` and `thetaB`, one value per block.

`print.safe2x2Sim` *Prints Results of Simulations for Comparing Hyperparameters for Savi Tests of Two Proportions*

Description

Prints Results of Simulations for Comparing Hyperparameters for Savi Tests of Two Proportions

Usage

```
## S3 method for class 'savi2x2Sim'
print(x, ...)
```

Arguments

`x`	a result object obtained through `simulateTwoProportions()`.
`...`	further arguments to be passed to or from methods.

Value

The data frame with simulation results, called for side effects to pretty print the simulation results.

Examples

```
priorList1 <- list(betaA1 = 10, betaA2 = 1, betaB1 = 1, betaB2 = 10)
priorList2 <- list(betaA1 = 0.18, betaA2 = 0.18, betaB1 = 0.18, betaB2 = 0.18)
priorList3 <- list(betaA1 = 1, betaA2 = 1, betaB1 = 1, betaB2 = 1)

simResult <- simulateTwoProportions(
  hyperparameterList = list(priorList1, priorList2, priorList3),
  alternativeRestriction = "none",
  alpha = 0.1, beta = 0.2, na = 1, nb = 1,
  deltamax = -0.4, deltamin = -0.9, deltaGridSize = 3,
  M = 10
  )
```

print.safeDesign *Print Method for Savi Test Objects*

Description

Printing objects of class 'saviTest' modelled after `print.power.htest()`.

Usage

```
## S3 method for class 'saviDesign'
print(x, digits = getOption("digits"), prefix = "\t", ...)
```

Arguments

`x`	a saviTest object.
`digits`	number of significant digits to be used.
`prefix`	string, passed to strwrap for displaying the method components.
`...`	further arguments to be passed to or from methods.

Value

No returned value, called for side effects.

Functions

- `print(safeDesign)`: Deprecated version of print.saviDesign

Examples

```
designSaviZ(meanDiffMin=0.5)
designSaviT(deltaMin=0.5)
designSaviLogrank(hrMin=0.7)
```

print.safeTest *Print Method for Savi Test Objects*

Description

Printing objects of class 'saviTest' modelled after `print.htest()`.

Usage

```
## S3 method for class 'saviTest'
print(
  x,
  digits = getOption("digits"),
  prefix = "\t",
  runningIntersection = NULL,
  ...
)
```

Arguments

x	a saviTest object.
digits	number of significant digits to be used.
prefix	string, passed to strwrap for displaying the method components.
...	further arguments to be passed to or from methods.
runningIntersection	logical, if TRUE then report the minimum of the upper confidence sequence across time, and the maximum of the lower confidence sequence across time

Value

No returned value, called for side effects.

Functions

- print(safeTest): Deprecated version of print.saviTest

Examples

```
saviTTest(rnorm(19))
```

---

pValueFromZStat *Computes the p-value for the Z-test*

Description

Computes the p-value for the Z-test

Usage

```
pValueFromZStat(z, alternative = c("twoSided", "less", "greater"), ...)

pValueZTest(
  x,
  y = NULL,
  paired = FALSE,
  ciValue = NULL,
  alpha = 0.05,
  sigma = 1,
  h0 = 0,
  alternative = c("twoSided", "greater", "less"),
  ...
)
```

Arguments

z	numeric that represents the observed z-statistic.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
...	further arguments to be passed to or from methods.
x	a (non-empty) numeric vector of data values.
y	an optional (non-empty) numeric vector of data values.
paired	a logical indicating whether you want the paired Z-test.
ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
h0	numeric, representing the null value, default h0=0.

Value

pValueTest object

Examples

```
pValueZTest(rnorm(10))
```

`returnOne` *Auxiliary function for sampling of the logrank simulations to return the integer 1 event per time.*

Description

Auxiliary function for sampling of the logrank simulations to return the integer 1 event per time.

Usage

```
returnOne()
```

Value

1

Examples

```
returnOne()
```

`rLogrank` *Randomly samples from a logrank distribution*

Description

Draws a number of occurrences in group 1 (treatment) out of obsTotal number of occurrences.

Usage

```
rLogrank(n = 1, y0, y1, obsTotal, theta)
```

Arguments

`n`	integer, number of observations to be sampled.
`y0`	Size of the risk set of group 0 (Placebo).
`y1`	Size of the risk set of group 1 (Treatment).
`obsTotal`	Total number of observations.
`theta`	Odds of group 1 over group 0 (treatment over placebo).

Value

integer representing the number of occurrences in group 1 out of obsTotal number of occurrences.

Author(s)

Muriel Felipe Perez-Ortiz and Alexander Ly

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
rLogrank(y0=360, y1=89, obsTotal=12, theta=3.14)
```

`safeLogrankTest` *Safe Anytime-Valid Logrank Test*

Description

A savi test to test whether there is a difference between two survival curves. This function builds on the Mantel-Cox version of the logrank test.

Usage

```
saviLogrankTest(
  formula,
  designObj = NULL,
  ciValue = NULL,
  data = NULL,
  survTime = NULL,
  group = NULL,
  pilot = FALSE,
  exact = TRUE,
  computeZ = TRUE,
  ...
)

saviLogrankTestStat(
  z,
  nEvents,
  designObj,
  ciValue = NULL,
  dataNull = 1,
  sigma = 1
)
```

Arguments

formula	a formula expression as for other survival models, of the form Surv(time, status) ~ groupingVariable, see Surv for more details.
designObj	a savi logrank design obtained from designSaviLogrank.
ciValue	numeric, represents the ciValue-level of the confidence sequence. Default ciValue=NULL, and ciValue = 1 - alpha, where alpha is taken from the design object.
data	an optional data frame in which to interpret the variables occurring in survTime and group.
survTime	an optional survival time object of class 'Surv' created with Surv, or a name of a column in the data set of class 'Surv'. Does not need specifying if a formula is provided, therefore set to NULL by default.
group	an optional factor, a grouping variable. Currently, only two levels allowed. Does not need specifying if a formula is provided, therefore set to NULL by default.
pilot	a logical indicating whether a pilot study is run. If TRUE, it is assumed that the number of samples is exactly as planned. The default null h0=1 is used, alpha=0.05, and alternative="twoSided" is used. To change these default values, please use designSaviLogrank.
exact	a logical indicating whether the exact savi logrank test needs to be performed based on the hypergeometric likelihood. Default is TRUE, if FALSE then the savi z-test (for Gaussian data) applied to the logrank z-statistic is used instead.
computeZ	logical. If TRUE computes the logrank z-statistic. Default is TRUE.
...	further arguments to be passed to or from methods.
z	numeric representing the observed logrank z statistic.
nEvents	numeric > 0, observed number of events.
dataNull	numeric > 0, the null hypothesis corresponding to the z statistics. By default dataNull = 1 representing equality of the hazard ratio.
sigma	numeric > 0, scaling in the data.

Value

Returns an object of class 'saviTest'. An object of class 'saviTest' is a list containing at least the following components:

statistic the value of the summary, i.e., z-statistic or the e-value.

nEvents The number of observed events.

eValue the e-value of the savi test.

confSeq An anytime-valid confidence sequence.

estimate To be implemented: An estimate of the hazard ratio.

testType "logrank".

dataName a character string giving the name(s) of the data.

designObj an object of class "saviDesign" obtained from designSaviLogrank.

sumStats a list containing.the time of events, the progression of the risk sets and events.

call the expression with which this function is called.

Functions

- `saviLogrankTestStat()`: Savi Logrank Test based on Summary Statistic Z All provided data (i.e., z-scores) are assumed to be centred on a hazard ratio = 1, thus, log(hr) = 0 , and the proper (e.g., hypergeometric) scaling is applied to the data, so sigma = 1. The null hypothesis in the design object pertains to the population and is allowed to differ from log(theta) = 0.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
# Example taken from survival::survdiff

designObj <- designSaviLogrank(hrMin=1/2)

ovData <- survival::ovarian
ovData$survTime <- survival::Surv(ovData$futime, ovData$fustat)

saviLogrankTest(formula=survTime~ rx, data=ovData, designObj=designObj)

saviLogrankTest(survTime=survTime, group=rx, data=ovData, designObj=designObj)

# Examples taken from coin::logrank_test
## Example data (Callaert, 2003, Tab. 1)
#'
callaert <- data.frame(
  time = c(1, 1, 5, 6, 6, 6, 6, 2, 2, 2, 3, 4, 4, 5, 5),
  group = factor(rep(0:1, c(7, 8)))
)

designObj <- designSaviLogrank(hrMin=1/2)

saviLogrankTest(survival::Surv(callaert$time)~callaert$group,
                designObj = designObj)

saviLogrankTest(survTime=survival::Surv(callaert$time),
                group=callaert$group, designObj = designObj)

result <- saviLogrankTest(survTime=survival::Surv(callaert$time),
                group=callaert$group, designObj = designObj)
```

```
result

##  Sequentially
# Greater
eValueGreater <- exp(cumsum(result$sumStats$logEValueGreater))
# Less
eValueLess <- exp(cumsum(result$sumStats$logEValueLess))

# twoSided
eValueTwoSided <- 1/2*eValueGreater+1/2*eValueLess

eValueTwoSided
result$eValue

###### Example switching between savi exact and savi Gaussian logrank test

designObj <- designSaviLogrank(0.8, alternative="less")

dat <- safestats::generateSurvData(300, 300, 2, 0.0065, 0.0065*0.8, seed=1)
survTime <- survival::Surv(dat$time, dat$status)

resultE <- saviLogrankTest(survTime ~ dat$group,
                           designObj = designObj)

resultG <- saviLogrankTest(survTime ~ dat$group,
                           designObj = designObj, exact=FALSE)

resultE
resultG

###### Example switching between savi exact and savi Gaussian logrank test other side

designObj <- designSaviLogrank(1/0.8, alternative="greater")

resultE <- saviLogrankTest(survTime ~ dat$group,
                           designObj = designObj)

resultG <- saviLogrankTest(survTime ~ dat$group,
                           designObj = designObj, exact=FALSE)

if (log(resultE$eValue) >= 0 && log(resultG$eValue) >= 0 )
  stop("one-sided wrong")
```

---

safeTTestStat *Safe Anytime-Valid Student's T-Test.*

Description

Savi one- and two-sample T-tests. Takes as input vector(s) of data and a designObj from designSaviT. The function is modelled after t.test().

