Package ‘vlad’

December 6, 2018

Type Package

Title Variable Life Adjusted Display and Other Risk-Adjusted Quality Control Charts

Version 0.2.0

Depends R (>= 2.10)

Maintainer Philipp Wittenberg <pwitten@hsu-hh.de>

BugReports https://github.com/wittenberg/vlad/issues

Description Contains functions to set up risk-adjusted quality control charts in health care. For the variable life adjusted display (VLAD) proposed by Lovegrove et al. (1997) <doi:10.1016/S0140-6736(97)06507-0> signaling rules derived in Wittenberg et al. (2018) <doi: 10.1002/sim.7647> are implemented. Additionally, for the risk-adjusted cumulative sum chart based on log-likelihood ratio statistic introduced by Steiner et al. (2000) <doi:10.1093/biostatistics/1.4.441> average run length and control limits can be computed.

License GPL (>= 2)

NeedsCompilation yes

SystemRequirements C++11

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.12)

Suggests dplyr, ggplot2, parallel, rmarkdown, spcadjust (>= 1.1), testthat, tidyr

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

Author Philipp Wittenberg [aut, cre] (<https://orcid.org/0000-0001-7151-8243>), Sven Knoth [aut, ths] (<https://orcid.org/0000-0002-9666-5554>)

Repository CRAN

Date/Publication 2018-12-06 08:50:03 UTC

R topics documented:

`vlad-package` *Variable Life Adjusted Display and Other Risk-Adjusted Quality Control Charts*

Description

Contains functions to set up risk-adjusted quality control charts in health care. For the variable life adjusted display (VLAD) proposed by Lovegrove et al. (1997) <doi:10.1016/S0140-6736(97)06507-0> signaling rules derived in Wittenberg et al. (2018) <doi: 10.1002/sim.7647> are implemented. Additionally, for the risk-adjusted cumulative sum chart based on log-likelihood ratio statistic introduced by Steiner et al. (2000) <doi:10.1093/biostatistics/1.4.441> average run length and control limits can be computed.

bcusum_arl_sim — *Compute ARLs of the Bernoulli CUSUM control charts using simulation*

Description

Compute ARLs of the Bernoulli CUSUM control charts using simulation.

Usage

```
bcusum_arl_sim(r, h, df, R0 = 1, RA = 2)
```

Arguments

`r`	Integer Vector. Number of runs.
`h`	Double. Control Chart limit for detecting deterioration/improvement.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0`/`1`) outcome values of each operation.
`R0`	Double. Odds ratio of death under the null hypotheses.
`RA`	Double. Odds ratio of death under the alternative hypotheses.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

bcusum_crit_sim — *Compute alarm threshold of Bernoulli CUSUM control charts using simulation*

Description

Compute alarm threshold of Bernoulli cumulative sum control charts using simulation.

Usage

```
bcusum_crit_sim(L0, df, R0 = 1, RA = 2, m = 100, nc = 1,
  jmax = 4, verbose = FALSE)
```

Arguments

`L0`	Double. Prespecified in-control Average Run Length.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
`R0`	Double. Odds ratio of death under the null hypotheses.
`RA`	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
`m`	Integer. Number of simulation runs.
`nc`	Integer. Number of cores.
`jmax`	Integer. Number of digits for grid search.
`verbose`	Logical. If `TRUE` verbose output is included, if `FALSE` a quiet calculation of `h` is done.

Details

The function `bcusum_crit_sim` determines the control limit for given in-control ARL (`L0`) by applying a multi-stage search procedure which includes secant rule and the parallel version of `bcusum_arl_sim` using `mclapply`.

Value

Returns a single value which is the control limit `h` for a given in-control ARL.

Author(s)

Philipp Wittenberg

`calceo` *Compute Expected minus Observed value*

Description

Compute Expected minus Observed value.

Usage

```
calceo(df, coeff, yemp = TRUE)
```

Arguments

df — Data Frame. First column Parsonnet Score and second column outcome of each operation.

coeff — Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.

yemp — Logical. If TRUE use observed outcome value, if FALSE use estimated binary logistc regression model.

Value

Returns a single value which is the difference between expected risk and observed outcome.

Author(s)

Philipp Wittenberg

References

Lovegrove J, Valencia O, Treasure T, Sherlaw-Johnson C and Gallivan S (1997). Monitoring the results of cardiac surgery by variable life-adjusted display. *The Lancet*, **350**(9085), pp. 1128–1130.

Poloniecki J, Valencia O and Littlejohns P (1998). Cumulative risk adjusted mortality chart for detecting changes in death rate: observational study of heart surgery. *BMJ*, **316**(7146), pp. 1697–1700.

Steiner S (2014). Risk-Adjusted Monitoring of Outcomes in Health Care. In Lawless JF (ed.), *Statistics in Action*, pp. 225–242. Informa UK Limited.

Examples

```
## Not run:
library("vlad")
# see Steiner (2014) p. 234
coeff <- c("(Intercept)"=-3.68, "Parsonnet"=0.077)
# penalty reward for death (E-O scores multiplied with -1 to get O-E scores)
calceo(df=data.frame(as.integer(0), 1), coeff=coeff)*-1
calceo(df=data.frame(as.integer(50), 1), coeff=coeff)*-1
# penalty reward for survival
calceo(df=data.frame(as.integer(0), 0), coeff=coeff)*-1
calceo(df=data.frame(as.integer(50), 0), coeff=coeff)*-1

# Plot a VLAD/CRAM chart
data("cardiacsurgery", package="spcadjust")
cardiacsurgery <- dplyr::mutate(cardiacsurgery, phase=factor(ifelse(date < 2*365, "I", "II")))
S2 <- subset(cardiacsurgery, c(surgeon==2), c("phase", "Parsonnet", "status"))
S2I <- subset(S2, c(phase=="I"))
S2II <- subset(S2, c(phase=="II"))
coeff <- coef(glm(status ~ Parsonnet, data=S2I, family="binomial"))
EO <- sapply(1:nrow(S2), function(i) calceo(df=S2[i, c("Parsonnet", "status")], coeff=coeff))
df1 <- data.frame(cbind(subset(S2, select=c("phase")), "n"=1:nrow(S2), "cEO"=cumsum(EO)))
df2 <- tidyr::gather(df1, "variable", value, c(-n, -phase))
```

```
ggplot2::qplot(data=df2, n, value, colour=phase, geom=c("line", "point"),
               xlab="Patient number", ylab="CUSUM E-O") +
  ggplot2::geom_hline(yintercept=0, linetype="dashed") +
  ggplot2::theme_classic()

## End(Not run)
```

`cusum_arl_h_sim` *Compute alarm threshold of Bernoulli CUSUM control charts using simulation*

Description

Compute alarm threshold of Bernoulli cumulative sum control charts using simulation.

Usage

```
cusum_arl_h_sim(L0, df, R0 = 1, RA = 2, m = 100, nc = 1,
  jmax = 4, verbose = FALSE)
```

Arguments

L0	Double. Prespecified in-control Average Run Length.
df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
R0	Double. Odds ratio of death under the null hypotheses.
RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
m	Integer. Number of simulation runs.
nc	Integer. Number of cores.
jmax	Integer. Number of digits for grid search.
verbose	Logical. If `TRUE` verbose output is included, if `FALSE` a quiet calculation of `h` is done.

Details

The function `cusum_arl_h_sim` determines the control limit for given in-control ARL (`L0`) by applying a multi-stage search procedure which includes secant rule and the parallel version of `cusum_arl_sim` using `mclapply`.

Value

Returns a single value which is the control limit `h` for a given in-control ARL.

Author(s)

Philipp Wittenberg

Examples

```
# This function is deprecated. See bcusum_crit_sim() instead.
```

`cusum_arl_sim` *Compute ARLs of Bernoulli CUSUM control charts using simulation*

Description

Compute ARLs of Bernoulli cumulative sum control charts using simulation.

Usage

```
cusum_arl_sim(r, h, df, R0 = 1, RA = 2)
```

Arguments

r	Integer Vector. Number of runs.
h	Double. Control Chart limit for detecting deterioration/improvement.
df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
R0	Double. Odds ratio of death under the null hypotheses.
RA	Double. Odds ratio of death under the alternative hypotheses.

Author(s)

Philipp Wittenberg

Examples

```
# This function is deprecated. See bcusum_arl_sim() instead.
```

ell *Estimated log-likelihood.*

Description

Estimated log-likelihood.

Usage

```
ell(s, y, delta)
```

Arguments

s	Integer vector. Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk.
y	Double. Binary (`0/1`) outcome values of each operation.
delta	Double. Box-Cox transformation parameter.

