# Package ‘vlad’

August 24, 2020

**Type** Package

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

**Version** 0.2.1

**Depends** R (>= 2.10)

**Maintainer** Philipp Wittenberg <pwritten@hsu-hh.de>

**URL** [https://github.com/wittenberg/vlad](https://github.com/wittenberg/vlad)

**BugReports** [https://github.com/wittenberg/vlad/issues](https://github.com/wittenberg/vlad/issues)


**License** GPL (>= 2)

**NeedsCompilation** yes

**SystemRequirements** C++11

**Encoding** UTF-8

**LazyData** true

**Imports** Rcpp (>= 0.12.12), checkmate, magrittr, dplyr, tidyr

**Suggests** ggplot2, rmarkdown, spcadjust (>= 1.1), testthat, knitr

**LinkingTo** BH, Rcpp, RcppArmadillo

**RoxygenNote** 7.1.1

**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** 2020-08-24 16:00:03 UTC
R topics documented:

vlad-package ................................................................. 2
bcusum_arl_sim ............................................................... 3
bcusum_crit_sim ............................................................. 3
ell ............................................................................... 5
eocusum_ad_sim ............................................................. 6
eocusum_arl_sim ............................................................ 7
eocusum_crit_sim ........................................................... 10
eocusum_scores ............................................................. 12
llr_score ....................................................................... 14
optimal_k ..................................................................... 15
QQ ................................................................................. 17
racusum_ad_sim ............................................................. 18
racusum_arl_mc .............................................................. 19
racusum_arl_sim ............................................................ 21
racusum_betabinomial_arl_sim .......................................... 23
racusum_betabinomial_crit_sim ......................................... 24
racusum_beta_arl_int ...................................................... 26
racusum_beta_arl_mc ....................................................... 27
racusum_beta_arl_sim ...................................................... 28
racusum_beta_crit_mc ...................................................... 29
racusum_beta_crit_sim ..................................................... 30
racusum_crit_mc ............................................................. 32
racusum_crit_sim .......................................................... 34
racusum_discretebeta_arl_sim .......................................... 36
racusum_discretebeta_crit_sim ......................................... 37
racusum_scores ............................................................. 39
search_delta ............................................................... 41
trafo ........................................................................... 42

Index 44

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

Description

**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
**ell**

*Estimated log-likelihood.*

**Description**

Estimated log-likelihood.

**Usage**

```r
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**

```r
## 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_ad_sim

**Description**

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

**Usage**

```r
eocusum_ad_sim(r, pmix, k, h, RQ = 1, side = "low", type = "cond", m = 50)
```

**Arguments**

- `r` Integer. Number of simulation runs.
- `pmix` Data Frame. A three column data frame. First column is the operation outcome. 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.
- `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.
- `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.
- `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**


Examples

```r
## 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'Ecuier-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_arl_sim`

$$\text{Compute ARLs of EO-CUSUM control charts using simulation}$$

Description

Compute ARLs of EO-CUSUM control charts using simulation.
Usage

eocusum_arl_sim(r, pmix, k, h, RQ = 1, yemp = FALSE, side = "low")

Arguments

- **r**
  Integer. Number of simulation runs.

- **pmix**
  Data Frame. A three column data frame. First column is the operation outcome. 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.

- **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.

- **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.

- **yemp**
  Logical. If TRUE use observed outcome value, if FALSE use estimated binary logistic 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


Examples

```r
## 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'Ecuery-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()
c1 <- 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

```r
eocusum_crit_sim(
  L0,
  pmix,
  k,
  RQ = 1,
  side = "low",
  yemp = FALSE,
  m = 10000,
  nc = 1,
  hmax = 30,
  jmax = 4,
  verbose = FALSE
)
```

Arguments

- **L0**: Double. Prespecified in-control Average Run Length.
- **pmix**: Data Frame. A three column data frame. First column is the operation outcome. 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.
- **k**: Double. Reference value of the CUSUM control chart. Either 0 or a positive value. Can be determined with function `optimal_k`.
- **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.
- **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.
- **yemp**: Logical. If TRUE use observed outcome value, if FALSE use estimated binary logistic regression model.
- **m**: Integer. Number of simulation runs.
- **nc**: Integer. Number of cores used for parallel processing. Value is passed to `parSapply`.
- **hmax**: Integer. Maximum value of h for the grid search.
jmax

verbose

Details

Determines the control limit ("h") for given in-control ARL ("L0") applying a grid search using eocusum_arl_sim and parSapply.