Usage

```
saviTTest(x, ...)

## Default S3 method:
saviTTest(
  x,
  y = NULL,
  designObj = NULL,
  paired = FALSE,
  varEqual = TRUE,
  ciValue = NULL,
  maxRoot = 10,
  sequential = NULL,
  tDensity = FALSE,
  nuMin = 2,
  wantCi = TRUE,
  ...
)

## S3 method for class 'formula'
saviTTest(formula, data, subset, na.action, ...)

savi.t.test(
  x,
  y = NULL,
  paired = FALSE,
  designObj = NULL,
  varEqual = TRUE,
  ciValue = NULL,
  ...
)
```

Arguments

tDensity	Uses the the representation of the savi T-test as the likelihood ratio of t densities.
paired	a logical, if TRUE then pair the data.
...	further arguments to be passed to or from methods.
x	a (non-empty) numeric vector of data values.
y	an optional (non-empty) numeric vector of data values.
designObj	an object obtained from designSaviT.
varEqual	a logical variable indicating whether to treat the two variances as being equal. Default varEqual=TRUE.

ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha
maxRoot	Used to bound the candidate set of width of the confidence interval/
sequential	a logical indicating whether a sequential analysis should be performed.
formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs either 1 for a one-sample or paired test or a factor with two levels giving the corresponding groups. If lhs is of class "Pair" and rhs is 1, a paired test is done
data	an optional matrix or data frame (or similar: see `model.frame()`) containing the variables in the formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action")..
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.
wantCi	default TRUE

Value

Returns an object of class 'saviTest'. An object of class 'saviTest' is a list containing at least the following components:

statistic the value of the t-statistic.

n The realised sample size(s).

eValue the realised e-value from the savi test.

confSeq A savi confidence interval for the mean (difference)

estimate the estimated means or mean (difference) depending on whether it was a one-sample test or a two-sample test.

stderr the standard error of the mean (difference), used as denominator in the t-statistic formula.

dataName a character string giving the name(s) of the data.

designObj an object of class "saviDesign" obtained from `designSaviT()`.

call the expression with which this function is called.

Functions

- `safeTTestStat()`: Deprecated version of saviTTestStat
- `safeTTestStatNEffNu()`: Deprecated version of saviTTestStatNEffNu #nu >0 degrees of freedom
- `safeTTestStatNEffNuMom()`: Deprecated version of saviTTestStatNEffNuMom
- `safeTTestStatTDensity()`: Deprecated version of saviTTestStatTDensity
- `safeTTest()`: Deprecated version of saviTTest
- `safeTTest.default()`: Deprecated version of saviTTest.default
- `safeTTest.formula()`: Deprecated version of saviTTest.formula
- `safe.t.test()`: Deprecated version of savi.t.test

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Pérez-Ortiz, M. F., Lardy, T., de Heide, R., & Grünwald, P. D. (2024). E-statistics, group invariance and anytime valid testing. *The Annals of Statistics*, **52**(*4*), 1410-1432, http://dx.doi.org/10.1214/24-AOS2394.

Wang, H., & Ramdas, A. (in press). Anytime-valid t-tests and confidence sequences for Gaussian means with unknown variance. *Sequential Analysis*, https://doi.org/10.48550/arXiv.2310.03722.

Examples

```
## Examples with simulated data ----
set.seed(1)
x <- rnorm(30, mean=0)
y <- rnorm(40, mean=0)

# Because no designObj is specified, a default
# designObj is used with deltaMin = 1/2,
# which can be thought of as a medium effect size
# according to Cohen.

res <- saviTTest(x=x, y=y)

# By default sequential=TRUE, because length(x) <= 200.
# This allows us to visualise the e-value as a function of
# the n1 and associated n2, where the ratio of sample sizes,
# ratio=n2/n1 is maintained. Here the e-value at n1=6 uses data
# x[1:6] and y[1:ceil(ratio*6)]
plot(res)

# Plots the confidence sequence
plot(res, wantConfSeqPlot=TRUE)

# See ?designSaviT for more info
# This designObj also allows for
# evidence quantification that the
# mean difference is minimal clinically
# relevant, here, larger than
# relevanceSize=meanDiffMin
designObj <- designSaviT(deltaMin=0.7, alpha=0.05,
                         alternative="twoSided",
                         testType="twoSample",
                         relevanceTest=TRUE)

res <- saviTTest(x=x, y=y, designObj=designObj)

plot(res)
```

```
# Note that the e-value against relevance falls below alphaRelevance
# We can reject the hypothesis that the effect is relevantly large,
# larger than designObj$relevanceTestSim$parameter, after a sample size of
min(which(res$eRelevanceVec <= designObj$relevanceTestSim$alpha))

set.seed(2)
x <- rnorm(30, mean=0.6)
y <- rnorm(40, mean=0)

res <- saviTTest(x, y, designObj=designObj)

plot(res)
# We could have stopped sampling after
min(which(res$eValueVec >= 1/designObj$alpha))
# the yellow curve crosses 1/alpha sooner,
# but we do **not** compare eRelevance >= 1/alpha, only eRelevance <= alphaRelevance.

## Classical example: Student's sleep data -----
plot(extra ~ group, data = sleep)

designObj <- designSaviT(deltaMin=0.6, testType="twoSample")

## Traditional interface
with(sleep, saviTTest(extra[group == 1], extra[group == 2],
                      designObj=designObj))

## Formula interface
saviTTest(extra ~ group, data = sleep, designObj=designObj)

## Formula interface to one-sample test
designObj1 <- designSaviT(deltaMin=0.6,
                          testType="oneSample",
                          sigma=2)

saviTTest(extra ~ 1, data = sleep, designObj=designObj1)

## Formula interface to paired test
## The sleep data are actually paired, so could have been in wide format:
designObjPaired <- designSaviT(deltaMin=0.6,
                               testType="paired",
                               sigma=1.4)
sleep2 <- reshape(sleep, direction = "wide",
                  idvar = "ID", timevar = "group")
saviTTest(Pair(extra.1, extra.2) ~ 1, data = sleep2,
          designObj=designObjPaired)
```

---

safeTwoProportionsTest

Perform a Savi Test for Two Proportions with Stream Data

Description

Perform a savi test for two proportions (a 2x2 contingency table test) with a result object retrieved through the design function for planning an experiment to compare two proportions in this package, `designSaviTwoProportions()`.

Usage

```
saviTwoProportionsTest(
  ya,
  yb,
  designObj = NULL,
  wantConfidenceSequence = FALSE,
  ciValue = NULL,
  confidenceBoundGridPrecision = 20,
  logOddsConfidenceSearchBounds = c(0.01, 5),
  pilot = FALSE
)

savi.prop.test(
  ya,
  yb,
  designObj = NULL,
  wantConfidenceSequence = FALSE,
  ciValue = NULL,
  confidenceBoundGridPrecision = 20,
  logOddsConfidenceSearchBounds = c(0.01, 5),
  pilot = FALSE
)
```

Arguments

`ya`	positive observations/ events per data block in group a: a numeric with integer values between (and including) 0 and `na`, the number of observations in group a per block.
`yb`	positive observations/ events per data block in group b: a numeric with integer values between (and including) 0 and `nb`, the number of observations in group b per block.
`designObj`	a savi test design for two proportions retrieved through `designSaviTwoProportions()`.
`wantConfidenceSequence`	logical that can be set to true when the user wants a savi confidence sequence to be estimated.
`ciValue`	coverage of the savi confidence sequence; default `NULL`, if NULL calculated as `1 - designObj[["alpha"]]`.
`confidenceBoundGridPrecision`	integer specifying the grid precision used to search for the confidence bounds. Default 20.

`logOddsConfidenceSearchBounds`
: vector of to positive doubles specifying the upper and lower bound of the grid to search over for finding the confidence bound for the logOddsRatio restriction. Default `(0.01, 5)`.

`pilot`
: logical that can be set to true when performing an exploratory analysis without a `designObj`; only allows for `na = nb = 1`.

Value

Returns an object of class 'saviTest'. An object of class 'saviTest' is a list containing at least the following components:

n The realised sample size(s).

eValue the e-value of the savi test.

dataName a character string giving the name(s) of the data.

designObj an object of class "saviDesign" described in `designSaviTwoProportions()`.

Examples

```
#balanced design
yb <- c(1,0,1,1,1,0,1)
ya <- c(1,0,1,0,0,0,1)
saviDesign <- designSaviTwoProportions(na = 1,
                                       nb = 1,
                                       beta = 0.20,
                                       delta = 0.6,
                                       alternativeRestriction = "none",
                                       M = 1e1)
saviTwoProportionsTest(ya = ya, yb = yb, designObj = saviDesign)

#pilot
saviTwoProportionsTest(ya = ya, yb = yb, pilot = TRUE)

#unbalanced design
yb <- c(1,0,1,1,1,0,1)
ya <- c(2,2,1,2,0,2,2)
saviDesign <- designSaviTwoProportions(na = 2,
                                       nb = 1,
                                       beta = 0.20,
                                       delta = 0.6,
                                       alternativeRestriction = "none",
                                       M = 1e1)
saviTwoProportionsTest(ya = ya, yb = yb, designObj = saviDesign)
```

safeZTest *Safe Anytime-Valid Z-Test*

Description

Savi one- and two-sample Z-tests. Takes as input vector(s) of data and a designObj from designSaviZ. The function is modelled after t.test().