Value

Returns a single value which is estimated log-likelihood.

Author(s)

Philipp Wittenberg

Examples

```
## Not run:
## load data
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset data to
## phase I (In-control) and phase II (monitoring)
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

## subset phase I (In-control)
SI <- filter(SALL, phase == "I") %>% select(s, y)

dML <- search_delta(SI$s, SI$y, type = "ML")
ell(SI$s, SI$y, dML)

## End(Not run)
```

eocusum_adoc_sim — *Compute steady-state ARLs of EO-CUSUM control charts using simulation*

Description

Compute steady-state ARLs of EO-CUSUM control charts using simulation.

Usage

```
eocusum_adoc_sim(r, k, h, df, coeff, coeff2, QS = 1, side = "low",
  type = "cond", m = 50)
```

Arguments

`r`	Integer. Number of of simulation runs.
`k`	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
`h`	Double. Decision interval (alarm limit, threshold) of the CUSUM control chart.
`df`	Data Frame. First column Parsonnet Score and second column outcome of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`coeff2`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
`QS`	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.
`side`	Character. Default is `"low"` to calculate ARL for the upper arm of the V-mask. If side = `"up"`, calculate the lower arm of the V-mask.
`type`	Character. Default argument is `"cond"` for computation of conditional steady-state. Other option is the cyclical steady-state `"cycl"`.
`m`	Integer. Simulated in-control observations.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

Examples

```
# This function is deprecated. See eocusum_ad_sim() instead.
```

`eocusum_ad_sim` *Compute steady-state ARLs of EO-CUSUM control charts using simulation*

Description

Compute steady-state ARLs of EO-CUSUM control charts using simulation.

Usage

```
eocusum_ad_sim(r, k, h, df, coeff, coeff2, QS = 1, side = "low",
  type = "cond", m = 50)
```

Arguments

`r`	Integer. Number of of simulation runs.
`k`	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
`h`	Double. Decision interval (alarm limit, threshold) of the CUSUM control chart.
`df`	Data Frame. First column Parsonnet Score and second column outcome of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`coeff2`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
`QS`	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.
`side`	Character. Default is `"low"` to calculate ARL for the upper arm of the V-mask. If side = `"up"`, calculate the lower arm of the V-mask.
`type`	Character. Default argument is `"cond"` for computation of conditional steady-state. Other option is the cyclical steady-state `"cycl"`.
`m`	Integer. Simulated in-control observations.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Taylor HM (1968). The Economic Design of Cumulative Sum Control Charts. *Technometrics*, **10**(3), pp. 479–488.

Crosier R (1986). A new two-sided cumulative quality control scheme. *Technometrics*, **28**(3), pp. 187–194.

Examples

```
## Not run:
data("cardiacsurgery", package = "spcadjust")
library("dplyr")

## preprocess data to 30 day mortality and subset phase I/II
cardiacsurgery <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

s5000 <- sample_n(cardiacsurgery, size = 5000, replace = TRUE)
df1 <- select(cardiacsurgery, s, y)
df2 <- select(s5000, s, y)

## estimate coefficients from logit model
coeff1 <- round(coef(glm(y ~ s, data = df1, family = "binomial")), 3)
coeff2 <- round(coef(glm(y ~ s, data = df2, family = "binomial")), 3)

## Number of simulation runs
m <- 10^3
## Number of cores
nc <- parallel::detectCores()
# steady state
RNGkind("L'Ecuyer-CMRG")
m <- 10^3
tau <- 50
kopt <- optimal_k(QA = 2, df = S2I, coeff = coeff1, yemp = FALSE)
# eocusum_arloc_h_sim(L0 = 370, df = df1, k = kopt, m = m, side = "low", coeff = coeff1,
 coeff2 = coeff2, nc = nc)
res <- sapply(0:(tau-1), function(i){
 RLS <- do.call(c, parallel::mclapply( 1:m, eocusum_ad_sim, k = kopt, QS = 2, h = 2.637854,
  df = df1, m = i, coeff = coeff1, coeff2 = coeff2, side = "low", mc.cores = nc))
  list(data.frame(cbind(ARL = mean(RLS), ARLSE = sd(RLS)/sqrt(m))))
} )
RES <- data.frame(cbind(M = 0:(tau-1), do.call(rbind, res)))
ggplot2::qplot(x = M, y = ARL, data = RES, geom = c("line", "point")) +
ggplot2::theme_classic()
```

```
## End(Not run)
```

---

eocusum_arloc_h_sim *Compute alarm threshold of Out of Control EO-CUSUM control charts using simulation*

Description

Compute alarm threshold (Out of Control ARL) of EO-CUSUM control charts using simulation.

Usage

```
eocusum_arloc_h_sim(L0, k, df, coeff, coeff2, m = 100, QS = 1,
  side = "low", nc = 1, jmax = 4, verbose = FALSE)
```

Arguments

`L0`	Double. Prespecified in-control Average Run Length.
`k`	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`coeff2`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
`m`	Integer. Number of simulation runs.
`QS`	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.
`side`	Character. Default is `"low"` to calculate ARL for the upper arm of the V-mask. If side = `"up"`, calculate the lower arm of the V-mask.
`nc`	Integer. Number of cores.
`jmax`	Integer. Number of digits for grid search.
`verbose`	Logical. If `TRUE` verbose output is included, if `FALSE` a quiet calculation of `h` is done.

Details

The function `eocusum_arloc_h_sim` determines the control limit for given in-control ARL (`L0`) by applying a multi-stage search procedure which includes secant rule and the parallel version of `eocusum_arloc_sim` using `mclapply`.

Value

Returns a single value which is the control limit h for a given ARL.

Author(s)

Philipp Wittenberg

References

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
data("cardiacsurgery", package = "spcadjust")
library("dplyr")

## preprocess data to 30 day mortality and subset phase I/II
cardiacsurgery <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

s5000 <- sample_n(cardiacsurgery, size = 5000, replace = TRUE)
df1 <- select(cardiacsurgery, s, y)
df2 <- select(s5000, s, y)

## estimate coefficients from logit model
coeff1 <- round(coef(glm(y ~ s, data = df1, family = "binomial")), 3)
coeff2 <- round(coef(glm(y ~ s, data = df2, family = "binomial")), 3)

## Number of simulation runs
m <- 10^3
## Number of cores
nc <- parallel::detectCores()

## Lower CUSUM (detecting deterioration)
## k = 0
eocusum_arloc_h_sim(L0 = 370, df = df1, k = 0, m = m, side = "low", coeff = coeff1, coeff2 =
coeff2, nc = nc)
## use function optimal_k() to determine k = kopt
kopt <- optimal_k(QA = 2, df = S2I, coeff = coeff1, yemp = FALSE)
eocusum_arloc_h_sim(L0 = 370, df = df1, k = kopt, m = m, side = "low", coeff = coeff1, coeff2 =
coeff2, nc = nc)

## Upper CUSUM (detecting improvement)
## k = 0
eocusum_arloc_h_sim(L0 = 370, df = df1, k = 0, m = m, side = "up", coeff = coeff1, coeff2 =
coeff2, nc = nc)
## use function optimal_k() to determine k = kopt
kopt <- optimal_k(QA = 1/2, df = S2I, coeff = coeff1, yemp = FALSE)
```

```
eocusum_arloc_h_sim(L0 = 370, df = df1, k = kopt, m = m, side = "up", coeff = coeff1, coeff2 =
 coeff2, nc = nc)

## End(Not run)
```

`eocusum_arloc_sim` *Compute Out of Control ARLs of EO-CUSUM control charts using simulation*

Description

Compute Out of Control ARLs of EO-CUSUM control charts using simulation.

Usage