Value

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

Author(s)

Philipp Wittenberg

References


Examples

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

## preprocess data to 30 day mortality
SALL <- cardiacsurgery %>%
    rename(s = Parsonnet) %>%
    mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
           phase = factor(ifelse(date < 2*365, "I", "II")))
SI <- subset(SALL, phase == "I")
y <- subset(SALL, select = y)
GLM <- glm(y ~ s, data = SI, family = "binomial")
pi1 <- predict(GLM, type = "response", newdata = data.frame(s = SALL$s))
pmix <- data.frame(y, pi1, pi1)

## (Deterioration)
kopt <- optimal_k(pmix = pmix, RA = 2)
h <- eocusum_crit_sim(L0=370, pmix=pmix, k=kopt, side = "low", verbose=TRUE, nc=4)

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

## (Improvement)
kopt <- optimal_k(pmix = pmix, RA = 1/2)
h <- eocusum_crit_sim(L0=370, pmix=pmix, k=kopt, side = "up", verbose=TRUE, nc=4)

## parameters to set up a tabular CUSUM or V-Mask (lower arm)
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 deterioration 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


Examples

```r
## 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 <- subset(SALL, phase == "I")

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

## set up patient mix
pi1 <- predict(GLM, type = "response", newdata = data.frame(s = SI$s))

## determine k for detecting improvement
k1opt <- optimal_k(pmix=pi1, RA = 1/2)

## determine k for detecting deterioration
k2opt <- optimal_k(pmix=pi1, RA = 2)

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

z <- predict(GLM, type = "response", newdata = data.frame(s = S2II$s)) - S2II$y

## CUSUM statistic without reset

## CUSUM statistic after signal

dm3 <- bind_rows(dm1, dm2, .id = "type")
dm3$type <- recode_factor(dm3$type, c("1"="No resetting", "2"="Resetting")

## CUSUM statistic reset after signal

dm3 <- bind_rows(dm1, dm2, .id = "type")
dm3$type <- recode_factor(dm3$type, c("1"="No resetting", "2"="Resetting")

## CUSUM statistic reset after signal
```
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 $\alpha$ and $\beta$ 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 logistic regression model.

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

Author(s)
Philipp Wittenberg
optimal_k

Compute approximate optimal k

References


Examples

```r
## 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)
```
Description

Compute approximate optimal k.

Usage

optimal_k(pmix, RA, yemp = FALSE)

Arguments

- **pmix**: Data Frame. A three column data frame. First column is the operation outcome. 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.

- **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.

- **yemp**: Logical. If TRUE, use empirical outcome values, else use model.

Details

Formula deterioration:

\[ k_{det} = \frac{RA - 1 - \log(RA)}{\log(RA)} \bar{p}, RA > 1 \]

Formula improvement:

\[ k_{imp} = \frac{1 - RA + \log(RA)}{\log(RA)} \bar{p}, RA < 1 \]

Value

Returns a single value which is the approximate optimal k.

Author(s)

Philipp Wittenberg

References

Examples

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

## preprocess data to 30 day mortality
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
  mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
         phase = factor(ifelse(date < 2*365, "I", "II")))
SI <- subset(SALL, phase == "I")
GLM <- glm(y ~ s, data = SI, family = "binomial")
pi1 <- predict(GLM, type = "response", newdata = data.frame(s = SI$s))

## (Deterioration)
optimal_k(pmix = pmix, RA = 2)

## manually find optimal k for detecting deterioration
RA <- 2
pbar <- mean(pmix$pi1)
kopt <- pbar * ( RA - 1 - log(RA) ) / log(RA)
all.equal(kopt, optimal_k(pmix = pmix, RA = 2))

## (Improvement)
optimal_k(pmix = pmix, RA = 1/2)

## manually find optimal k for detecting improvement
RA <- 1/2
pbar <- mean(pmix$pi1)
kopt <- pbar * ( 1 - RA + log(RA) ) / log(RA)
all.equal(kopt, optimal_k(pmix = pmix, RA = 1/2))

## 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

```r
## 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)
```

Description

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

Usage

```r
racusum_ad_sim(r, pmix, h, RA = 2, RQ = 1, m = 50, type = "cond")
```
**Arguments**

- **r**  
  Integer Vector. Number of runs.