Usage

```
saviZTest(x, ...)

## Default S3 method:
saviZTest(
  x,
  y = NULL,
  paired = FALSE,
  designObj = NULL,
  ciValue = NULL,
  maxRoot = 10,
  sequential = NULL,
  ...
)

## S3 method for class 'formula'
saviZTest(formula, data, subset, na.action, ...)

savi.z.test(x, y = NULL, paired = FALSE, designObj = NULL, ...)
```

Arguments

x	a (non-empty) numeric vector of data values.
...	further arguments to be passed to or from methods.
y	an optional (non-empty) numeric vector of data values.
paired	a logical indicating whether you want the paired Z-test.
designObj	an object obtained from designSaviZ().
ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha
maxRoot	Used to bound the candidate set of width of the confidence interval, whenever eType="eCauchy"
sequential	a logical indicating whether a sequential analysis should be performed.
formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs either 1 for a one-sample or paired test or a factor with two levels giving the corresponding groups. If lhs is of class "Pair" and rhs is 1, a paired test is done

`data`	an optional matrix or data frame (or similar: see `model.frame()`) containing the variables in the formula. By default the variables are taken from environment(formula).
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to getOption("na.action").

Value

Returns an object of class 'saviTest'. An object of class 'saviTest' is a list containing at least the following components:

statistic the value of the test statistic. Here the z-statistic.

n The realised sample size(s).

eValue the e-value of the savi test.

confSeq a savi confidence interval for the mean (difference).

estimate the estimated means or mean (difference) depending on whether it was a one-sample test or a two-sample test.

dataName a character string giving the name(s) of the data.

designObj an object of class "saviDesign" described in `designSaviZ()`.

call the expression with which this function is called.

Functions

- `safeZTest()`: Deprecated version of saviZTest
- `safeZTest(default)`: Deprecated version of saviZTest.default
- `safe.z.test()`: Deprecated version of saviZTest

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
## Examples with simulated data ----
set.seed(1)
x <- rnorm(30, mean=0)
y <- rnorm(40, mean=0)

# Because no designObj is specified, a default
# designObj is used with meanDiffMin = 1/2 and sigma=1,
# which can be thought of as a medium effect size
# according to Cohen.
```

```
res <- saviZTest(x=x, y=y)

# By default sequential=TRUE, because length(x) <= 200.
# This allows us to visualise the e-value as a function of
# the n1 and associated n2, where the ratio of sample sizes,
# ratio=n2/n1 is maintained. Here the e-value at n1=6 uses data
# x[1:6] and y[1:ceil(ratio*6)]
plot(res)

# Plots the confidence sequence
plot(res, wantConfSeqPlot=TRUE)

# See ?designSaviZ for more info
# This designObj also allows for
# evidence quantification that the
# mean difference is minimal clinically
# relevant, here, larger than
# relevanceSize=meanDiffMin
designObj <- designSaviZ(meanDiffMin=0.7, alpha=0.05,
                         alternative="twoSided",
                         testType="twoSample",
                         relevanceTest=TRUE)

res <- saviZTest(x=x, y=y, designObj=designObj)

plot(res)

# Note that the e-value against relevance falls below alphaRelevance
# We can reject the hypothesis that the effect is relevantly larger,
# larger than designObj$relevanceTestSim$parameter after a sample size of
min(which(res$eRelevanceVec <= designObj$relevanceTestSim$alpha))

set.seed(2)
x <- rnorm(30, mean=0.6)
y <- rnorm(40, mean=0)

res <- saviZTest(x, y, designObj=designObj)

plot(res)
# We could have stopped sampling after
min(which(res$eValueVec >= 1/designObj$alpha))
# the yellow curve crosses 1/alpha sooner,
# but we do **not** compare eRelevance >= 1/alpha, only eRelevance <= alphaRelevance

## Classical example: Student's sleep data -----
plot(extra ~ group, data = sleep)

designObj <- designSaviZ(meanDiffMin=0.6, sigma=2,
                         testType="twoSample")

## Traditional interface
with(sleep, saviZTest(extra[group == 1], extra[group == 2],
```

```
                      designObj=designObj))

## Formula interface
saviZTest(extra ~ group, data = sleep, designObj=designObj)

## Formula interface to one-sample test
designObj1 <- designSaviZ(meanDiffMin=0.6,
                          testType="oneSample",
                          sigma=2)

saviZTest(extra ~ 1, data = sleep, designObj=designObj1)

## Formula interface to paired test
## The sleep data are actually paired, so could have been in wide format:
designObjPaired <- designSaviZ(meanDiffMin=0.6,
                               testType="paired",
                               sigma=1.4)
sleep2 <- reshape(sleep, direction = "wide",
                  idvar = "ID", timevar = "group")
saviZTest(Pair(extra.1, extra.2) ~ 1, data = sleep2,
          designObj=designObjPaired)
```

`safeZTestStat` *Computes E-Values Based on the Z-Statistic*

Description

Computes e-values using the z-statistic and the sample sizes based on the test defining parameter phiS.

Evidence for practical equivalence requires the e-value to be small, i.e. smaller than alphaRelevance. If the alternative holds true, i.e. meanDiffTrue >= relevanceSize, then there is no more than alphaRelevance probability of ever seeing eRelevance <= alphaRelevance.

Usage

```
saviZTestStat(
  z,
  n1,
  n2 = NULL,
  parameter,
  alternative = c("twoSided", "less", "greater"),
  paired = FALSE,
  sigma = 1,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  ...
)

saviRelevanceZStat(
```

```
  z,
  n1,
  n2 = NULL,
  parameter = NULL,
  alternative = c("twoSided", "less", "greater"),
  paired = FALSE,
  sigma = 1,
  eType = "grow",
  relevanceSize = NULL,
  ...
)
```

Arguments

z	numeric that represents the observed z-statistic.
n1	integer that represents the size in a one-sample Z-test, (n2=NULL). When n2 is not NULL, this specifies the size of the first sample for a two-sample test.
n2	an optional integer that specifies the size of the second sample. If it's left unspecified, thus, NULL then the z-statistic is assumed to be one-sample.
parameter	numeric > 0, the savi test defining parameter, see matchEParameterWith for details.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
paired	a logical indicating whether you want the paired Z-test.
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
...	further arguments to be passed to or from methods.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)

Value

Returns an e-value.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
saviZTestStat(z=3, n1=100, parameter=0.4, eType="grow")
saviZTestStat(z=3, n1=100, parameter=0.4^2, eType="eGauss")
saviZTestStat(z=3, n1=100, parameter=0.4, eType="eCauchy")
# evidence for the alternative over minimal efficacy
saviRelevanceZStat(z=3, n1=100, parameter=0.4)
# evidence for minimal efficacy over the alternative
saviRelevanceZStat(z=0.35, n1=100, parameter=0.4)
```

---

sampleLogrankStoppingTimes

Simulate stopping times for the exact savi logrank test

Description

Simulate stopping times for the exact savi logrank test

Usage

```
sampleLogrankStoppingTimes(
  hrTrue,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  m0 = 50000L,
  m1 = 50000L,
  nSim = 1000L,
  groupSizePerTimeFunction = returnOne,
  seed = NULL,
  power = NULL,
  beta = NULL,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  wantEValuesAtNMax = FALSE,
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  parameter = NULL,
  nMax = Inf,
  pb = TRUE,
  hrMin = NULL,
  ...
)
```

Arguments

hrTrue	numeric that defines the data generating hazard ratio with which data are sampled.

`alpha` numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.

`alternative` a character string specifying the alternative hypothesis, which must be one of "twoSided" (default),"greater" or "less". The alternative is pitted against the null hypothesis of equality of the survival distributions. More specifically, let lambda1 be the hazard rate of group 1 (i.e., placebo), and lambda2 the hazard ratio of group 2 (i.e., treatment), then the null hypothesis states that the hazard ratio theta = lambda2/lambda1 = 1. If alternative = "less", the null hypothesis is compared to theta < 1, thus, lambda2 < lambda1, that is, the hazard of group 2 (i.e., treatment) is less than that of group 1 (i.e., placebo), hence, the treatment is beneficial. If alternative = "greater", then the null hypothesis is compared to theta > 1, thus, lambda2 > lambda1, hence, harm.

`m0` Number of subjects in the control group 0/1 at the beginning of the trial, i.e., `nPlan[1]`.

`m1` Number of subjects in the treatment group 1/2 at the beginning of the trial, i.e., `nPlan[2]`.

`nSim` integer > 0, the number of simulations needed to compute power or the number of events for the exact savi logrank test under continuous monitoring

`groupSizePerTimeFunction`
A function without parameters and integer output. This function provides the number of events at each time step. For instance, if `rpois(1, 7)` leads to a random number of events at each time step.

`seed` integer, seed number.

`power` numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop for the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. This overrides the "beta" argument

`beta` numerical in (0,1). Old parameter now replaced by the power parameter

`relevanceTest` logical, if `TRUE` then impose rule to stop for minimal efficiency if e <= alphaRelevance. Default `FALSE`.

`relevanceSize` numeric, the minimal clinical relevant standardised mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)

`wantEValuesAtNMax`
logical. If `TRUE` then compute eValues at nMax. Default `FALSE`.

`wantSamplePaths`
logical, if `TRUE` then also outputs the sample paths.

`wantSimData` logical. If `TRUE`, then output the simulated data.

`parameter` Numeric > 0, represents the savi tests defining thetaS. Default NULL so it's decided by the algorithm, typically, this equals hrMin, which corresponds to the GROW choice.

`nMax` An integer. Once nEvents hits nMax the experiment terminates, if it didn't stop due to threshold crossing crossing already. Default set to Inf.

`pb` logical, if `TRUE`, then show progress bar.

hrMin	numeric that defines the minimal relevant hazard ratio, the smallest hazard ratio that we want to detect.
...	further arguments to be passed to or from methods.

Value

a list with stoppingTimes and breakVector. Entries of breakVector are 0, 1. A 1 represents stopping due to exceeding nMax, and 0 due to 1/alpha threshold crossing, or running out of participants, which implies that the corresponding stopping time is Inf.