```
eocusum_arloc_sim(r, k, h, df, coeff, coeff2, QS = 1, side = "low")
```

Arguments

`r`	Integer. Number of of simulation runs.
`k`	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
`h`	Double. Decision interval (alarm limit, threshold) of the CUSUM control chart.
`df`	Data Frame. First column Parsonnet Score and second column outcome of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`coeff2`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
`QS`	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.
`side`	Character. Default is `"low"` to calculate ARL for the upper arm of the V-mask. If side = `"up"`, calculate the lower arm of the V-mask.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
## Datasets
data("cardiacsurgery", package = "spcadjust")
s5000 <- dplyr::sample_n(cardiacsurgery, size = 5000, replace = TRUE)
df1 <- subset(cardiacsurgery, select=c(Parsonnet, status))
df2 <- subset(s5000, select=c(Parsonnet, status))
## estimate coefficients from logit model
coeff1 <- round(coef(glm(status~Parsonnet, data=df1, family="binomial")), 3)
coeff2 <- round(coef(glm(status~Parsonnet, data=df2, family="binomial")), 3)

## Serial simulation
## set seed for reproducibility
RNGkind("L'Ecuyer-CMRG")
m <- 10^3
RLS <- do.call(c, lapply(1:m, eocusum_arloc_sim, h=4.498, k=0, df=df1, side="low", coeff=coeff1,
                         coeff2=coeff2))
data.frame(cbind("ARL"=mean(RLS), "ARLSE"=sd(RLS)/sqrt(m)))
## ARL=366.697; ARLSE=9.457748
## Parallel simulation (FORK)
## set seed for reproducibility
RNGkind("L'Ecuyer-CMRG")
RLS <- simplify2array(parallel::mclapply(1:m, eocusum_arloc_sim, h=4.498, k=0, df=df1, side="low",
                                         coeff=coeff1, coeff2=coeff2,
                                         mc.cores=parallel::detectCores()))
data.frame(cbind("ARL"=mean(RLS), "ARLSE"=sd(RLS)/sqrt(m)))

## Parallel simulation (PSOCK)
## set seed for reproducibility
RNGkind("L'Ecuyer-CMRG")
no_cores <- parallel::detectCores()
cl <- parallel::makeCluster(no_cores)
side <- 1
h_vec <- 4.498
QS_vec <- 1
m <- 10^3
k <- 0
parallel::clusterExport(cl, c("h_vec", "eocusum_arloc_sim", "df1", "coeff1", "coeff2",
                              "QS_vec", "side", "k"))
time <- system.time( {
RLS <- array(NA, dim=c( length(QS_vec), length(h_vec), m))
for (h in h_vec) {
  for (QS in QS_vec) {
    cat(h, " ", QS, "\n")
   RLS[which(QS_vec==QS), which(h==h_vec), ] <- parallel::parSapply(cl, 1:m, eocusum_arloc_sim,
                                                                    side=side, QS=QS, h=h, k=k,
                                                                         df=df1,  coeff=coeff1,
                                                                            coeff2=coeff2,
                                                                             USE.NAMES=FALSE)
    }
  }
} )
```

```
ARL <- apply(RLS, c(1, 2), mean)
ARLSE <- sqrt(apply(RLS, c(1, 2), var)/m)
print(list(ARL, ARLSE, time))
parallel::stopCluster(cl)

## End(Not run)
```

`eocusum_arl_h_sim` *Compute alarm threshold of EO-CUSUM control charts using simulation*

Description

Compute alarm threshold of EO-CUSUM control charts using simulation.

Usage

```
eocusum_arl_h_sim(L0, k, df, coeff, m = 100, yemp = TRUE,
  side = "low", nc = 1, jmax = 4, verbose = FALSE)
```

Arguments

`L0`	Double. Prespecified in-control Average Run Length.
`k`	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model. For more information see details.
`m`	Integer. Number of simulation runs.
`yemp`	Logical. If `TRUE` use observed outcome value, if `FALSE` use estimated binary logistc regression model.
`side`	Character. Default is `"low"` to calculate ARL for the upper arm of the V-mask. If side = `"up"`, calculate the lower arm of the V-mask.
`nc`	Integer. Number of cores.
`jmax`	Integer. Number of digits for grid search.
`verbose`	Logical. If `TRUE` verbose output is included, if `FALSE` a quiet calculation of `h` is done.

Details

The function `eocusum_arl_h_sim` determines the control limit for given in-control ARL (`L0`) by applying a multi-stage search procedure which includes secant rule and the parallel version of `eocusum_arl_sim` using `mclapply`.

Value

Returns a single value which is the control limit h for a given ARL.

Author(s)

Philipp Wittenberg

Examples

```
# This function is deprecated. See eocusum_crit_sim() instead.
```

`eocusum_arl_sim` *Compute ARLs of EO-CUSUM control charts using simulation*

Description

Compute ARLs of EO-CUSUM control charts using simulation.

Usage

```
eocusum_arl_sim(r, k, h, df, coeff, yemp = TRUE, side = "low")
```

Arguments

`r`	Integer. Number of of simulation runs.
`k`	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
`h`	Double. Decision interval (alarm limit, threshold) of the CUSUM control chart.
`df`	Data Frame. First column Parsonnet Score and second column outcome of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`yemp`	Logical. If `TRUE` use observed outcome value, if `FALSE` use estimated binary logistc regression model.
`side`	Character. Default is `"low"` to calculate ARL for the upper arm of the V-mask. If side = `"up"`, calculate the lower arm of the V-mask.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library("dplyr")
library("tidyr")
library(ggplot2)

## Datasets
data("cardiacsurgery", package = "spcadjust")
cardiacsurgery <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0))
s5000 <- sample_n(cardiacsurgery, size = 5000, replace = TRUE)
df1 <- select(cardiacsurgery, s, y)
df2 <- select(s5000, s, y)

## estimate coefficients from logit model
coeff1 <- round(coef(glm(y ~ s, data = df1, family = "binomial")), 3)
coeff2 <- round(coef(glm(y ~ s, data = df2, family = "binomial")), 3)

## set up
RNGkind("L'Ecuyer-CMRG")
m <- 10^3
kopt <- optimal_k(QA = 2, df = S2I, coeff = coeff1, yemp = FALSE)
h <- eocusum_arloc_h_sim(L0 = 370, df = df1, k = kopt, m = m, side = "low", coeff = coeff1,
                         coeff2 = coeff2, nc = 4)

## Serial simulation
RLS <- do.call(c, lapply(1:m, eocusum_arloc_sim, h = h, k = kopt, df = df1, side = "low",
                         coeff = coeff1, coeff2 = coeff2))
data.frame(cbind(ARL = mean(RLS), ARLSE = sd(RLS)/sqrt(m)))

## Parallel simulation (FORK)
RLS <- simplify2array(parallel::mclapply(1:m, eocusum_arloc_sim, h = h, k = kopt, df = df1,
                                         side = "low", coeff = coeff1, coeff2 = coeff2,
                                         mc.cores = parallel::detectCores()))
data.frame(cbind(ARL = mean(RLS), ARLSE = sd(RLS)/sqrt(m)))

## Parallel simulation (PSOCK)
no_cores <- parallel::detectCores()
cl <- parallel::makeCluster(no_cores)
side <- "low"
h_vec <- h
QS_vec <- 1
```

```
k <- kopt
parallel::clusterExport(cl, c("h_vec", "eocusum_arloc_sim", "df1", "coeff1", "coeff2",
                              "QS_vec", "side", "k"))
time <- system.time( {
  RLS <- array(NA, dim = c( length(QS_vec), length(h_vec), m))
  for (h in h_vec) {
    for (QS in QS_vec) {
      cat(h, " ", QS, "\n")
     RLS[which(QS_vec==QS), which(h==h_vec), ] <- parallel::parSapply(cl, 1:m, eocusum_arloc_sim,
                                                              side = side, QS = QS, h = h,
                                                                        k = k, df = df1,
                                                                        coeff = coeff1,
                                                                        coeff2 = coeff2,
                                                                       USE.NAMES = FALSE)
    }
  }
} )
ARL <- apply(RLS, c(1, 2), mean)
ARLSE <- sqrt(apply(RLS, c(1, 2), var)/m)
print(list(ARL, ARLSE, time))
parallel::stopCluster(cl)

## End(Not run)
```

---

eocusum_crit_sim *Compute alarm threshold of EO-CUSUM control charts using simulation*

Description

Compute alarm threshold of EO-CUSUM control charts using simulation.

Usage