- **pmix**  
  Data Frame. A three column data frame. First column is the operation outcome. 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.

- **h**  
  Double. Control Chart limit for detecting deterioration/improvement.

- **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.

- **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

---

**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**

```r
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 of Knoth et al. (2019) 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 Average Run Length.

Author(s)

Philipp Wittenberg

References


Examples

```r
## 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 frequencies 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)
```

Description

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

racusum_arl_sim(r, pmix, h, RA = 2, RQ = 1, yemp = FALSE)

Arguments

r    Integer Vector. Number of runs.

pmix Data Frame. A three column data frame. First column is the operation outcome. 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.

h Double. Control Chart limit for detecting deterioration/improvement.

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.

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

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References


Examples

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

set.seed(1234)
SALLI <- cardiacsurgery %>% mutate(s = Parsonnet) %>%
mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
```r
phase = factor(ifelse(date < 2*365, "I", "II"))
filter(phase == "I")
select(s, y)

## estimate risk model, get relative frequencies and probabilities
mod1 <- glm(y ~ s, data = SALLI, family = "binomial")
y <- SALLI$y
pi1 <- fitted.values(mod1)

## set up patient mix (risk model)
pi1 <- data.frame(y, pi1, pi1)
h <- 2.75599
m <- 1e4
RLS <- sapply(1:m, racusum_arl_sim, h=h, pmix=pi1, RA=2)
data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m), h, m))

## End(Not run)
```

---

**racusum_betabinomial_arl_sim**

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

**Description**

Compute ARLs of RA-CUSUM control charts using simulation.

**Usage**

```r
racusum_betabinomial_arl_sim(
  r,
  shape1,
  shape2,
  coeff,
  h,
  RA = 2,
  rs = 71,
  RQ = 1
)
```

**Arguments**

- `r` Integer Vector. Number of runs.
- `shape1` Double. Shape parameter $\alpha > 0$ of the beta-binomial distribution.
- `shape2` Double. Shape parameter $\beta > 0$ of the beta-binomial distribution.
- `coeff` Numeric Vector. Estimated intercept and slope coefficients from a binary logistic regression model.
- `h` Double. Control Chart limit for detecting deterioration/improvement.
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 $RA = 1/2$.

**rs**
Integer. Maximum risk score.

**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

**Examples**
```r
## Not run:
library(vlad)
m <- 1e3
RLS <- sapply(1:m, racusum_betabinomial_arl_sim, shape1=1, shape2=3, coeff=c(-3.6798, 0.0768),
h=4.5, RA=2, rs=71, RQ=1)
data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m)))
## End(Not run)
```

**Description**
Compute alarm threshold of RA-CUSUM control charts using simulation.

**Usage**
```r
racusum_betabinomial_crit_sim(
  L0,
  shape1,
  shape2,
  coeff,
  RA = 2,
  rs = 71,
  RQ = 1,
  m = 10000,
)```
nc = 1,
    hmax = 30,
    jmax = 4,
    verbose = FALSE
  )

Arguments

L₀ Double. Prespecified in-control Average Run Length.
shape1 Double. Shape parameter $\alpha > 0$ of the beta-binomial distribution.
shape2 Double. Shape parameter $\beta > 0$ of the beta-binomial distribution.
coeff Numeric Vector. Estimated intercept and slope coefficients from a binary logistic regression model.
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 $RA = 1/2$.
rs Integer. Maximum risk score.
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 used for parallel processing. Value is passed to parSapply.
hmax Integer. Maximum value of $h$ for the grid search.
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

Determines the control limit ("h") for given in-control ARL ("L₀") applying a grid search using racusum_betabinomial_arl_sim and parSapply.

Value

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

Author(s)

Philipp Wittenberg
Examples

```r
## Not run:
library(vlad)
racusum_betabinomial_crit_sim(L0=100, shape1=1, shape2=3, coeff=c(-3.6798, 0.0768), RA = 2, 
rs = 71, RQ = 1, verbose=TRUE)
## End(Not run)
```

---

`racusum_beta_arl_int` *Compute ARL of RA-CUSUM control charts assuming a beta distributed patient mix with using collocation methods.*

Description

Compute ARL of RA-CUSUM control charts assuming a beta distributed patient mix with using collocation methods.