Author(s)

Muriel Felipe Perez-Ortiz and Alexander Ly

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

ter Schure, J., Pérez-Ortiz, M. F., Ly, A., & Grünwald, P. D. (2024). The Safe Logrank Test: Error control under continuous monitoring with unlimited horizon. *The New England Journal of Statistics in Data Science*, **2**(*2*), 190-214, https://doi.org/10.51387/24-NEJSDS65.

Examples

```
sampleLogrankStoppingTimes(0.7, nSim=10, nMax=30)
```

---

sampleStoppingTimesSafeT

Simulate stopping times for the savi T-test

Description

Simulate stopping times for the savi T-test

Usage

```
sampleStoppingTimesSaviT(
  deltaTrue,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  testType = c("oneSample", "paired", "twoSample"),
  ratio = 1,
  deltaMin = NULL,
  parameter = NULL,
  lowN = 3L,
  nMax = 100000000L,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
```

```
  wantEValuesAtNMax = FALSE,
  nuMin = 2,
  power = NULL,
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  sigma = 1,
  sigma2 = 1,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  beta = NULL,
  alphaRelevance = NULL,
  ...
)
```

Arguments

deltaTrue	numeric, data governing effect size used for simulations. Default deltaTrue=deltaMin.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
lowN	integer, smallest sample size (of the first group).
nMax	integer > 0, maximum sample size of the (first) sample in each sample path.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantEValuesAtNMax	logical. If TRUE then compute eValues at nMax. Default FALSE.
wantSamplePaths	logical. If TRUE then output the (stopped) sample paths. Default TRUE.
wantSimData	logical. If TRUE, then output the simulated data.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi t test under continuous monitoring.

...	further arguments to be passed to or from methods, but mainly to perform do.calls.
deltaMin	numeric that defines the minimal relevant standardised mean difference, the smallest population effect size that we would like to detect (with sufficient power).
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
sigma	numeric > 0 representing the population standard deviation used for the test.
sigma2	numeric > 0 representing the population standard deviation used for the test, for the second group in a two-sample t-test
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
beta	numerical in (0,1). Old parameter now replaced by the power parameter
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.

Value

a list with stoppingTimes and breakVector. Entries of breakVector are 0, 1. A 1 represents stopping due to exceeding nMax, and 0 due to 1/alpha threshold crossing, which implies that in corresponding stopping time is Inf.

Functions

- sampleStoppingTimesSafeT(): Deprecated version of sampleStoppingTimesSaviT

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
sampleStoppingTimesSaviT(0.7, nSim=10, nMax=20)
```

sampleStoppingTimesSafeZ *Simulate stopping times for the savi Z-test*

Description

Simulate stopping times for the savi Z-test

Usage

```
sampleStoppingTimesSaviZ(
  meanDiffTrue,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  testType = c("oneSample", "paired", "twoSample"),
  sigma = 1,
  kappa = sigma,
  ratio = 1,
  meanDiffMin = NULL,
  parameter = NULL,
  nMax = 1000L,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow"),
  wantEValuesAtNMax = FALSE,
  power = NULL,
  wantSamplePaths = TRUE,
  wantSimData = TRUE,
  pb = TRUE,
  seed = NULL,
  nSim = 1000L,
  relevanceTest = FALSE,
  relevanceSize = NULL,
  beta = NULL,
  alphaRelevance = NULL,
  ...
)
```

Arguments

meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
testType	either one of "oneSample", "paired", "twoSample".

sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
kappa	the true population standard deviation. Default kappa=sigma.
ratio	numeric > 0 representing the randomisation ratio of condition 2 over condition 1. If testType is not equal to "twoSample", or if nPlan is of length(1) then ratio=1.
parameter	numeric, an optional savi test defining parameter. Default set to NULL. and adapts to meanDiffMin and eType, see matchEParameterWith for details.
nMax	integer > 0, maximum sample size of the (first) sample in each sample path.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
wantEValuesAtNMax	logical. If TRUE, then compute eValues at nMax. Default FALSE.
wantSamplePaths	logical. If TRUE, then output the (stopped) sample paths. Default TRUE.
wantSimData	logical. If TRUE, then output the simulated data.
pb	logical, if TRUE, then show progress bar.
seed	integer, seed number.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
relevanceTest	logical, if TRUE then impose a rule to stop for minimal efficiency if e <= alphaRelevance. Default FALSE.
beta	numerical in (0,1). Old parameter now replaced by the power parameter
...	further arguments to be passed to or from methods.
meanDiffMin	numeric that defines the minimal relevant mean difference, the smallest population mean difference that we would like to detect (with sufficient power).
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true. Note that prior to version 0.8.8 power <- 1-beta. The "beta" argument does not need to be specified anymore.
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)
alphaRelevance	numeric, the threshold for relevance test. Taken to be minimum of alpha and 1-power.

Value

a list with stoppingTimes and breakVector. Entries of breakVector are 0, 1. A 1 represents stopping due to exceeding nMax, and 0 due to 1/alpha threshold crossing, which implies that in corresponding stopping time is Inf.

Functions

- `sampleStoppingTimesSafeZ()`: Deprecated version of sampleStoppingTimesSaviZ

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(5), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
sampleStoppingTimesSaviZ(0.7, nSim=10, nMax=20)
```

---

sampleStoppingTimesSavi2x2

Simulate stopping times for the savi 2x2 test

Description

Draws `nSim` streams of tables on the curve `thetaA = thetaB + |propDiffMin|` for `"greater"`, `thetaA = thetaB - |propDiffMin|` for `"less"`, and on both curves for `"twoSided"`, at the worst-case baseline of each curve from `solveWorstCaseTheta2x2PropDiff()`, and records when the plain grow e-process first crosses `1 / alpha`. `nPlan` is the worst `power` quantile of the stopping time over the curves; with `power = NULL` it is skipped. There is no planning on `logOdds`.

Usage

```
sampleStoppingTimesSavi2x2(
  propDiffMin,
  na,
  nb,
  power = NULL,
  alpha = 0.05,
  alternative = c("twoSided", "less", "greater"),
  eType = c("grow"),
  nSim = 1000L,
  nMax = 10000L,
  nBoot = 10000L,
  seed = NULL,
  wantEValuesAtNMax = FALSE,
  wantSamplePaths = FALSE,
  wantSimData = TRUE,
  pb = TRUE
)
```

Arguments

propDiffMin	numeric in (-1, 1) that defines the minimal relevant difference in proportions thetaA - thetaB, the smallest difference that we would like to detect (with sufficient power).
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
power	numeric in (0, 1) that specifies the desired power, that is, the targetted chance to stop in favour of the alternative over the null hypothesis, when the alternative holds true.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.
eType	"grow", the only e-variable with planning.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi 2x2 test under continuous monitoring.
nMax	integer > 0, maximum number of data blocks in each sample path.
nBoot	integer > 0, the number of bootstrap samples; not used by the sampler itself, see computePowerSavi2x2() and computeNPlanSavi2x2().
seed	integer, seed number. Default seed=NULL yields seed=2026.
wantEValuesAtNMax	logical. If TRUE, then compute eValues at nMax. Default FALSE. Not yet implemented.
wantSamplePaths	logical, if TRUE then also outputs the sample paths.
wantSimData	logical. If TRUE, then output the simulated data. Not yet implemented.
pb	logical, if TRUE, then show progress bar.

Value

a list with thetaA, thetaB (one per curve) and, one row per curve, stoppingTimes and breakVector. Entries of breakVector are 0, 1. A 1 represents stopping due to exceeding nMax, which implies that the corresponding stopping time is Inf, and 0 due to 1/alpha threshold crossing. Further eValuesStopped, samplePaths (a list of nSim x nMax sparse matrices, or NULL), n1Vector (the block index), nPlan, and the baseline it was taken at, worstCaseThetaA, worstCaseThetaB.

savi2x2TestStat *Safe Anytime-Valid Test of Two Proportions*

Description

Tests thetaA = thetaB on a stream of 2x2 tables, one per block, with the e-process chosen by designObj[["eType"]]. Both effects are A minus B, propDiff = thetaA - thetaB and logOdds = logit(thetaA) - logit(thetaB), so "greater" means group A has the larger proportion, as in stats::t.test(). Block 1 of every e-process is replaced by the UMP conditional e-value of savi2x2TestStatUmp() when one exists at level alpha; otherwise it keeps its plain factor, with a warning.

Usage

```
savi2x2TestStat(
  ya,
  yb,
  designObj = NULL,
  wantCi = FALSE,
  wantConfidenceSequence = FALSE,
  ciValue = NULL
)
```

Arguments

ya	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and na, the number of observations in group A per block.
yb	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and nb, the number of observations in group B per block.
designObj	an object obtained from designSavi2x2(), which also supplies na and nb.
wantCi	default FALSE, compute a confidence interval.
wantConfidenceSequence	logical that can be set to true when the user wants a savi confidence sequence to be estimated, one row per block; takes precedence over wantCi.
ciValue	numeric representing the confidence level. Default ciValue=NULL yields ciValue = 1 - alpha

Details

- "eBeta" (propDiff): independent Beta posterior means of thetaA and thetaB from the previous blocks, against the pooled null mean.
- "grow" (propDiff or logOdds, whichever minimal effect the design holds): the alternative restricted to the signed minimal effect; "twoSided" averages the cumulative e-processes of both signs.

- `"eGauss"` (`logOdds`): the conditional likelihood of Fisher's noncentral hypergeometric distribution mixed under the design's Normal prior on a `logOdds` grid, against the hypergeometric null. The mixture may include the current block because it conditions on the block's total.

Only `"eBeta"` and `"eGauss"` give a confidence interval or sequence, each on its own effect.