```
eocusum_crit_sim(L0, k, df, coeff, m = 100, yemp = TRUE,
  side = "low", nc = 1, jmax = 4, verbose = FALSE)
```

Arguments

L0	Double. Prespecified in-control Average Run Length.
k	Double. Reference value of the CUSUM control chart. Either `0` or a positive value. Can be determined with function `optimal_k`.
df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
coeff	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model. For more information see details.
m	Integer. Number of simulation runs.

yemp	Logical. If TRUE use observed outcome value, if FALSE use estimated binary logistc regression model.
side	Character. Default is "low" to calculate ARL for the upper arm of the V-mask. If side = "up", calculate the lower arm of the V-mask.
nc	Integer. Number of cores.
jmax	Integer. Number of digits for grid search.
verbose	Logical. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

The function eocusum_crit_sim determines the control limit for given in-control ARL (L0) by applying a multi-stage search procedure which includes secant rule and the parallel version of eocusum_arl_sim using mclapply.

Value

Returns a single value which is the control limit h for a given ARL.

Author(s)

Philipp Wittenberg

References

Barnard GA (1959). Control charts and stochastic processes. *J R Stat Soc Series B Stat Methodol*, **21**(2), pp. 239–271.

Kemp KW (1961). The Average Run Length of the Cumulative Sum Chart when a V-mask is used. *J R Stat Soc Series B Stat Methodol*, **23**(1),pp. 149–153.

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
data("cardiacsurgery", package = "spcadjust")
library("dplyr")

## preprocess data to 30 day mortality and subset phase I (In-control) of surgeons 2
S2I <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II"))) %>%
  filter(phase == "I", surgeon == 2) %>% select(s, y)

## estimate coefficients from logit model
coeff1 <- coef(glm(y ~ s, data = S2I, family = "binomial"))
## Number of simulation runs
m <- 10^3
```

```
set.seed(1234)
## Number of cores
nc <- parallel::detectCores()

## determine k for detecting deterioration
kopt <- optimal_k(QA = 2, df = S2I, coeff = coeff, yemp = FALSE)

## compute threshold for prespecified in-control ARL
h <- eocusum_crit_sim(L0 = 370, df = S2I, k = kopt, m = m, coeff = coeff1, side = "low",
nc = nc)

## parameters to set up a tabular CUSUM or V-Mask
d <- h/kopt
theta <- atan(kopt)*180/pi
cbind(kopt, h, theta, d)

## End(Not run)
```

`eocusum_scores` *Compute CUSUM scores based on E-O*

Description

Compute CUSUM scores based on E-O.

Usage

```
eocusum_scores(z, k1, k2, reset = FALSE, h1 = NULL, h2 = NULL)
```

Arguments

`z`	NumericVector. `E-O` values.
`k1`	Double. Reference value `k` for detecting improvement can be determined from function `optimal_k`.
`k2`	Double. Reference value `k` for detecting deteroration can be determined from function `optimal_k`.
`reset`	Logical. If `FALSE` CUSUM statistic is not reset. If `TRUE` CUSUM statistic is reset to `0` after a signal is issued.
`h1`	Double. Upper control limit of the CUSUM chart.
`h2`	Double. Lower control limit of the CUSUM chart.

Value

Returns a list with two components for the CUSUM scores.

Author(s)

Philipp Wittenberg

References

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library("dplyr")
library("tidyr")
library(ggplot2)
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset phase I (In-control) of surgeons 2
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

## subset phase I (In-control)
SI <- filter(SALL, phase == "I") %>% select(s, y)

## estimate coefficients from logit model
coeff1 <- coef(glm(y ~ s, data = SI, family = "binomial"))

## determine k for detecting deterioration
kopt <- optimal_k(QA = 2, df = SI, coeff = coeff, yemp = FALSE)

## subset phase II of surgeons 2
S2II <- filter(SALL, phase == "II", surgeon == 2) %>% select(s, y)
n <- nrow(S2II)

## CUSUM statistic without reset
z <- sapply(1:n, function(i) calceo(df = S2II[i, c("s", "y")], coeff = coeff1))
cv <- eocusum_scores(z = z, k = kopt)
s1 <- cv$s1; s1l <- cv$s1l
dm1 <- data.frame(cbind("n" = 1:length(s1), "Cup" = s1, "Clow" = s1l, "h1" = 2, "h2" = -2))

## CUSUM statistic reset after signal
cv <- eocusum_scores(z = z, k = kopt, reset = TRUE, h1 = 2, h2 = 2)
s1 <- cv$s1; s1l <- cv$s1l
dm2 <- data.frame(cbind("n" = 1:length(s1), "Cup" = s1, "Clow" = s1l, "h1" = 2, "h2" = -2))

dm3 <- bind_rows(dm1, dm2, .id = "type")
dm3$type <- recode_factor(dm3$type, `1`="No resetting", `2`="Resetting")
dm3 %>%
  gather("CUSUM", value, c(-n, - type)) %>%
  ggplot(aes(x = n, y = value, colour = CUSUM, group = CUSUM)) +
  geom_hline(yintercept = 0, colour = "darkgreen", linetype = "dashed") +
  geom_line(size = 0.5) +
  facet_wrap( ~ type, ncol = 1, scales = "free") +
  labs(x = "Patient number n", y = "CUSUM values") + theme_classic() +
  scale_y_continuous(sec.axis = dup_axis(name = NULL, labels = NULL)) +
```

```
  scale_x_continuous(sec.axis = dup_axis(name = NULL, labels = NULL)) +
  guides(colour = "none") +
  scale_color_manual(values = c("blue", "orange", "red", "red"))

## End(Not run)
```

---

gettherisk *Compute Risk of death*

Description

Compute Risk of death.

Usage

```
gettherisk(parsonnetscore, coeff)
```

Arguments

parsonnetscore	Integer. Parsonnet Score.
coeff	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.

Value

Returns a single value which is the expected risk based on a risk model.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Steiner S (2014). Risk-Adjusted Monitoring of Outcomes in Health Care. In Lawless JF (ed.), *Statistics in Action*, pp. 225–242. Informa UK Limited.

Parsonnet V, Dean D, Bernstein AD (1989). A method of uniform stratification of risk for evaluating the results of surgery in acquired adult heart disease. *Circulation*, **79**(6):I3–12.

Rigdon SE and Fricker RD (2015). Health Surveillance. In Chen DG and Wilson J (eds) *Innovative Statistical Methods for Public Health Data*, pp. 203–249. Springer, Cham.

Examples

```
## Not run:
library(vlad)
## see Steiner et al. 2000 p. 445 or Steiner (2014) p. 234
coeff <- c("(Intercept)" = -3.68, "Parsonnet" = 0.077)
## low risk patient (Parsonnet score=0) has a risk of death 2.5%
gettherisk(0L, coeff = coeff)
## high risk patient (Parsonnet score=71) has a risk of death 86%
gettherisk(71L, coeff = coeff)
## high risk patient (Parsonnet score=50) has a risk of death 54%
gettherisk(50L, coeff = coeff)

## see Rigdon and Fricker (2015) p. 221 and p. 225
coeff <- c("(Intercept)" = -3.67, "Parsonnet" = 0.077)
## patients probability of death 0.09912 for Parsonnet score 19
round(gettherisk(19L, coeff), 5)
## patients probability of death 0.02484 for Parsonnet score 0
round(gettherisk(0L, coeff), 5)

## preprocess data to 30 day mortality and subset phase I (In-control)
library("dplyr")
data("cardiacsurgery", package = "spcadjust")
SI <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
        phase = factor(ifelse(date < 2*365, "I", "II"))) %>%
  filter(phase == "I") %>% select(s, y)

## Get mortality and probability of death of a phase I dataset
GLM1 <- glm(y ~ s, data = SI, family = "binomial")
coeff1 <- coef(GLM1)
mprob <- as.numeric(table(SI$s) / length(SI$s))

## Use estimated model coefficients and parsonnet scores in function gettherisk()
## or predicted values from a GLM
usi <- sort(unique(SI$s))
mort <- sapply(usi, gettherisk, coeff = coeff1)
mort1 <- predict(GLM1, newdata = data.frame(s = usi), type = "response")
all.equal(as.numeric(mort), as.numeric(mort1))
df1 <- data.frame(s = usi, mprob, mort)

## Plot mortality and estimated probability to die of phase I data
library(ggplot2)
qplot(data = df1, s, mprob, geom = c("line", "point")) + theme_classic()
xx <- tapply(SI$y, SI$s, sum)
nn <- tapply(SI$y, SI$s, length)
ll <- binom::binom.confint(xx, nn, conf.level = 0.99, methods = "exact")$lower
uu <- binom::binom.confint(xx, nn, conf.level = 0.99, methods = "exact")$upper
ybar <- tapply(SI$y, SI$s, mean)
ggplot(data = df1, aes(s, mort)) +
 geom_point(data = data.frame(s = usi, ybar), aes(s, ybar), inherit.aes = FALSE) +
 geom_errorbar(aes(ymax = uu, ymin = ll), width = 0.9, position = "dodge", alpha = 0.3) +
 geom_line(colour = "red") + labs(x = "Parsonnet score", y = "Estimated Probability to die") +
```

```
 theme_classic()

## End(Not run)
```

---

llr_score *Compute the log-likelihood ratio score*

Description

Compute the log-likelihood ratio score.

Usage