Usage

```r
racusum_beta_arl_int(h, N, RA, RQ, g0, g1, shape1, shape2, pw)
```

Arguments

- `h`  
  Double. h is the control limit (>0).
- `N`  
  Integer. Number of quadrature nodes, dimension of the resulting linear equation system is equal to N.
- `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.
- `g0`  
  Double. Estimated intercept coefficient from a binary logistic regression model.
- `g1`  
  Double. Estimated slope coefficient from a binary logistic regression model.
- `shape1`  
  Double. Shape parameter $\alpha > 0$ of the beta distribution.
- `shape2`  
  Double. Shape parameter $\beta > 0$ of the beta distribution.
- `pw`  
  Logical. If FALSE full collocation is applied. If TRUE a piece-wise collocation method is used.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg
racusum_beta_arl_mc

Examples

```r
## Not run:
library(vlad)
## full colocation
racusum_beta_arl_int(h=4.5, RA=2, RQ=1, shape1=1, shape2=6, g0=-3.6798, g1=0.0768*71, N=50, pw=FALSE)
## piece-wise colocation
racusum_beta_arl_int(h=4.5, RA=2, RQ=1, shape1=1, shape2=6, g0=-3.6798, g1=0.0768*71, N=49, pw=TRUE)
## End(Not run)
```

---

**racusum_beta_arl_mc**  
Compute ARL of RA-CUSUM control charts assuming patient mix with beta distribution using Markov chain approximation.

**Description**

Compute ARL of risk-adjusted-CUSUM control charts assuming patient mix with beta distribution using Markov chain approximation.

**Usage**

```r
racusum_beta_arl_mc(h, RA, g0, g1, shape1, shape2, r = 600, method = 1, RQ = 1)
```

**Arguments**

- `h`  
  Double. h is the control limit (>0).

- `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.

- `g0`  
  Double. Estimated intercept coefficient from a binary logistic regression model.

- `g1`  
  Double. Estimated slope coefficient from a binary logistic regression model.

- `shape1`  
  Double. Shape parameter α > 0 of the beta distribution.

- `shape2`  
  Double. Shape parameter β > 0 of the beta distribution.

- `r`  
  Double. Matrix system dimension.

- `method`  
  Character. If method = "1" a combination of Sequential Probability Ratio Test and Toeplitz matrix structure is used to calculate the ARL. "2" solves a linear equation system using the classical approach of Brook and Evans (1972) to calculate the ARL.

- `RQ`  
  Double. Defines the performance of a surgeon with the odds ratio ratio of death.
racusum_beta_arl_sim

**Value**

Returns a single value which is the Run Length.

**Author(s)**

Philipp Wittenberg

**Examples**

```r
## Not run:
library(vlad)
racusum_beta_arl_mc(h=4.5, RA=2, g0=-3.6798, g1=0.0768*71, shape1=1, shape2=6, r=1e4, method=1)
## End(Not run)
```

---

**Description**

Compute ARLs of RA-CUSUM control charts using simulation.

**Usage**

```r
racusum_beta_arl_sim(r, shape1, shape2, coeff, h, RA = 2, rs = 71, RQ = 1)
```

**Arguments**

- `r`: Integer Vector. Number of runs.
- `shape1`: Double. Shape parameter $\alpha > 0$ of the beta distribution.
- `shape2`: Double. Shape parameter $\beta > 0$ of the beta distribution.
- `coeff`: Numeric Vector. Estimated coefficients $\alpha$ and $\beta$ from the binary logistic regression model.
- `h`: Double. Control Chart limit for detecting deterioration/improvement.
- `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 $RA = 1/2$.
- `rs`: Integer. Maximum risk score.
- `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

Examples

```r
## Not run:
library(vlad)

m <- 1e3
RLS <- sapply(1:m, racusum_beta_arl_sim, h=4.5, shape1=1, shape2=3, coeff=c(-3.6798, 0.0768),
RA = 2, RQ = 1)
data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m)))

## End(Not run)
```

Description

Compute alarm threshold of RA-CUSUM control chart assuming patient mix with beta distribution using Markov chain approximation.

