Package: chartreview (via r-universe)

January 17, 2025

Type Package

Title Adaptive Multi-Wave Sampling for Efficient Chart Validation

Version 1.0

Date 2025-01-14

Author Georg Hahn [aut, cre], Sebastian Schneeweiss [ctb], Shirley Wang [ctb]

Maintainer Georg Hahn <ghahn@bwh.harvard.edu>

Description Functionality to perform adaptive multi-wave sampling for efficient chart validation. Code allows one to define strata, adaptively sample using several types of confidence bounds for the quantity of interest (Lai's confidence bands, Bayesian credible intervals, normal confidence intervals), and sampling strategies (random sampling, stratified random sampling, Neyman's sampling, see Neyman (1934) <doi:10.2307/2342192> and Neyman (1938) <doi:10.1080/01621459.1938.10503378>).

License GPL (>= 2)

RoxygenNote 7.3.2

Encoding UTF-8

Imports Rdpack, anesrake, weights, grDevices, graphics, methods, stats

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-17 09:20:02 UTC

Config/pak/sysreqs cmake make libicu-dev libx11-dev zlib1g-dev

Contents

credibleinterval 2
fullrun 3
lai 4
makeplot 5
normalci 6

stoppingcriterion 7
stratum 7
subsetInterval 8

Index **10**

`credibleinterval` *Bayesian credible interval for binomial quantity*

Description

Bayesian credible interval for binomial quantity

Usage

```
credibleinterval(k, S, alpha)
```

Arguments

`k`	Number of experiments.
`S`	Observed number of successes.
`alpha`	Level.

Value

Bayesian credible interval.

References

.

Examples

```
require(chartreview)
print(credibleinterval(10,5,0.05))
```

`fullrun` *Adaptive sampling algorithm which implements several types of sampling strategies*

Description

Adaptive sampling algorithm which implements several types of sampling strategies

Usage

```
fullrun(
  dat1,
  S,
  dat2,
  mode = 1,
  batchsize = 100,
  raking = TRUE,
  rakingmode = 3,
  rakingthreshold = 0.05,
  sdEstimate = mad,
  minSamples = 10
)
```

Arguments

`dat1`	First dataset on which the strata are computed.
`S`	Matrix defining the strata.
`dat2`	Second dataset on which confidence intervals are computed.
`mode`	Sampling mode (1 for random sampling, 2 for stratified random sampling, 3 for Neyman's sampling).
`batchsize`	Batch size in each wave.
`raking`	Boolean flag to switch on raking.
`rakingmode`	Option for raking (1 for random sampling, 2 for deterministic allocation, 3 for residual resampling).
`rakingthreshold`	Threshold for applying raking to a stratum.
`sdEstimate`	The estimate of the standard deviation as a function handle (usually sd or mad).
`minSamples`	Minimum number of samples used in each iteration.

Value

List with the resampled datasets per wave.

References

.

Examples

```
require(chartreview)
```

`lai` *Lai confidence sequence for binomial quantity*

Description

Lai confidence sequence for binomial quantity

Usage

```
lai(n, x, alpha)
```

Arguments

`n`	Number of experiments
`x`	Observed number of successes.
`alpha`	Error probability.

Value

Binomial confidence interval.

References

Lai, TL (1976). On Confidence Sequences. Ann Statist 4(2):265-280.

Examples

```
require(chartreview)
print(lai(10,5,0.05))
```

makeplot *Generate plots on confidence intervals and prediction*

Description

Generate plots on confidence intervals and prediction

Usage

```
makeplot(
  dataset2,
  dat2,
  optionCI = 1,
  stopCI = NULL,
  alpha = 0.05,
  stoppingoption = 2,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  makePlot = TRUE
)
```

Arguments

dataset2	The output dataset of the function 'fullrun'.
dat2	Second dataset on which confidence intervals are computed, see function 'fullrun'.
optionCI	Parameter to switch between confidence intervals (1 for Lai's confidence bands, 2 for Bayesian credible intervals, 3 for normal confidence intervals).
stopCI	The stopping bounds.
alpha	The error used to compute confidence bands.
stoppingoption	Type of stopping criterion (1 for confidence interval included in stopCI, 2 for upper bound below or lower bound above stopCI, 3 for length restriction on confidence interval).
xlim	Optional parameter to set x-axis in plots.
ylim	Optional parameter to set y-axis in plots.
main	Optional parameter to set title of plots.
makePlot	Parameter to control plot output.

Value

List with confidence intervals (slot CIs), the stopping point (slot stopline), and the reason for stopping (stopreason, see function 'stoppingcriterion').

References

.

Examples

```
require(chartreview)
```

`normalci` *Normal confidence interval for continuous quantity*

Description

Normal confidence interval for continuous quantity

Usage

```
normalci(x, a)
```

Arguments

`x`	Vector of samples.
`a`	Error probability.

Value

Normal confidence interval.

References

.

Examples

```
require(chartreview)
x <- rnorm(10)
print(normalci(x,0.05))
```

stoppingcriterion *Different options for the stopping criterion*

Description

Different options for the stopping criterion

Usage

```
stoppingcriterion(ci, stopCI, stoppingoption = 2)
```

Arguments

ci	Confidence interval as tuple vector.
stopCI	Either a confidence interval for stoppingoption=1 and stoppingoption=2, or a scalar for stoppingoption=3.
stoppingoption	Option to determine if the stopping criterion is satisfied (1 for confidence interval included in stopCI, 2 for upper bound below or lower bound above stopCI, 3 for length restriction on confidence interval).

Value

Boolean answer if stopping criterion reached.

References

.

Examples

```
require(chartreview)
stoppingcriterion(c(0.5,0.6), c(0.7,0.8), stoppingoption=1)
```

stratum *Statification of input data matrix into given strata*

Description

Statification of input data matrix into given strata

Usage

```
stratum(x, S, index)
```

Arguments

x	Input data matrix.
S	Strata by row in matrix S, with 2 columns per variable aka startpoint [included] and endpoint [excluded].
index	Index of the stratum in S.

Value

Vector of indices belong to the given stratum

References

.

Examples

```
require(chartreview)
x <- matrix(runif(10),ncol=1)
strata <- (0:10)/10
S <- cbind(strata[-length(strata)],strata[-1])
print(stratum(x,S,1))
```

---

subsetInterval *Check if some interval is a subset of another interval*

Description

Check if some interval is a subset of another interval

Usage

```
subsetInterval(x, y)
```

Arguments

x	First interval given by tuple.
y	Second interval given by tuple.

Value

Boolean answer if "x subseteq y".

References

.

Examples

```
require(chartreview)
x <- sort(runif(2))
y <- sort(runif(2))
print(subsetInterval(x,y))
```

Index

credibleinterval, 2

fullrun, 3

lai, 4

makeplot, 5

normalci, 6

stoppingcriterion, 7
stratum, 7
subsetInterval, 8