```
llr_score(df, coeff, R0 = 1, RA = 2, yemp = TRUE)
```

Arguments

df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
coeff	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
R0	Double. Odds ratio of death under the null hypotheses.
RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
yemp	Logical. If `TRUE` use observed outcome value, if `FALSE` use estimated binary logistc regression model.

Value

Returns a single value which is the log-likelihood ratio score.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Steiner S (2014). Risk-Adjusted Monitoring of Outcomes in Health Care. In Lawless JF (ed.), *Statistics in Action*, pp. 225–242. Informa UK Limited.

Rigdon SE and Fricker RD (2015). Health Surveillance. In Chen DG and Wilson J (eds) *Innovative Statistical Methods for Public Health Data*, pp. 203–249. Springer, Cham.

Examples

```
## Not run:
library(vlad)
## see Steiner et al. (2000) p. 446 or Steiner (2014) p. 234
coeff <- c("(Intercept)" = -3.68, "Parsonnet" = 0.077)
## Log-likelihood ratio scores for detecting an increase in the failure rate:
## low risk patients with a Parsonnet score of zero

llr_score(df = data.frame(as.integer(0), 0), coeff = coeff, RA = 2)
llr_score(df = data.frame(as.integer(0), 1), coeff = coeff, RA = 2)

## higher risk patients with a Parsonnet score of 50
llr_score(df = data.frame(as.integer(50), 0), coeff = coeff, RA = 2)
llr_score(df = data.frame(as.integer(50), 1), coeff = coeff, RA = 2)

## see Steiner (2014) p. 234
## Log-likelihood ratio scores for detecting an decrease in the failure rate:
## low risk patients with a Parsonnet score of zero
llr_score(df = data.frame(as.integer(0), 0), coeff = coeff, RA = 1/2)
llr_score(df = data.frame(as.integer(0), 1), coeff = coeff, RA = 1/2)

## higher risk patients with a Parsonnet score of 50
llr_score(df = data.frame(as.integer(50), 0), coeff = coeff, RA = 1/2)
llr_score(df = data.frame(as.integer(50), 1), coeff = coeff, RA = 1/2)

## see Rigdon and Fricker p. 225 and 226
## detecting an increase in the failure rate:
coeff <- c("(Intercept)" = -3.67, "Parsonnet" = 0.077)
df <- data.frame(Parsonnet = c(19L, 19L, 0L, 0L), status = c(0, 1, 0, 1))
lapply(seq_along(df$Parsonnet), function(i) round(llr_score(df = df[i, ], coeff = coeff,
 RA = 2), 4))

## detecting an decrease in the failure rate:
round(llr_score(df = data.frame(19L, 0), coeff = coeff, RA = 1/2), 5)

## End(Not run)
```

---

optimal_k *Compute optimal k*

Description

Compute optimal k.

Usage

```
optimal_k(QA, df, coeff, yemp = TRUE)
```

Arguments

QA
: Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.

df
: Data Frame. First column Parsonnet Score and second column outcome of each operation.

coeff
: Numeric Vector. Estimated coefficients α and β from the binary logistic regression model. For more information see details.

yemp
: Logical. If `TRUE` use observed outcome value, if `FALSE` use estimated binary logistc regression model.

Details

Formula deterioration:

$$kdet = \frac{QA - 1 - log(QA)}{log(QA)} \bar{p}, QA > 1$$

Formula improvement:

$$kimp = \frac{1 - QA + log(QA)}{log(QA)} \bar{p}, QA < 1$$

Value

Returns a single value which is the approximate optimal `k` for a set of given Parsonnet scores.

Author(s)

Philipp Wittenberg

References

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library("dplyr")
library(vlad)
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset phase I (In-control) of surgeons 2
S2I <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
        phase = factor(ifelse(date < 2*365, "I", "II"))) %>%
  filter(phase == "I", surgeon == 2) %>% select(s, y)

coeff <- coef(glm(y ~ s, data = S2I, family = "binomial"))

## (Deterioration)
```

```
optimal_k(QA = 2, df = S2I, coeff = coeff, yemp = FALSE)

## manually find optimal k for detecting deterioration
QA <- 2
pbar <- mean(sapply(S2I[, 1], gettherisk, coef = coeff))
kopt <- pbar * ( QA - 1 - log(QA) ) / log(QA)

all.equal(kopt, optimal_k(QA = 2, df = S2I, coeff = coeff, yemp = FALSE))

## (Improvement)
optimal_k(QA = 1/2, df = S2I, coeff = coeff, yemp = FALSE)

## manually find optimal k for detecting improvement
QA <- 1/2
pbar <- mean(sapply(S2I[, 1], gettherisk, coef = coeff))
kopt <- pbar * ( 1 - QA + log(QA) ) / log(QA)

all.equal(kopt, optimal_k(QA = 1/2, df = S2I, coeff = coeff, yemp = FALSE))

## End(Not run)
```

QQ *Pearson measure*

Description

Pearson measure.

Usage

```
QQ(s, y, delta)
```

Arguments

s	Integer vector. Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk.
y	Numeric Vector. Binary (`0/1`) outcome values of each operation.
delta	Double. Box-Cox transformation parameter.

Value

Returns a single value.

Author(s)

Philipp Wittenberg

Examples

```
## Not run:
## load data
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset data to
## phase I (In-control) and phase II (monitoring)
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

## subset phase I (In-control)
SI <- filter(SALL, phase == "I") %>% select(s, y)

dQQ <- search_delta(SI$s, SI$y, type = "Pearson")
QQ(SI$s, SI$y, dQQ)

## End(Not run)
```

`racusum_adoc_sim` *Compute steady-state ARLs of RA-CUSUM control charts using simulation*

Description

Compute steady-state ARLs of risk-adjusted cumulative sum control charts using simulation.

Usage

```
racusum_adoc_sim(r, coeff, coeff2, h, df, R0 = 1, RA = 2, RQ = 1,
  m = 50, type = "cond")
```

Arguments

`r`	Integer Vector. Number of runs.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`coeff2`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
`h`	Double. Control Chart limit for detecting deterioration/improvement.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
`R0`	Double. Odds ratio of death under the null hypotheses.

RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
RQ	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.
m	Integer. Simulated in-control observations.
type	Character. Default argument is `"cond"` for computation of conditional steady-state. Other option is the cyclical steady-state `"cycl"`.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452. doi: doi: 10.1093/biostatistics/1.4.441.

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Taylor HM (1968). The Economic Design of Cumulative Sum Control Charts. *Technometrics*, **10**(3), pp. 479–488.

Crosier R (1986). A new two-sided cumulative quality control scheme. *Technometrics*, **28**(3), pp. 187–194.

Examples

```
## Not run:
library(vlad)
data("cardiacsurgery", package="spcadjust")
# build data set
df1 <- subset(cardiacsurgery, select=c(Parsonnet, status))

# estimate coefficients from logit model
coeff1 <- round(coef(glm(status ~ Parsonnet, data=df1, family="binomial")), 3)

# simulation of conditional steady state
m <- 10^3
tau <- 50
res <- sapply(0:(tau-1), function(i){
 RLS <- do.call(c, parallel::mclapply( 1:m, racusum_adoc_sim, RQ=2, h=2.0353, df=df1, m=i,
                                        coeff=coeff1, coeff2=coeff1,
                                        mc.cores=parallel::detectCores()) )
```

```
 list(data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m))))
} )

# plot
RES <- data.frame(cbind(M=0:(tau-1), do.call(rbind, res)))
ggplot2::qplot(x=M, y=ARL, data=RES, geom=c("line", "point")) +
ggplot2::theme_classic()

## End(Not run)
```

`racusum_arloc_h_sim` *Compute alarm threshold (Out of Control ARL) of RA-CUSUM control charts using simulation*

Description

Compute alarm threshold (Out of Control ARL) of risk-adjusted cumulative sum control charts using simulation.

Usage

```
racusum_arloc_h_sim(L0, df, coeff, coeff2, R0 = 1, RA = 2, RQ = 1,
  m = 100, nc = 1, jmax = 4, verbose = FALSE)
```

Arguments

`L0`	Double. Prespecified in-control Average Run Length.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`coeff2`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
`R0`	Double. Odds ratio of death under the null hypotheses.
`RA`	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
`RQ`	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.
`m`	Integer. Number of simulation runs.
`nc`	Integer. Number of cores.
`jmax`	Integer. Number of digits for grid search.
`verbose`	Logical. If `TRUE` verbose output is included, if `FALSE` a quiet calculation of `h` is done.

Details

The function racusum_arloc_h_sim determines the control limit h for given in-control ARL (L0) by applying a multi-stage search procedure which includes secant rule and the parallel version of racusum_arloc_sim using mclapply.