Usage

```r
racusum_beta_crit_mc(
  L0,
  RA,
  g0,
  g1,
  shape1,
  shape2,
  method = 1,
  r = 600,
  jmax = 4,
  verbose = TRUE,
  RQ = 1
)
```

Arguments

- **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.
Racsusum Beta Crit Sim

Description

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

Details

Determines the control limit for given in-control ARL ("L0") using racusum_beta_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

Examples

```r
## Not run:
library(vlad)
racusum_beta_crit_mc(L0=7500, RA=2, g0=-3.6798, g1=0.0768*71, shape1=.61, shape2=4.09, r=1000)
## End(Not run)
```
Usage

```r
racusum_beta_crit_sim(
  L0,
  shape1,
  shape2,
  coeff,
  rs = 71,
  RA = 2,
  RQ = 1,
  nc = 1,
  hmax = 30,
  jmax = 4,
  m = 10000,
  verbose = FALSE
)
```

Arguments

- **L0**: Double. Prespecified in-control Average Run Length.
- **shape1**: Double. Shape parameter $\alpha > 0$ of the beta distribution.
- **shape2**: Double. Shape parameter $\beta > 0$ of the beta distribution.
- **coeff**: Numeric Vector. Estimated coefficients $\alpha$ and $\beta$ from the binary logistic regression model.
- **rs**: Integer. Maximum risk score.
- **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 $Q$.
- **nc**: Integer. Number of cores used for parallel processing. Value is passed to `parSapply`.
- **hmax**: Integer. Maximum value of $h$ for the grid search.
- **jmax**: Integer. Number of digits for grid search.
- **m**: Integer. Number of simulation runs.
- **verbose**: Logical. If TRUE verbose output is included, if FALSE a quiet calculation of $h$ is done.

Details

Determines the control limit ("h") for given in-control ARL ("L0") applying a grid search using `racusum_beta_arl_sim` and `parSapply`.

Value

Returns a single value which is the control limit $h$ for a given in-control ARL.
Author(s)
Philipp Wittenberg

Examples

```r
## Not run:
library(vlad)
racusum_beta_crit_sim(L0=500, shape1=1, shape2=3, coeff=c(-3.6798, 0.0768), RA = 2, rs = 71, RQ = 1, verbose=TRUE, m=1e3)
## 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

```r
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 of Knoth et al. (2019) 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 ("\(L_0\)) 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


## Examples

```r
## 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

```r
racusum_crit_sim(
  L0,
  pmix,
  RA = 2,
  RQ = 1,
  yemp = FALSE,
  m = 10000,
  nc = 1,
  hmax = 30,
  jmax = 4,
  verbose = FALSE
)
```
Arguments

L0 Double. Prespecified in-control Average Run Length.

pmix Data Frame. A three column data frame. First column is the operation outcome. 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.

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.

yemp Logical. If TRUE, use empirical outcome values, else use model.

m Integer. Number of simulation runs.

nc Integer. Number of cores used for parallel processing. Value is passed to \texttt{parSapply}.

hmax Integer. Maximum value of \texttt{h} for the grid search.

jmax Integer. Number of digits for grid search.

verbose Logical. If TRUE verbose output is included, if FALSE a quiet calculation of \texttt{h} is done.

Details

Determines the control limit ("\texttt{h}") for given in-control ARL ("L0") applying a grid search using \texttt{racusum_arl_sim} and \texttt{parSapply}.

Value

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

Author(s)

Philipp Wittenberg

References


Examples

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

## preprocess data to 30 day mortality
SALL <- cardiacsurgery %>% rename(s = Parsonnet) %>%
   mutate(y = ifelse(status == 1 & time <= 30, 1, 0),
      phase = factor(ifelse(date < 2*365, "I", "II")))
SI <- subset(SALL, phase == "I")
y <- subset(SALL, select = y)
GLM <- glm(y ~ s, data = SI, family = "binomial")
pi1 <- predict(GLM, type = "response", newdata = data.frame(s = SALL$s))

## End(Not run)

racsdiscbeta_arl_sim

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

### Description

Compute ARLs of RA-CUSUM control charts using simulation.

#### Usage