Value

Returns an object of class 'saviTest'. An object of class 'saviTest' is a list containing at least the following components:

n The realised sample size(s): `c(na = sum(na), nb = sum(nb), nBlocks = length(ya))`.

eValue the e-value of the savi test on all blocks; reject when it is at least `1 / alpha`.

eValueVec the realised e-values after each block, element `i` on blocks `1..i`.

n1Vec the block index, used for plotting.

estimate the estimated proportions `c(thetaA, thetaB)`, pooled over all blocks.

confSeq a savi confidence interval for the design's effect at level `ciValue` on all blocks; `NA` when the set is empty, `NULL` for "grow" or when no interval was requested.

confSeqMatrix with `wantConfidenceSequence`, the savi confidence sequence as an `nBlocks x 2` matrix, row `i` the interval on blocks `1..i`; `confSeq` is its last row. With `runningIntersection` the rows are nested and stay `NA` after the first empty row.

ciValue the confidence level used.

betaParameter for "eBeta" only, `list(betaA1, betaA2, betaB1, betaB2)`, the Beta prior updated with all blocks.

alternative any of "twoSided", "greater", "less" copied from the design.

h0 the null value copied from the design.

dataName a character string giving the name(s) of the data.

designObj an object of class "saviDesign" described in `designSavi2x2()`.

call the expression with which this function is called.

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(5), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Turner, R., Ly, A., & Grünwald, P. D. (2024). Generic e-variables for exact sequential k-sample tests that allow for optional stopping. *Journal of Statistical Planning and inference* **230**, 106116, https://doi.org/10.1016/j.jspi.2023.106116.

Turner, R., & Grünwald, P. D. (2024). Exact anytime-valid confidence intervals for contingency tables and beyond. *Statistics and Probability letters*, **198**, 109835, https://doi.org/10.1016/j.spl.2023.109835.

Examples

```
designObj <- designSavi2x2(na = 10, nb = 10, eType = "eBeta")
ya <- c(8, 7, 9, 6)
yb <- c(4, 5, 3, 5)
savi2x2TestStat(ya, yb, designObj = designObj)
```

savi2x2TestStatUmp *UMP conditional e-value of one 2x2 table*

Description

The e-value of the conditional test on one table: the likelihood of `logLikelihoodFNCH()` at the uniformly most powerful log odds ratio of `solveUmpLogOdds()` over the hypergeometric null. `"twoSided"` averages the two one-sided e-values. The test uses it in place of block 1's plain factor.

Usage

```
savi2x2TestStatUmp(
  ya,
  yb,
  na,
  nb,
  alpha,
  alternative = c("twoSided", "greater", "less")
)
```

Arguments

`ya`	positive observations/ events per data block in group A: a numeric with integer values between (and including) 0 and `na`, the number of observations in group A per block.
`yb`	positive observations/ events per data block in group B: a numeric with integer values between (and including) 0 and `nb`, the number of observations in group B per block.
`na`	number of observations in group A per data block, one value or one per block.
`nb`	number of observations in group B per data block, one value or one per block.
`alpha`	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.

Value

A numeric, or `NULL` when no one-sided UMP log odds ratio exists at level `alpha`.

saviAnovaOneB *Safe Anytime-Valid One-Way Anova*

Description

To compute equality of all means in K groups. Still in beta stage.

Usage

```
saviAnovaOneB(formula, data, subset, na.action, parameter, sigma = NULL, ...)
```

Arguments

formula	a formula of the form lhs ~ rhs where lhs is a numeric variable giving the data values and rhs a factor with at least two levels/groups.
data	a data frame (or similar: see model.frame()) containing the variables in the formula. By default the variables are taken from environment(formula).
subset	an optional vector specifying a subset of observations to be used.
na.action	a function which indicates what should happen when the data contain NAs. Defaults to getOption("na.action").
parameter	numeric > 0 representing the scaling the in the multivatiate Gaussian prior
sigma	numeric > 0 if a population standard deviation is assumed to be known. Default NULL to convey that the it is unknown.
...	further arguments to be passed to or from methods.

Value

a list

Examples

```
# Data
plantData <- data.frame(
  treatment = factor(rep(c("control", "fertiliserA", "fertiliserB"), each = 10)),
  height = c(
    c(12, 14, 11, 13, 12, 15, 13, 11, 14, 12), # Control group
    c(18, 20, 19, 22, 21, 20, 17, 19, 21, 23), # Fertiliser A group
    c(15, 17, 16, 15, 18, 14, 16, 17, 15, 16)  # Fertiliser B group
  )
)

saviAnovaOneB(height ~ treatment, data = plantData, parameter=1)
```

saviAnovaStat *Safe Anytime-Valid One-Way Anova based on summary statistics*

Description

To compute equality of all means in K groups. Still in beta stage.

Usage

```
saviAnovaStat(sumX, n, sumSquares = NULL, parameter, sigma = NULL)
```

Arguments

sumX	vector of numerics representing the sum of the observations of the K groups
n	vector of integers representing the sample sizes of the K groups
sumSquares	vector of numerics representing the sum of the square of the observations of the K groups
parameter	numeric > 0 representing the scaling the in the multivatiate Gaussian prior
sigma	numeric > 0 if a population standard deviation is assumed to be known. Default NULL to convey that the it is unknown.

Value

a list with the e-value

Examples

```
saviAnovaStat(1:6, 1:6, (1:6)^2, parameter=1)
```

saviTTestStat *Computes E-Values Based on the T-Statistic*

Description

A summary stats version of saviTTest() with the data replaced by t, n1 and n2, and the design object by parameter

This is saviTTestStat() based on t-densities instead of hypergeometric functions.

Evidence for practical equivalence requires the e-value to be small, i.e. smaller than alphaRelevance. If the alternative holds true, i.e. deltaTrue >= relevanceSize, then there is no more than alphaRelevance probability of ever seeing eRelevance <= alphaRelevance.

Usage

```
saviTTestStat(
  t,
  n1,
  n2 = NULL,
  parameter,
  alternative = c("twoSided", "less", "greater"),
  tDensity = FALSE,
  paired = FALSE,
  nuMin = 2,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  ...
)

saviTTestStatNEffNu(
  t,
  nEff,
  nu,
  parameter,
  alternative = c("twoSided", "less", "greater"),
  tDensity = FALSE,
  paired = FALSE,
  nuMin = 2,
  eType = c("mom", "eGauss", "imom", "eCauchy", "grow", "lai"),
  ...
)

saviTTestStatNEffNuMom(
  t,
  nEff,
  nu,
  parameter,
  alternative = c("twoSided", "less", "greater"),
  tDensity = FALSE,
  paired = FALSE,
  k = 1,
  ...
)

saviTTestStatNEffNuGrow(
  t,
  nEff,
  nu,
  parameter,
  alternative = c("twoSided", "less", "greater"),
  tDensity = FALSE,
  paired = FALSE,
  ...
```

```
)

saviTTestStatTDensity(
  t,
  parameter,
  nu,
  nEff,
  alternative = c("twoSided", "less", "greater"),
  paired = FALSE,
  ...
)

saviRelevanceTStatNEffNu(
  t,
  nEff,
  nu,
  parameter = NULL,
  alternative = c("twoSided", "less", "greater"),
  eType = "grow",
  tDensity = FALSE,
  paired = FALSE,
  relevanceSize,
  nuMin = 2,
  ...
)
```

Arguments

t	numeric that represents the observed t-statistic.
n1	integer that represents the size of the (first) sample. Default n2=NULL implies a one-sample T-test.
n2	an optional integer that represents the size of the second sample, which implies a two-sample t-statistic
parameter	numeric > 0, the savi test defining parameter, see matchEParameterWith for details.
alternative	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
tDensity	Uses the the representation of the savi T-test as the likelihood ratio of t densities.
paired	a logical, if TRUE then pair the data.
nuMin	numeric > 0, the minimum degrees of freedom under which the results are trivial, thus, 1.
eType	character one of "mom", "grow", "eGauss", and "eCauchy". "mom" is default and uses a non-local moment prior with bump(s) at meanDiffMin, "grow" uses point prior(s) at meanDiffMin, "eGauss" a zero-centred normal prior, "eCauchy" a zero centred Cauchy prior.
...	further arguments to be passed to or from methods, but mainly to perform do.calls.

nEff	numeric > 0, the effective sample size. For one sample tests, this is just n.
nu	numeric > 0, the degrees of freedom.
k	the moment used for the non-local moment prior. Default 1
relevanceSize	numeric, the minimal clinical relevant mean difference that we do not want to miss under the alternative. Default relevanceSize=NULL implies relevanceSize=abs(meanDiffMin)

Value

Returns a numeric that represent the e10, that is, the e-value in favour of the alternative over the null

References

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Pérez-Ortiz, M. F., Lardy, T., de Heide, R., & Grünwald, P. D. (2024). E-statistics, group invariance and anytime valid testing. *The Annals of Statistics*, **52**(*4*), 1410-1432, http://dx.doi.org/10.1214/24-AOS2394.

Wang, H., & Ramdas, A. (in press). Anytime-valid t-tests and confidence sequences for Gaussian means with unknown variance. *Sequential Analysis*, https://doi.org/10.48550/arXiv.2310.03722.

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Pérez-Ortiz, M. F., Lardy, T., de Heide, R., & Grünwald, P. D. (2024). E-statistics, group invariance and anytime valid testing. *The Annals of Statistics*, **52**(*4*), 1410-1432, http://dx.doi.org/10.1214/24-AOS2394.

Wang, H., & Ramdas, A. (in press). Anytime-valid t-tests and confidence sequences for Gaussian means with unknown variance. *Sequential Analysis*, https://doi.org/10.48550/arXiv.2310.03722.

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Pérez-Ortiz, M. F., Lardy, T., de Heide, R., & Grünwald, P. D. (2024). E-statistics, group invariance and anytime valid testing. *The Annals of Statistics*, **52**(*4*), 1410-1432, http://dx.doi.org/10.1214/24-AOS2394.

Wang, H., & Ramdas, A. (in press). Anytime-valid t-tests and confidence sequences for Gaussian means with unknown variance. *Sequential Analysis*, https://doi.org/10.48550/arXiv.2310.03722.

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Pérez-Ortiz, M. F., Lardy, T., de Heide, R., & Grünwald, P. D. (2024). E-statistics, group invariance and anytime valid testing. *The Annals of Statistics*, **52**(*4*), 1410-1432, http://dx.doi.org/10.1214/24-AOS2394.