Value

Returns a single value which is the control limit h for a given in-control ARL.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library(vlad)
# Set seed for reproducibility
RNGkind("L'Ecuyer-CMRG")
set.seed(1234)
parallel::mc.reset.stream()
# Datasets
data("cardiacsurgery", package = "spcadjust")
s5000 <- dplyr::sample_n(cardiacsurgery, size = 5000, replace = TRUE)
df1 <- subset(cardiacsurgery, select = c(Parsonnet, status))
df2 <- subset(s5000, select = c(Parsonnet, status))

# Estimate coefficients from logit model
coeff1 <- round(coef(glm(status ~ Parsonnet, data = df1, family = "binomial")), 3)
coeff2 <- round(coef(glm(status ~ Parsonnet, data = df2, family = "binomial")), 3)

# Number of simulation runs
m <- 10^3

# Deterioration:
# 1. Determine critical value for given ARL
racusum_arloc_h_sim(L0 = 370, df = df1, coeff = coeff1, coeff2 = coeff2, m = m, RA = 2, nc = 6)
# h = 2.030933

# 2. Determine ARL and Standard Error
RLS <- do.call(c, parallel::mclapply(1:m, racusum_arloc_sim, h = 2.035, df = df1, RA = 2,
                                     coeff = coeff1, coeff2 = coeff2, mc.cores = 6))
data.frame(cbind("ARL"=mean(RLS), "ARLSE"=sd(RLS)/sqrt(m)))
```

```
# ARL = 371.125; ARLSE = 11.36053

# Improvement:
# 1. Determine critical value for given ARL
racusum_arloc_h_sim(L0 = 370, df = df1, coeff = coeff1, coeff2 = coeff2, m = m, RA = 1/2,
                    nc = 6)
# h = 1.710999
#
# 2. Determine ARL and Standard Error
RLS <- do.call(c, parallel::mclapply(1:m, racusum_arloc_sim, h = 1.760, df = df1, RA = 1/2,
                                     coeff = coeff1, coeff2 = coeff2, mc.cores = 6))
data.frame(cbind("ARL" = mean(RLS), "ARLSE" = sd(RLS)/sqrt(m)))
# ARL = 399.613; ARLSE = 10.7601

## End(Not run)
```

---

racusum_arloc_sim *Compute Out of Control ARLs of RA-CUSUM control charts using simulation*

Description

Compute Out of Control ARLs of risk-adjusted cumulative sum control charts using simulation.

Usage

```
racusum_arloc_sim(r, coeff, coeff2, h, df, R0 = 1, RA = 2, RQ = 1)
```

Arguments

r	Integer Vector. Number of runs.
coeff	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
coeff2	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
h	Double. Control Chart limit for detecting deterioration/improvement.
df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
R0	Double. Odds ratio of death under the null hypotheses.
RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
RQ	Double. Defines the performance of a surgeon with the odds ratio ratio of death `Q`.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library("vlad"); library("ggplot2")
## Set seed for reproducibility
RNGkind("L'Ecuyer-CMRG")
## Datasets
data("cardiacsurgery", package = "spcadjust")
s5000 <- dplyr::sample_n(cardiacsurgery, size = 5000, replace = TRUE)
df1 <- subset(cardiacsurgery, select = c(Parsonnet, status))
df2 <- subset(s5000, select = c(Parsonnet, status))

## Estimate coefficients from logit model
coeff1 <- round(coef(glm(status ~ Parsonnet, data = df1, family = "binomial")), 3)
coeff2 <- round(coef(glm(status ~ Parsonnet, data = df2, family = "binomial")), 3)

## Number of simulation runs
m <- 10^3

## Deterioration RA=2:
## 1. Determine critical value for given ARL
h0 <- racusum_arloc_h_sim(L0 = 370, df = df1, coeff = coeff1, coeff2 = coeff2, m = m, RA = 2,
nc = 6)
## 2. Compute Out of Control ARL
RQ <- seq(1, 4, 0.1)
rl <- array(NA, dim = c(m, length(RQ)))
RLS <- sapply(RQ, function(i) {
  cat("RQ: ", i, "\n" )
 rl[, i] <- do.call(c, parallel::mclapply(1:m, racusum_arloc_sim, h = h0, df = df1, RA = 2,
  RQ = i, coeff = coeff1, coeff2 = coeff2, mc.cores = 6))
})
df3 <- data.frame(cbind(RQ, "ARL" = apply(RLS, 2, mean), "ARLSE" = apply(RLS, 2, mean)/sqrt(m)))
ggplot(df3, aes(RQ, ARL)) + geom_line() + theme_classic()

## Improvement RA=1/2:
## 1. Determine critical value for given ARL
```

```
h0 <- racusum_arloc_h_sim(L0 = 370, df = df1, coeff = coeff1, coeff2 = coeff2, m = m, RA = 1/2,
                          nc = 6)
## 2. Compute Out of Control ARL
RQ <- seq(1/4, 1, 1/40)
rl <- array(NA, dim = c(m, length(RQ)))
RLS <- sapply(RQ, function(i) {
  cat("RQ: ", i, "\n" )
 rl[, i] <- do.call(c, parallel::mclapply(1:m, racusum_arloc_sim, h = h0, df = df1, RA = 1/2,
                                          RQ = i, coeff = coeff1, coeff2 = coeff2,
                                          mc.cores = 6))
})
df4 <- data.frame(cbind(RQ, "ARL" = apply(RLS, 2, mean), "ARLSE" = apply(RLS, 2, mean)/sqrt(m)))
ggplot(df4, aes(RQ, ARL)) + geom_line() + theme_classic()

## End(Not run)
```

`racusum_arl_h_sim` *Compute alarm threshold of RA-CUSUM control charts using simulation*

Description

Compute alarm threshold of risk-adjusted cumulative sum control charts using simulation.

Usage

```
racusum_arl_h_sim(L0, df, coeff, R0 = 1, RA = 2, m = 100,
  yemp = TRUE, nc = 1, jmax = 4, verbose = FALSE)
```

Arguments

`L0`	Double. Prespecified in-control Average Run Length.
`df`	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
`coeff`	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
`R0`	Double. Odds ratio of death under the null hypotheses.
`RA`	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
`m`	Integer. Number of simulation runs.
`yemp`	Logical. If `TRUE`, use emirical outcome values, else use model.
`nc`	Integer. Number of cores used for parallel processing.
`jmax`	Integer. Number of digits for grid search.
`verbose`	Logical. If `TRUE` verbose output is included, if `FALSE` a quiet calculation of `h` is done.

Details

The function racusum_arl_h_sim determines the control limit h for given in-control ARL (L0) by applying a multi-stage search procedure which includes secant rule and the parallel version of racusum_arl_sim using mclapply.

Value

Returns a single value which is the control limit h for a given in-control ARL.

Author(s)

Philipp Wittenberg

Examples

```
# This function is deprecated. See racusum_crit_sim() instead.
```

---

racusum_arl_mc *Compute ARLs of RA-CUSUM control charts using Markov chain approximation*

Description

Computes the Average Run Length of a risk-adjusted cumulative sum control chart using Markov chain approximation.

Usage

```
racusum_arl_mc(pmix, RA, RQ, h, scaling = 600, rounding = "p",
  method = "Toep")
```

Arguments

pmix	Numeric Matrix. A three column matrix. First column is the risk score distribution. Second column are the predicted probabilities from the risk model. Third column can be either the predicted probabilities from the risk model or average outcome per risk score, see examples.
RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio RA = 2. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death RA = 1/2. Odds ratio of death under the null hypotheses is 1.