```r
racsdiscbeta_arl_sim(
  r,
  shape1,
  shape2,
  coeff,
  h,
  RA = 2,
  rs = 72,
  RQ = 1
)
```

#### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>r</code></td>
<td>Integer Vector. Number of runs.</td>
</tr>
<tr>
<td><code>shape1</code></td>
<td>Double. Shape parameter $\alpha &gt; 0$ of the beta distribution.</td>
</tr>
<tr>
<td><code>shape2</code></td>
<td>Double. Shape parameter $\beta &gt; 0$ of the beta distribution.</td>
</tr>
<tr>
<td><code>coeff</code></td>
<td>Numeric Vector. Estimated intercept and slope coefficients from a binary logistic regression model.</td>
</tr>
</tbody>
</table>
racusum_discretebeta_crit_sim

**Description**

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

**Usage**

```r
cacusum_discretebeta_crit_sim(
L0,
shape1,
shape2,
coeff,
rs = 72,
RA = 2,
h,
RA
```

- `h` Double. Control Chart limit for detecting deterioration/improvement.
- `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.
- `rs` Integer. Number of intervals between 0 and the maximum risk score.
- `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

**Examples**

```r
## Not run:
library(vlad)
m <- 1e3
RLS <- sapply(1:m, racusum_discretebeta_arl_sim, shape1=1, shape2=3, coeff=c(-3.6798, 0.0768),
h=4.5, RA=2, rs=71+1, RQ=1)
data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m)))
## End(Not run)
```
RQ = 1,
nc = 1,
hmax = 30,
jmax = 4,
m = 10000,
verbose = FALSE
)

Arguments

L0 Double. Prespecified in-control Average Run Length.
shape1 Double. Shape parameter $\alpha > 0$ of the beta distribution.
shape2 Double. Shape parameter $\beta > 0$ of the beta distribution.
coeff Numeric Vector. Estimated intercept and slope coefficients from a binary logistic regression model.
rs Integer. Number of intervals between $0$ and the maximum risk score.
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.
nc Integer. Number of cores used for parallel processing. Value is passed to `parSapply`.
hmax Integer. Maximum value of $h$ for the grid search.
jmax Integer. Number of digits for grid search.
m Integer. Number of simulation runs.
verbose Logical. If TRUE verbose output is included, if FALSE a quiet calculation of $h$ is done.

Details

Determines the control limit ($"h"$) for given in-control ARL ($"L0"$) applying a grid search using `racusum_discretebeta_arl_sim` and `parSapply`.

Value

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

Author(s)

Philipp Wittenberg
Examples

```r
## Not run:
library(vlad)
racusum_discretebeta_crit_sim(L0=7500, shape1=0.61, shape2=4.09, rs=(71+1),
coeff=c(-3.6798, 0.0768), RA=2, RQ=1, nc=4, verbose=TRUE, m=1e3)
## 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**

```r
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**


Examples

```r
## 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 = "spcadj")

## 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)) +
```
search_delta

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 \hspace{1cm} Search Box-Cox transformation parameter

**Description**

Search Box-Cox transformation parameter.

**Usage**

```r
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

**References**

Examples

```r
## 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

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

### Arguments

- `delta` Numeric. Box-Cox transformation parameter.
- `x` Numeric 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

bcusum_arl_sim, 3, 4
bcusum_crit_sim, 3

ell, 5
eocusum_ad_sim, 6
eocusum_arl_sim, 7, 11
eocusum_crit_sim, 10
eocusum_scores, 12

llr_score, 14, 39

mclapply, 4

optimal_k, 6, 8, 10, 12, 15

parSapply, 10, 11, 25, 31, 35, 38

QQ, 17

racusum_ad_sim, 18
racusum_arl_mc, 19, 33
racusum_arl_sim, 21, 35
racusum_beta_arl_int, 26
racusum_beta_arl_mc, 27, 30
racusum_beta_arl_sim, 28, 31
racusum_beta_crit_mc, 29
racusum_beta_crit_sim, 30
racusum_betabinomial_arl_sim, 23, 25
racusum_betabinomial_crit_sim, 24
racusum_crit_mc, 32
racusum_crit_sim, 34
racusum_discretebeta_arl_sim, 36, 38
racusum_discretebeta_crit_sim, 37
racusum_scores, 39

search_delta, 41

trafo, 42

vlad-package, 2