Wang, H., & Ramdas, A. (in press). Anytime-valid t-tests and confidence sequences for Gaussian means with unknown variance. *Sequential Analysis*, https://doi.org/10.48550/arXiv.2310.03722.

Grünwald, P. D., de Heide, R., & Koolen, W. (2024). Safe testing. *Journal of the Royal Statistical Society. Series B (Methodological),* **86**(*5*), 1091-1128. (With discussions), https://doi.org/10.1093/jrsssb/qkae011.

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Pérez-Ortiz, M. F., Lardy, T., de Heide, R., & Grünwald, P. D. (2024). E-statistics, group invariance and anytime valid testing. *The Annals of Statistics*, **52**(*4*), 1410-1432, http://dx.doi.org/10.1214/24-AOS2394.

Wang, H., & Ramdas, A. (in press). Anytime-valid t-tests and confidence sequences for Gaussian means with unknown variance. *Sequential Analysis*, https://doi.org/10.48550/arXiv.2310.03722.

Examples

```
saviTTestStat(t=1, n1=100, parameter=0.4)
saviTTestStat(t=3, n1=100, parameter=0.3)
# evidence for the alternative over minimal efficacy
saviRelevanceTStatNEffNu(t=3, nEff=100, nu=60, parameter=0.4)
# evidence for minimal efficacy over the alternative
saviRelevanceTStatNEffNu(t=0.35, nEff=100, nu=60, parameter=0.4)
```

---

selectivelyContinueZOrTTestData

Helper function to demonstrate the selective continuation of paired z-tests

---

Description

Helper function to demonstrate the selective continuation of paired z-tests

Usage

```
selectivelyContinueZOrTTestData(
  designObj,
  deltaTrue = NULL,
  testName = c("Z-Test", "T-Test"),
```

```
  n1New,
  muGlobal,
  sigma,
  meanDiffTrue = NULL,
  nSim = 1000L,
  eValuesOld,
  eOverOld,
  trackCrossingOld,
  firstPassageTimeOld,
  eStoppedOld,
  seed = NULL
)
```

Arguments

designObj	an object obtained from designSaviZ().
deltaTrue	numeric, the value of the true standardised effect size (test-relevant parameter). This argument is used by designSaviT() with deltaTrue <- deltaMin
testName	character string either "Z-Test" or "T-Test".
n1New	Number of samples in the follow-up study
muGlobal	numeric, population grand mean
sigma	numeric > 0 representing the assumed population standard deviation used to scale the data.
meanDiffTrue	numeric, data governing mean difference used for simulations. Default meanDiffTrue=meanDiffMin.
nSim	integer > 0, the number of simulations needed to compute power or the number of samples paths for the savi z test under continuous monitoring.
eValuesOld	Matrix of eValues of the previous studies with nSim number of rows
eOverOld	Vector of 0s and 1s, where 1 indicates that
trackCrossingOld	Vector of same number of columns as eValuesOld providing the number of studies that got rejected at the indeced time
firstPassageTimeOld	Vector indicating the first passage time. If Inf, then not yet passed the threshold of 1/alpha
eStoppedOld	The stopped e-value either at the time it crosses the threshold, or at the previous nPlan
seed	integer, seed number.

Value

a list with the new e-values amongst other

`setLowAndHighEsTrueZ` *Helper function to determine the lower and upper bound for the search space for standardised meanDiffMin*

Description

Helper function to determine the lower and upper bound for the search space for standardised meanDiffMin

Usage

```
setLowAndHighEsTrueZ(
  nEff,
  eType = "mom",
  alternative = "twoSided",
  lowEsTrue = 1e-07,
  highEsTrue = 0.001
)
```

Arguments

`nEff`	numeric > 0, the effective sample size.
`eType`	character string, one of "eCauchy", "eGauss", "grow", "freq", "credibleInterval". This is somewhat a misnomer as "freq" and "credibleInterval" do not correspond to e-value tests. "eCauchy" yields an anytime-valid confidence interval based on a Cauchy mixture, whereas "eGauss" yields an anytime-valid confidence interval based on a Gaussian mixture. "grow" yields an anytime-valid confidence interval based on a mixture of point masses at the minimal clinically relevant mean difference. This confidence interval unfortunately does not shrink as the sample size tends to infinity. "freq" yields the standard unsafe frequentist confidence interval, which is not safe. "credibleInterval" yields the Bayesian credible interval based on a conjugate prior as is usual in Bayesian analysis. This interval is also not safe.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less".
`lowEsTrue`	numeric, lower bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.
`highEsTrue`	numeric, upper bound for the candidate set of the targeted minimal clinically relevant effect size for scenario 3.a.

Value

a list of low and high values for the parameter

Examples

```
setLowAndHighEsTrueZ(10)
```

`setSafeStatsPlotOptionsAndReturnOldOnes` *Sets 'safestats' Plot Options and Returns the Current Plot Options.*

Description

Sets 'safestats' Plot Options and Returns the Current Plot Options.

Usage

```
setSafeStatsPlotOptionsAndReturnOldOnes(...)
```

Arguments

`...`	further arguments to be passed to or from methods.

Value

Returns a list with the user specified plot options.

Examples

```
oldPar <- setSafeStatsPlotOptionsAndReturnOldOnes()
graphics::plot(1:10, 1:10)
setPar <- graphics::par(oldPar)
```

`simulateCoverageDifferenceTwoProportions` *Simulate the coverage of a savi confidence sequence for differences between proportions for a given distribution and savi design.*

Description

Simulate the coverage of a savi confidence sequence for differences between proportions for a given distribution and savi design.

Usage

```
simulateCoverageDifferenceTwoProportions(
  successProbabilityA,
  trueDelta,
  saviDesign,
  precision = 100,
  M = 1000,
  numberForSeed = NA
)
```

Arguments

successProbabilityA	probability of observing a success in group A.
trueDelta	difference in probability between group A and B.
saviDesign	a savi test design for two proportions retrieved through designSaviTwoProportions().
precision	precision of the grid to search over for the confidence sequence bounds. Default 100.
M	number of simulations to carry out. Default 1000.
numberForSeed	number for seed to set, default NA.

Value

the proportion of simulations where the trueDelta was included in the confidence sequence.

Examples

```
balancedSaviDesign <- designSaviTwoProportions(na = 1,
                                               nb = 1,
                                               nBlocksPlan = 20)
simulateCoverageDifferenceTwoProportions(successProbabilityA = 0.2,
                                         trueDelta = 0,
                                         saviDesign = balancedSaviDesign,
                                         M = 100,
                                         precision = 20,
                                         numberForSeed = 1082021)
```

---

simulateIncorrectStoppingTimesFisher

Simulate incorrect optional stopping with fisher's exact test's p-value as the stopping rule.

Description

Simulate incorrect optional stopping with fisher's exact test's p-value as the stopping rule.

Usage

```
simulateIncorrectStoppingTimesFisher(
  thetaA,
  thetaB,
  alpha,
  na,
  nb,
  maxSimStoptime = 10000,
  M = 1000,
  numberForSeed = NULL
)
```

Arguments

thetaA	Bernoulli distribution parameter in group A
thetaB	Bernoulli distribution parameter in group B
alpha	Significance level
na	number of observations in group a per data block
nb	number of observations in group b per data block
maxSimStoptime	maximal number of blocks to sample in each experiment
M	Number of simulations to carry out, default 1e3.
numberForSeed	number for seed to set, default NULL.

Value

list with stopping times and rejection decisions.

Examples

```
simulateIncorrectStoppingTimesFisher(thetaA = 0.3,
                                     thetaB = 0.3,
                                     alpha = 0.05,
                                     na = 1,
                                     nb = 1,
                                     M = 10,
                                     maxSimStoptime = 100,
                                     numberForSeed = 251)
```

---

simulateOptionalStoppingScenarioTwoProportions

Simulate an optional stopping scenario according to a savi design for two proportions

---

Description

Simulate an optional stopping scenario according to a savi design for two proportions

Usage

```
simulateOptionalStoppingScenarioTwoProportions(saviDesign, M, thetaA, thetaB)
```

Arguments

saviDesign	a 'saviDesign' object obtained through designSaviTwoProportions().
M	integer, the number of data streams to sample.
thetaA	Bernoulli distribution parameter in group A
thetaB	Bernoulli distribution parameter in group B

Value

list with the simulation results of the savi test under optional stopping with the following components:

powerOptioStop Proportion of sequences where H0 was rejected

nMean Mean stopping time

probLessNDesign Proportion of experiments stopped before nBlocksPlan was reached

lowN Minimum stopping time

eValues All achieved E values

allN All stopping times

allSaviDecisions Decisions on rejecting H0 for each M

allRejectedN Stopping times of experiments where H0 was rejected

Examples

```
balancedSaviDesign <- designSaviTwoProportions(na = 1,
                                               nb = 1,
                                               nBlocksPlan = 30)
optionalStoppingSimulationResult <- simulateOptionalStoppingScenarioTwoProportions(
  saviDesign = balancedSaviDesign,
  M = 1e2,
  thetaA = 0.2,
  thetaB = 0.5
)
```

---

simulateTwoProportions

Compare Different Hyperparameter Settings for Savi Tests of Two Proportions.

Description

Simulates for a range of divergence parameter values (differences or log odds ratios) the worst-case stopping times (i.e., number of data blocks collected) and expected stopping times needed to achieve the desired power for each hyperparameter setting provided.