`RQ`	Double. Defines the true performance of a surgeon with the odds ratio ratio of death `RQ`. Use `RQ = 1` to compute the in-control ARL and other values to compute the out-of-control ARL.
`h`	Double. `h` is the control limit (>`0`).
`scaling`	Double. The `scaling` parameter controls the quality of the approximation, larger values achieve higher accuracy but increase the computation burden (larger transition probability matrix).
`rounding`	Character. If `rounding = "p"` a paired rounding implementation similar to *Webster and Pettitt (2007)* is used, if `rounding = "s"` a simple rounding method of *Steiner et al. (2000)* is used.
`method`	Character. If `method = "Toep"` a combination of Sequential Probability Ratio Test and Toeplitz matrix structure is used to calculate the ARL. `"ToepInv"` computes the inverted matrix using Toeplitz matrix structure. `"BE"` solves a linear equation system using the classical approach of *Brook and Evans (1972)* to calculate the ARL.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Brook D and Evans DA (1972) An approach to the probability distribution of CUSUM run length. *Biometrika*, **59**(3), pp. 539–549

Webster RA and Pettitt AN (2007) Stability of approximations of average run length of risk-adjusted CUSUM schemes using the Markov approach: comparing two methods of calculating transition probabilities. *Communications in Statistics - Simulation and Computation* **36**(3), pp. 471–482

Examples

```
## Not run:
library(vlad)
library(dplyr)
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset phase I (In-control) of surgeons 2
SALLI <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II"))) %>%
  filter(phase == "I") %>% select(s, y)

## estimate risk model, get relative frequences and probabilities
mod1 <- glm(y ~ s, data = SALLI, family = "binomial")
```

```
fi  <- as.numeric(table(SALLI$s) / length(SALLI$s))
usi <- sort(unique(SALLI$s))
pi1 <- predict(mod1, newdata = data.frame(s = usi), type = "response")
pi2 <- tapply(SALLI$y, SALLI$s, mean)

## set up patient mix (risk model)
pmix1  <- data.frame(fi, pi1, pi1)

## Average Run Length for detecting deterioration RA = 2:
racusum_arl_mc(pmix = pmix1, RA = 2, RQ = 1, h = 4.5)

## Average Run Length for detecting improvement RA = 1/2:
racusum_arl_mc(pmix = pmix1, RA = 1/2, RQ = 1, h = 4)

## set up patient mix (model free)
pmix2  <- data.frame(fi, pi1, pi2)

## Average Run Length for detecting deterioration RA = 2:
racusum_arl_mc(pmix = pmix2, RA = 2, RQ = 1, h = 4.5)

## Average Run Length for detecting improvement RA = 1/2:
racusum_arl_mc(pmix = pmix2, RA = 1/2, RQ = 1, h = 4)

## compare results with R-code function 'findarl()' from Steiner et al. (2000)
source("https://bit.ly/2KC0SYD")
all.equal(findarl(pmix = pmix1, R1 = 2, R = 1, CL = 4.5, scaling = 600),
       racusum_arl_mc(pmix = pmix1, RA = 2, RQ = 1, h = 4.5, scaling = 600, rounding = "s"))

## End(Not run)
```

---

racusum_arl_sim *Compute ARLs of RA-CUSUM control charts using simulation*

Description

Computes the Average Run Length of a risk-adjusted cumulative sum control chart using simulation.

Usage

```
racusum_arl_sim(r, coeff, h, df, R0 = 1, RA = 2, yemp = TRUE)
```

Arguments

r	Integer Vector. Number of runs.
coeff	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
h	Double. Control Chart limit for detecting deterioration/improvement.

df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
R0	Double. Odds ratio of death under the null hypotheses.
RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.
yemp	Logical. If `TRUE` use observed outcome value, if `FALSE` use estimated binary logistc regression model.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library(vlad)
set.seed(1234)
data("cardiacsurgery", package="spcadjust")
df1 <- subset(cardiacsurgery, select=c(Parsonnet, status))
coeff1 <- round(coef(glm(status ~ Parsonnet, data=df1, family="binomial")), 3)

## Parallel Simulation 1: y = random (10^4 runs, RA=2)
m <- 10^4; h_vec <- 2.7; yemp <- FALSE
no_cores <- parallel::detectCores()
cl <- parallel::makeCluster(no_cores)
parallel::clusterExport(cl, c("h_vec", "racusum_arl_sim", "coeff1", "df1", "yemp"))
time <- system.time( {
  ARL <- array(NA, dim=c( length(h_vec), m))
  for (h in h_vec) {
   ARL[which(h_vec==h), ] <- parallel::parSapply(cl, 1:m, racusum_arl_sim, h=h, coeff=coeff1,
                                                  df=df1, yemp=yemp, USE.NAMES=FALSE) }
} )
simMean <- apply(ARL, c(1), mean)
simSE <- sqrt(apply(ARL, c(1), var)/m)
print(list(simMean, simSE, time))
```

```
parallel::stopCluster(cl)
df.sim1 <- data.frame("RA"=2, "h"=h, "ARL"=simMean, "ARLSE"=simSE, "nsim"=m)

## Parallel Simulation 2: y = empirical (10^4 runs, RA=2)
m <- 10^4; h_vec <- 2.7
no_cores <- parallel::detectCores()
cl <- parallel::makeCluster(no_cores)
parallel::clusterExport(cl, c("h_vec", "racusum_arl_sim", "coeff1", "df1"))
time <- system.time( {
  ARL <- array(NA, dim=c( length(h_vec), m))
  for (h in h_vec) {
   ARL[which(h_vec==h), ] <- parallel::parSapply(cl, 1:m, racusum_arl_sim, h=h, coeff=coeff1,
                                                 df=df1, USE.NAMES=FALSE) }
} )
simMean <- apply(ARL, c(1), mean)
simSE <- sqrt(apply(ARL, c(1), var)/m)
print(list(simMean, simSE, time))
parallel::stopCluster(cl)
df.sim2 <- data.frame("RA"=2, "h"=h, "ARL"=simMean, "ARLSE"=simSE, "nsim"=m)

rbind(df.sim1, df.sim2)

## End(Not run)
```

`racusum_crit_mc` *Compute alarm threshold of RA-CUSUM control chart using Markov chain approximation*

Description

Computes alarm threshold of a risk-adjusted cumulative sum control chart using Markov chain approximation.

Usage

```
racusum_crit_mc(pmix, L0, RA, RQ, scaling = 600, rounding = "p",
  method = "Toep", jmax = 4, verbose = FALSE)
```

Arguments

`pmix`	Numeric Matrix. A three column matrix. First column is the risk score distribution. Second column are the predicted probabilities from the risk model. Third column can be either the predicted probabilities from the risk model or average outcome per risk score, see examples.
`L0`	Double. Prespecified Average Run Length.
`RA`	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`. Odds ratio of death under the null hypotheses is `1`.

`RQ`	Double. Defines the true performance of a surgeon with the odds ratio ratio of death `RQ`. Use `RQ = 1` to compute the in-control ARL and other values to compute the out-of-control ARL.
`scaling`	Double. The `scaling` parameter controls the quality of the approximation, larger values achieve higher accuracy but increase the computation burden (larger transition probability matrix).
`rounding`	Character. If `rounding = "p"` a paired rounding implementation similar to *Webster and Pettitt (2007)* is used, if `rounding = "s"` a simple rounding method of *Steiner et al. (2000)* is used.
`method`	Character. If `method = "Toep"` a combination of Sequential Probability Ratio Test and Toeplitz matrix structure is used to calculate the ARL. `"ToepInv"` computes the inverted matrix using Toeplitz matrix structure. `"BE"` solves a linear equation system using the classical approach of *Brook and Evans (1972)* to calculate the ARL.
`jmax`	Integer. Number of digits for grid search.
`verbose`	Logical. If `FALSE` a quiet calculation of `h` is done. If `TRUE` verbose output of the search procedure is included.

Details

Determines the control limit for given in-control ARL (`"L0"`) using `racusum_arl_mc` by applying a grid search.

Value

Returns a single value which is the control limit `h` for a given In-control ARL.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Brook D and Evans DA (1972) An approach to the probability distribution of CUSUM run length. *Biometrika*, **59**(3), pp. 539–549

Webster RA and Pettitt AN (2007) Stability of approximations of average run length of risk-adjusted CUSUM schemes using the Markov approach: comparing two methods of calculating transition probabilities. *Communications in Statistics - Simulation and Computation* **36**(3), pp. 471–482

Examples

```
## Not run:
library(vlad)
library(dplyr)
data("cardiacsurgery", package = "spcadjust")
```

```
## preprocess data to 30 day mortality and subset phase I (In-control) of surgeons 2
S2I <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II"))) %>%
  filter(phase == "I", surgeon == 2) %>% select(s, y)

## estimate risk model, get relative frequences and probabilities
mod1 <- glm(y ~ s, data = S2I, family = "binomial")
fi  <- as.numeric(table(S2I$s) / length(S2I$s))
usi <- sort(unique(S2I$s))
pi1 <- predict(mod1, newdata = data.frame(s = usi), type = "response")

## set up patient mix
pmix  <- data.frame(fi, pi1, pi1)

## control limit for detecting deterioration RA = 2:
racusum_crit_mc(pmix = pmix, L0 = 740, RA = 2, RQ = 1)
## control limit for detecting improvement RA = 1/2:
racusum_crit_mc(pmix = pmix, L0 = 740, RA = 0.5, RQ = 1)

## End(Not run)
```

---

racusum_crit_sim — *Compute alarm threshold of RA-CUSUM control charts using simulation*

Description

Compute alarm threshold of risk-adjusted cumulative sum control charts using simulation.

Usage

```
racusum_crit_sim(L0, df, coeff, R0 = 1, RA = 2, m = 100,
  yemp = TRUE, nc = 1, jmax = 4, verbose = FALSE)
```

Arguments

L0	Double. Prespecified in-control Average Run Length.
df	Data Frame. First column are Parsonnet Score values within a range of `0` to `100` representing the preoperative patient risk. The second column are binary (`0/1`) outcome values of each operation.
coeff	Numeric Vector. Estimated coefficients α and β from the binary logistic regression model.
R0	Double. Odds ratio of death under the null hypotheses.
RA	Double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio `RA = 2`. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death `RA = 1/2`.

m	Integer. Number of simulation runs.
yemp	Logical. If TRUE, use emirical outcome values, else use model.
nc	Integer. Number of cores used for parallel processing.
jmax	Integer. Number of digits for grid search.
verbose	Logical. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

The function racusum_crit_sim determines the control limit h for given in-control ARL (L0) by applying a multi-stage search procedure which includes secant rule and the parallel version of racusum_arl_sim using mclapply.

Value

Returns a single value which is the control limit h for a given in-control ARL.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441–452.

Wittenberg P, Gan FF, Knoth S (2018). A simple signaling rule for variable life-adjusted display derived from an equivalent risk-adjusted CUSUM chart. *Statistics in Medicine*, **37**(16), pp 2455–2473.

Examples

```
## Not run:
library(vlad)
library("dplyr")
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset phase I (In-control) of surgeons 2
S2I <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II"))) %>%
  filter(phase == "I", surgeon == 2) %>% select(s, y)

## estimate coefficients from logit model
coeff1 <- round(coef(glm(y ~ s, data = S2I, family = "binomial")), 3)

## control limit for detecting deterioration RA = 2:
racusum_crit_sim(L0 = 740, df = S2I, coeff = coeff1, m = 10^3, nc = 4)

## End(Not run)
```

racusum_scores *Compute CUSUM scores based on the log-likelihood ratio statistic*

Description

Compute CUSUM scores based on the log-likelihood ratio statistic.

Usage

```
racusum_scores(wt1, wt2, reset = FALSE, h1 = NULL, h2 = NULL)
```

Arguments

`wt1`	Double. Log-likelihood ratio scores from function `llr_score` for upper CUSUM.
`wt2`	Double. Log-likelihood ratio scores from function `llr_score` for lower CUSUM.
`reset`	Logical. If `FALSE` CUSUM statistic is not reset. If `TRUE` CUSUM statistic is reset to `0` after a signal is issued.
`h1`	Double. Upper control limit of the CUSUM chart.
`h2`	Double. Lower control limit of the CUSUM chart.

Value

Returns a list with two components for the CUSUM scores.

Author(s)

Philipp Wittenberg

References

Steiner SH, Cook RJ, Farewell VT and Treasure T (2000). Monitoring surgical performance using risk-adjusted cumulative sum charts. *Biostatistics*, **1**(4), pp. 441-452.

Parsonnet V, Dean D, Bernstein AD (1989). A method of uniform stratification of risk for evaluating the results of surgery in acquired adult heart disease. *Circulation*, **79**(6):I3-12.

Rigdon SE and Fricker RD (2015). Health Surveillance. In Chen DG and Wilson J (eds) *Innovative Statistical Methods for Public Health Data*, pp. 203–249. Springer, Cham.

Examples

```
## Not run:
#' library(vlad)
# patient Cusum values with different odds ratios, see Rigdon and Fricker p. 225, 226
coeff <- c("(Intercept)" = -3.67, "Parsonnet" = 0.077)
wt1 <- round(llr_score(df = data.frame(19L, 0), coeff = coeff, R0 = 1, RA = 2), 4)
wt2 <- round(llr_score(df = data.frame(19L, 0), coeff = coeff, R0 = 1, RA = 1/2), 5)
all.equal(racusum_scores(wt1 = wt1, wt2 = wt2), list(s1 = 0, s1l = 0.05083))
```

```
library("dplyr")
library("tidyr")
library(ggplot2)
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset phase I (In-control)
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

## subset phase I (In-control)
SI <- filter(SALL, phase == "I") %>% select(s, y)

## estimate coefficients from logit model
coeff1 <- round(coef(glm(y ~ s, data = SI, family = "binomial")), 3)

## subset phase II of surgeons 2
S2II <- filter(SALL, phase == "II", surgeon == 2) %>% select(s, y)
n <- nrow(S2II)

## CUSUM statistic without reset
wt1 <- sapply(1:n, function(i) llr_score(S2II[i, c("s", "y")], coeff = coeff, RA = 2))
wt2 <- sapply(1:n, function(i) llr_score(S2II[i, c("s", "y")], coeff = coeff, RA = 1/2))
cv <- racusum_scores(wt1 = wt1, wt2 = wt2)
s1 <- cv$s1; s1l <- cv$s1l
dm1 <- data.frame(cbind("n" = 1:length(s1), "Cup" = s1, "Clow" = -s1l, "h1" = 2, "h2" = -2))

## CUSUM statistic reset after signal
cv <- racusum_scores(wt1 = wt1, wt2 = wt2, reset = TRUE, h1 = 2, h2 = 2)
s1 <- cv$s1; s1l <- cv$s1l
dm2 <- data.frame(cbind("n" = 1:length(s1), "Cup" = s1, "Clow" = -s1l, "h1" = 2, "h2" = -2))

## plot
dm3 <- bind_rows(dm1, dm2, .id = "type")
dm3$type <- recode_factor(dm3$type, `1`="No resetting", `2`="Resetting")
dm3 %>%
  gather("CUSUM", value, c(-n, - type)) %>%
  ggplot(aes(x = n, y = value, colour = CUSUM, group = CUSUM)) +
  geom_hline(yintercept = 0, colour = "darkgreen", linetype = "dashed") +
  geom_line(size = 0.5) +
  facet_wrap( ~ type, ncol = 1, scales = "free") +
  labs(x = "Patient number n", y = "CUSUM values") + theme_classic() +
  scale_y_continuous(sec.axis = dup_axis(name = NULL, labels = NULL)) +
  scale_x_continuous(sec.axis = dup_axis(name = NULL, labels = NULL)) +
  guides(colour = "none") +
  scale_color_manual(values = c("blue", "orange", "red", "red"))

## End(Not run)
```

search_delta *Search Box-Cox transformation parameter*

Description

Search Box-Cox transformation parameter.

Usage

```
search_delta(s, y, type = "ML", dmin = -2, dmax = 2)
```

Arguments

s	Integer vector. Parsonnet Score values within a range of 0 to 100 representing the preoperative patient risk.
y	Double. Binary (0/1) outcome values of each operation.
type	Character. If type = "ML" Maximum Likelihood used to search the Box-Cox transformation parameter, type = "Pearson" uses a Pearson measure.
dmin	Double. Minimum value for the grid search.
dmax	Double. Maximum value for the grid search.

Value

Returns a single value for the Box-Cox transformation parameter.

Author(s)

Philipp Wittenberg

Examples

```
## Not run:
## load data
data("cardiacsurgery", package = "spcadjust")

## preprocess data to 30 day mortality and subset data to
## phase I (In-control) and phase II (monitoring)
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))

## subset phase I (In-control)
SI <- filter(SALL, phase == "I") %>% select(s, y)

## search delta
dML <- search_delta(SI$s, SI$y, type = "ML")
dQQ <- search_delta(SI$s, SI$y, type = "Pearson")

## show Log-likelihood (ell()) and Pearson measure (QQ()) for each delta
delta <- c(-2, -1, 0, dML, dQQ, 0.5, 1, 2)
r <- sapply(delta, function(i) rbind(i, ell(SI$s, SI$y, i), QQ(SI$s, SI$y, i)))
rownames(r) <- c("d", "l", "S")
t(r)
data.frame(t(r)) %>% filter(l == max(l) | S == min(S))
```

```
## End(Not run)
```

`trafo` *Box-Cox transformation of data.*

Description

Box-Cox transformation of data.

Usage

```
trafo(delta, x)
```

Arguments

`delta`	Numeric. Box-Cox transformation parameter.
`x`	Numceric Vector. Parsonnet Score values within a range of 0 to 100 representing the preoperative patient risk.

Value

Returns a transformed Numeric vector.

Author(s)

Philipp Wittenberg

Index