Usage

```
simulateTwoProportions(
  hyperparameterList,
  alternativeRestriction = c("none", "difference", "logOddsRatio"),
  deltaDesign = NULL,
  alpha,
  beta,
  na,
```

```
  nb,
  deltamax = 0.9,
  deltamin = 0.1,
  deltaGridSize = 8,
  M = 100,
  maxSimStoptime = 10000,
  thetaAgridSize = 8
)
```

Arguments

hyperparameterList	list object, its components hyperparameter lists with a format as described in designSaviTwoProportions().
alternativeRestriction	a character string specifying an optional restriction on the alternative hypothesis; must be one of "none" (default), "difference" (difference group mean b minus group b) or "logOddsRatio" (the log odds ratio between group means b and a).
deltaDesign	optional; when using a restricted alternative, the value of the divergence measure used. Either a numeric between -1 and 1 for a restriction on difference, or a real for a restriction on the log odds ratio.
alpha	numeric in (0, 1) that specifies the tolerable type I error control –independent on n– that the designed test has to adhere to. Note that it also defines the rejection rule e10 >= 1/alpha.
beta	numeric in (0, 1) that specifies the tolerable type II error control in the study. Necessary to calculate the worst case stopping time.
na	number of observations in group a per data block
nb	number of observations in group b per data block
deltamax	maximal effect size to calculate power for; between -1 and 1 for designs without restriction or a restriction on difference; real number for a restriction on the log odds ratio. Default 0.9.
deltamin	minimal effect size to calculate power for; between -1 and 1 for designs without restriction or a restriction on difference; real number for a restriction on the log odds ratio. Default 0.1.
deltaGridSize	numeric, positive integer: size of grid of delta values worst case and expected sample sizes are simulated for.
M	number of simulations used to estimate sample sizes. Default 100.
maxSimStoptime	maximal stream length in simulations; when the e value does not reach the rejection threshold before the end of the stream, the maximal stream length is returned as the stopping time. Default 1e4.
thetaAgridSize	numeric, positive integer: size of the grid of probability distributions examined for each delta value to find the worst case sample size over.

Value

Returns an object of class "savi2x2Sim". An object of class "savi2x2Sim" is a list containing at least the following components:

simData A data frame containing simulation results with worst case and expected stopping times for each hyperparameter setting, for the specified or default range of effect sizes.

alpha the significance threshold used in the simulations

beta the type-II error control used in the simulations

deltaDesign the value of restriction on the alternative hypothesis parameter space used for the E variables in the simulations

restriction the type of restriction used for the E variables in the simulation

hyperparameters list of the hyperparameters tested in the simulation

Examples

```
priorList1 <- list(betaA1 = 10, betaA2 = 1, betaB1 = 1, betaB2 = 10)
priorList2 <- list(betaA1 = 0.18, betaA2 = 0.18, betaB1 = 0.18, betaB2 = 0.18)
priorList3 <- list(betaA1 = 1, betaA2 = 1, betaB1 = 1, betaB2 = 1)

simResult <- simulateTwoProportions(
  hyperparameterList = list(priorList1, priorList2, priorList3),
  alternativeRestriction = "none",
  alpha = 0.1, beta = 0.2, na = 1, nb = 1,
  deltamax = -0.4, deltamin = -0.9, deltaGridSize = 3,
  M = 10
  )

print(simResult)
plot(simResult)
```

---

solveRIPr2x2PropDiff *Reverse information projection onto a fixed propDiff*

Description

The null pair (`nullThetaA, nullThetaA - propDiff`) closest in KL divergence to (`thetaA, thetaB`) for one block: the root of the derivative of `na KL(thetaA || nullThetaA) + nb KL(thetaB || nullThetaB)`.

Usage

```
solveRIPr2x2PropDiff(thetaA, thetaB, na, nb, propDiff, tol = 1e-12)
```

Arguments

thetaA, thetaB	numeric in (0, 1), the alternative's proportions.
na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
propDiff	numeric in (-1, 1), the null difference thetaA - thetaB.
tol	numeric > 0, the tolerance of stats::uniroot(), also kept from the edges of the search interval.

Value

A numeric, nullThetaA.

---

solveUmpLogOdds *Uniformly most powerful log odds ratio of a one-sided conditional test*

Description

Solves the conditional KL divergence of the alternative at logOdds from logOddsNull equal to log(1 / alpha), searching up to searchBound from logOddsNull in the direction of alternative.

Usage

```
solveUmpLogOdds(
  na,
  nb,
  totalSuccesses,
  alpha,
  alternative = c("greater", "less"),
  logOddsNull = 0,
  searchBound = 100
)
```

Arguments

na	number of observations in group A per data block, one value or one per block.
nb	number of observations in group B per data block, one value or one per block.
totalSuccesses	nonnegative integer, the total successes ya + yb in the table, at most na + nb.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.
alternative	"greater" or "less", the side searched.
logOddsNull	numeric, the null log odds ratio.
searchBound	numeric > 0, the width of the search interval.

Value

A numeric, or NULL with a warning when the target is not reached on the search interval.

`solveWorstCaseTheta2x2PropDiff` *Worst-case baseline of the propDiff grow test*

Description

The pair on the curve of each alternative at which the e-process grows slowest, so a block count planned there holds for every baseline. The expected log e-increment per block is the KL divergence of the truth from its pooled null projection,

```
R(theta) = nLow KL(theta || theta0) + nHigh KL(theta + d || theta0),
theta0   = theta + nHigh d / (nLow + nHigh),
```

with `theta` the lower proportion, `nLow` its group size and `nHigh` the size of the group at `theta + d`. Its minimiser is the root of

```
nLow logit(theta) + nHigh logit(theta + d) = (nLow + nHigh) logit(theta0),
```

summed over blocks when the sizes vary per block. Equal sizes give exactly the midpoint `(1 - d) / 2`; in general the root is `(1 - d) / 2 + d (nHigh - nLow) / (6 (nLow + nHigh)) + O(d^3)`. The rate ignores the first-block UMP factor, which moves the simulated worst case when one group is very small.

Usage

```
solveWorstCaseTheta2x2PropDiff(
  propDiffMin,
  na,
  nb,
  alternative = c("twoSided", "greater", "less")
)
```

Arguments

`propDiffMin`	numeric in (-1, 1) that defines the minimal relevant difference in proportions `thetaA - thetaB`, the smallest difference that we would like to detect (with sufficient power).
`na`	number of observations in group A per data block, one value or one per block.
`nb`	number of observations in group B per data block, one value or one per block.
`alternative`	a character string specifying the alternative hypothesis. Must be one of "twoSided" (default), "greater" or "less", where "greater" means that group A has the larger proportion.

Details

`"greater"`: `thetaA = thetaB + d`, so `thetaB` is the lower proportion and `nLow = nb`. `"less"`: `thetaA = thetaB - d`, so `thetaA` is the lower one and `nLow = na`. `"twoSided"` gives both, greater first.

Value

A data frame with one row per curve and columns `thetaA`, `thetaB`.

`subjectiveBfZStat` *A subjective Bayes factor for the two-sample Z-test*

Description

Based on conjugate priors with a total of 6 hyperparameters.

Usage

```
subjectiveBfZStat(
  x1,
  s21,
  n1,
  x2,
  s22,
  n2,
  sigma = 1,
  a1 = 5,
  b1 = 2,
  a2 = -5,
  b2 = 2,
  a0 = 0,
  b0 = 1,
  log = FALSE
)
```

Arguments

`x1`	numeric, sample mean of group 1
`s21`	numeric, sample variance of group 1
`n1`	integer sample size of group 1
`x2`	numeric, sample mean of group 2
`s22`	numeric, sample variance of group 2
`n2`	integer sample size of group 2
`sigma`	numeric > 0, population standard deviation
`a1`	numeric, prior mean of the population mean mu1 of group 1
`b1`	numeric > 0, prior standard deviation of the population mean mu1 of group 1
`a2`	numeric, prior mean of the population mean mu2 of group 2
`b2`	numeric > 0, prior standard deviation of the population mean mu2 of group 2
`a0`	numeric, prior mean of the overall population mean mu0 of both groups
`b0`	numeric > 0, prior standard deviation of the population mean mu0 of both groups
`log`	logical, default FALSE, if TRUE then return logarithm of the subjective Bayes factor outcome

Value

numeric > 0 representing the subjective Bayes factor outcome in favour of the alternative over the null

Examples

```
subjectiveBfZStat(5.2, 2, 3, 3.4, 2, 12)
```

---

tryOrFailWithNA *Tries to Evaluate an Expression and Fails with* NA

Description

The evaluation fails with NA by default, but it is also able to fail with other values.

Usage

```
tryOrFailWithNA(expr, value = NA_real_)
```

Arguments

expr	Expression to be evaluated.
value	Return value if there is an error, default is NA_real_.

Value

Returns the evaluation of the expression, or value if it doesn't work out.

---

tTestWidthDerivative *Internal function to solve the smallest width of an eGauss t-test*

Description

Internal function to solve the smallest width of an eGauss t-test

Usage

```
tTestWidthDerivative(g, nEff, nu, alpha = 0.05)
```

Arguments

g	prior variance of the eGauss t-test
nEff	numeric > 0, the effective sample size. For one sample tests, this is just n.
nu	numeric > 0, the degrees of freedom.
alpha	numeric in (0, 1) that specifies the tolerable type I error and the null rejection rule e >= 1/alpha.

Value

a number that should be zero when g is optimal

References

Ly, A, Boehm, Grünwald, P. D., Ramdas, A., & van Ravenzwaaij, D. (2024). Safe Anytime-Valid Inference: Practical maximally flexible sampling designs for experiments based on e-values. *PsyArXiv Preprint*, https://doi.org/10.31234/osf.io/h5vae.

Examples

```
tTestWidthDerivative(1, 1, 1)
```

---

zMarg *Logarithmic marginal likelihood of the normal with conjugate priors*

Description

Logarithmic marginal likelihood of the normal with conjugate priors

Usage

```
zMarg(n, x, s2, a = 0, b = 1, sigma = 1)
```

Arguments

n	integer, sample size
x	numeric, sample mean
s2	numeric > 0, sample variance
a	numeric, prior mean of the population mean mu
b	numeric > 0, prior standard deviation of the population mean mu
sigma	numeric > 0, population standard deviation

Value

numeric, the logarithm of the marginal likelihood

Examples

```
zMarg(12, 0.3, 2)
```

Index

addCite, 5

boot, *13*, *15*, *30*, *32*, *44*, *46*
boot::boot(), *48*, *50*

ceil, 5
checkAndReturnEsMinParameterSide, 6
checkAndReturnNPlan, 7
checkAndReturnsEsMinParameterSide, 7
checkDoubleArgumentsDesignObject, 8
compute1F1AllVersions, 9
computeBetaBatchSafeZ, 10
computeBetaBatchSaviZ *(*computeBetaBatchSafeZ*)*, 10
computeBetaBootstrapper, 11
computeBetaSafeT, 12
computeBetaSafeZ, 14
computeBootObj, 16
computeConfidenceBoundForLogOddsTwoProportions, 17
computeConfidenceBoundsForDifferenceTwoProportions, 18
computeConfidenceInterval2x2LogOdds, 19
computeConfidenceInterval2x2LogOdds(), *24*
computeConfidenceInterval2x2PropDiff, 20
computeConfidenceInterval2x2PropDiff(), *25*
computeConfidenceIntervalT, 21
computeConfidenceIntervalZ, 22
computeConfidenceSequence2x2LogOdds, 24
computeConfidenceSequence2x2PropDiff, 25
computeConjugateCredibleIntervalTwoSampleT, 26
computeEsMinSavi2x2, 27
computeLogrankBetaFrom, 29
computeLogrankNEvents, 31
computeLogrankZ, 33
computeMinEsBatchSafeT *(*designSafeT*)*, 68
computeMinEsBatchSafeZ, 34
computeMinEsBatchSaviT, 36
computeMinEsBatchSaviZ *(*computeMinEsBatchSafeZ*)*, 34
computeNEff, 37
computeNPlanBatchSafeT *(*designSafeT*)*, 68
computeNPlanBatchSafeZ, 38
computeNPlanBatchSaviT, 40
computeNPlanBatchSaviZ *(*computeNPlanBatchSafeZ*)*, 38
computeNPlanBootstrapper, 41
computeNPlanSafeT, 42
computeNPlanSafeZ, 44
computeNPlanSavi2x2, 47
computeNPlanSavi2x2(), *151*
computeNPlanSaviT *(*computeNPlanSafeT*)*, 42
computeNPlanSaviZ *(*computeNPlanSafeZ*)*, 44
computePowerBootstrapper, 48
computePowerSavi2x2, 49
computePowerSavi2x2(), *27*, *151*
computePowerSaviT *(*computeBetaSafeT*)*, 12
computePowerSaviZ *(*computeBetaSafeZ*)*, 14
computeStatsForLogrank, 50
computeZTSumStats, 51
conjugateBfTStat, 52
conjugateBfTStatOld, 54
constructSafeDesignObj, 55
constructSafeTestObj, 56
constructSampleStoppingTimesList, 57
constructSaviDesignObj *(*constructSafeDesignObj*)*, 55
constructSaviTestObj *(*constructSafeTestObj*)*, 56

defineTTestN, 57

designFreqT, 58
designFreqZ, 59
designSafe1aHelper, 61
designSafe2Helper, 62
designSafeLogrank, 63
designSafeLogrank2WantBeta, 66
designSafeT, 68
designSafeT1aWantNPlan, 72
designSafeT2WantBeta (designSafeT), 68
designSafeT3bWantParameter (designSafeT), 68
designSafeT3WantEsMin (designSafeT), 68
designSafeTwoProportions, 74
designSafeZ, 77
designSafeZ1aWantNPlan, 81
designSafeZ2WantBeta, 83
designSafeZ3bWantParameter, 85
designSafeZ3WantEsMin, 87
designSavi1aHelper (designSafe1aHelper), 61
designSavi2Helper (designSafe2Helper), 62
designSavi2x2, 88
designSavi2x2(), *152, 153*
designSaviLogrank, *129*
designSaviLogrank (designSafeLogrank), 63
designSaviLogrank2WantBeta (designSafeLogrank2WantBeta), 66
designSaviT, *132, 133*
designSaviT (designSafeT), 68
designSaviT1aWantNPlan (designSafeT1aWantNPlan), 72
designSaviT2WantPower, 92
designSaviT3bWantParameter, 94
designSaviT3WantEsMin, 95
designSaviTwoProportions, *17*, *19*, *120*, *136, 137*, *164, 165*, *167*
designSaviTwoProportions (designSafeTwoProportions), 74
designSaviZ, *138, 139*, *161*
designSaviZ (designSafeZ), 77
designSaviZ(), *90*
designSaviZ1aWantNPlan (designSafeZ1aWantNPlan), 81
designSaviZ2WantPower (designSafeZ2WantBeta), 83
designSaviZ3bWantParameter (designSafeZ3bWantParameter), 85
designSaviZ3WantEsMin (designSafeZ3WantEsMin), 87

extractNameFromArgs, 96

fnchLogPartition, 97

generateNormalData, 97
generateSurvData, 98
genhypergeo, *9*
getArgs, 100
getNameAlternative, 100
getNameTestType, 101

hist, *116*

isTryError, 101

lcosh, 102
logEValueVec2x2LogOddsEGauss, 102
logEValueVec2x2LogOddsGrow, 103
logEValueVec2x2PropDiffEBeta, 104
logEValueVec2x2PropDiffGrow, 105
logLikelihoodFNCH, 106
logLikelihoodFNCH(), *19*, *103*, *154*
logrankSingleEExact, 107
logrankSingleZ, 108

makeRunningIntersection, 109
matchAlphaRelevanceWith, 110
matchEParameterWith, *10*, *12*, *15*, *22, 23*, *35*, *37*, *39*, *41*, *43*, *45*, *69–71*, *73*, *78–80*, *82*, *84*, *86, 87*, *93, 94*, *96*, *111*, 111, *112*, *142*, *146*, *149*, *158*
matchEsMinWith, 112
matchPowerWith, 113
matchRelevanceParameterWith, 113
model.frame, *133*, *139*

plot.safe2x2Sim, 114
plot.safeDesign, 115
plot.safeTest, 117
plot.savi2x2Sim (plot.safe2x2Sim), 114
plot.saviDesign (plot.safeDesign), 115
plot.saviTest (plot.safeTest), 117
plotConfidenceSequenceTwoProportions, 120

plotHistogramDistributionStoppingTimes, 121
polygon, *119*
predictiveThetas2x2, 122
predictiveThetas2x2(), *20*, *104*
print.htest, *124*
print.power.htest, *124*
print.safe2x2Sim, 123
print.safeDesign, 124
print.safeTest, 124
print.savi2x2Sim *(*print.safe2x2Sim*)*, 123
print.saviDesign *(*print.safeDesign*)*, 124
print.saviTest *(*print.safeTest*)*, 124
pValueFromZStat, 125
pValueZTest *(*pValueFromZStat*)*, 125

returnOne, 127
rLogrank, 127
rweibull, *99*

safe.prop.test *(*safeTwoProportionsTest*)*, 135
safe.t.test *(*safeTTestStat*)*, 131
safe.z.test *(*safeZTest*)*, 138
safeLogrankTest, 128
safeLogrankTestStat *(*safeLogrankTest*)*, 128
safeTTest *(*safeTTestStat*)*, 131
safeTTestStat, 131
safeTTestStatNEffNu *(*safeTTestStat*)*, 131
safeTTestStatNEffNuMom *(*safeTTestStat*)*, 131
safeTTestStatTDensity *(*safeTTestStat*)*, 131
safeTwoProportionsTest, 135
safeZTest, 138
safeZTestStat, 141
sampleLogrankStoppingTimes, 143
sampleStoppingTimesSafeT, 145
sampleStoppingTimesSafeZ, 148
sampleStoppingTimesSavi2x2, 150
sampleStoppingTimesSavi2x2(), *47*, *49*, *90*
sampleStoppingTimesSaviT *(*sampleStoppingTimesSafeT*)*, 145
sampleStoppingTimesSaviZ *(*sampleStoppingTimesSafeZ*)*, 148
savi.prop.test *(*safeTwoProportionsTest*)*, 135
savi.t.test *(*safeTTestStat*)*, 131
savi.z.test *(*safeZTest*)*, 138
savi2x2TestStat, 152
savi2x2TestStat(), *90*
savi2x2TestStatUmp, 154
savi2x2TestStatUmp(), *152*
saviAnovaOneB, 155
saviAnovaStat, 156
saviLogrankTest *(*safeLogrankTest*)*, 128
saviLogrankTestStat *(*safeLogrankTest*)*, 128
saviRelevanceTStatNEffNu *(*saviTTestStat*)*, 156
saviRelevanceZStat *(*safeZTestStat*)*, 141
saviTTest, *156*
saviTTest *(*safeTTestStat*)*, 131
saviTTestStat, *156*, 156
saviTTestStatNEffNu *(*saviTTestStat*)*, 156
saviTTestStatNEffNuGrow *(*saviTTestStat*)*, 156
saviTTestStatNEffNuMom *(*saviTTestStat*)*, 156
saviTTestStatTDensity *(*saviTTestStat*)*, 156
saviTwoProportionsTest *(*safeTwoProportionsTest*)*, 135
saviZTest *(*safeZTest*)*, 138
saviZTestStat *(*safeZTestStat*)*, 141
selectivelyContinueZOrTTestData, 160
setLowAndHighEsTrueZ, 162
setSafeStatsPlotOptionsAndReturnOldOnes, 163
simulateCoverageDifferenceTwoProportions, 163
simulateIncorrectStoppingTimesFisher, 164
simulateOptionalStoppingScenarioTwoProportions, 165
simulateTwoProportions, *114*, *123*, 166
solveRIPr2x2PropDiff, 168
solveRIPr2x2PropDiff(), *20*
solveUmpLogOdds, 169
solveUmpLogOdds(), *154*
solveWorstCaseTheta2x2PropDiff, 170
solveWorstCaseTheta2x2PropDiff(), *150*
stats::t.test(), *88*, *152*
stats::uniroot(), *27*, *169*
subjectiveBfZStat, 171
Surv, *129*

survdiff, *33*

t.test, *132*, *138*
tryOrFailWithNA, 172
tTestWidthDerivative, 172

zMarg, 173