Package ‘vlad’

October 12, 2022

Type Package

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

Version 0.2.2

Depends R (>= 2.10)

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

URL `https://github.com/wittenberg/vlad`

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 with fast and accurate Markov chain approximations developed in Knoth et al. (2019) <doi:10.1002/sim.8104>.

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 2021-02-15 10:30:02 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 with fast and accurate Markov chain approximations developed in Knoth et al. (2019) <doi:10.1002/sim.8104>.

`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

`compute_vmask` *Compute V-Masks arms, nose and alarm points*

Description

Function for plotting truncated symeterical/asymetrical vmask

Usage

```
compute_vmask(z, d1, d2, theta1, theta2)
```

Arguments

`z`	Numeric Vector. ...
`d1`	Double. For the XYZ CUSUM Distance d from vertex of V-Mask. d=h/k
`d2`	Double. For the XYZ CUSUM Distance d from vertex of V-Mask. d=h/k
`theta1`	Double. Angle ...
`theta2`	Double. Angle ...

Value

...

Author(s)

Philipp Wittenberg

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

Arguments

`r`	Integer. Number of 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

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_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, pmix, k, h, RQ = 1, yemp = FALSE, side = "low")
```

Arguments

`r`	Integer. Number of 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 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, 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 logistc 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`	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 `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

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:
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 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 <- 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$s))
pmix <- data.frame(SI$y, pi1, pi1)

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

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

## subset phase II of surgeons 2
S2II <- filter(SALL, phase == "II", surgeon == 2) %>% select(s, y)
n <- nrow(S2II)
z <- predict(GLM, type = "response", newdata = data.frame(s = S2II$s))-S2II$y

## CUSUM statistic without reset
cv <- eocusum_scores(z = z, k1 = k1opt, k2 = k2opt)
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, k1 = k1opt, k2 = k2opt, 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)
```

`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 approximate optimal k*

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 emirical outcome values, else use model.

Details

Formula deterioration:

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

Formula improvement:

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

Value

Returns a single value which is the approximate optimal `k`.

Author(s)

Philipp Wittenberg

References

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
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$s))
pmix <- data.frame(SI$y, pi1, pi1)

## (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

```
## 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_ad_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_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` *ARL of RA-CUSUM charts*

Description

Compute the ARL of risk-adjusted CUSUM charts.

Usage

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

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

Arguments

`h`	Double. `h` is the control limit (`>0`).
`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`. Use `RQ = 1` to compute the in-control ARL and other values to compute the out-of-control ARL.
`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.
`r`	Integer. Number of runs.
`yemp`	Logical. If `TRUE` use observed outcome value, if `FALSE` use estimated binary logistc regression model.

Value

Returns a single value which is the Average Run Length for `"racusum_arl_mc"` and the Run Length for `"racusum_arl_sim"`.

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.

Knoth S, Wittenberg P and Gan FF (2019). Risk-adjusted CUSUM charts under model error. *Statistics in Medicine*, **38**(12), pp. 2206–2218.

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.

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")

## Markov Chain
## 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"))
```

```
## Monte Carlo simulation
set.seed(1234)
SALLI <- cardiacsurgery %>% mutate(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")
y <- SALLI$y
pi1 <- fitted.values(mod1)

## set up patient mix (risk model)
pmix <- data.frame(y, pi1, pi1)
h <- 2.75599

m <- 1e4
RLS <- sapply(1:m, racusum_arl_sim, h=h, pmix=pmix, 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

```
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.
`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

```
## 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)
```

`racusum_betabinomial_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_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

```
## 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` *ARL of Beta RA-CUSUM charts*

Description

Compute the ARL of risk-adjusted CUSUM charts assuming a beta distributed patient mix.

Usage

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

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

racusum_beta_arl_sim(h, shape1, shape2, g0, g1, r, RA = 2, RQ = 1, rs = 71)
```

Arguments

`h`	Double. `h` is the control limit (>`0`).
`shape1`	Double. Shape parameter α > `0` of the beta distribution.
`shape2`	Double. Shape parameter β > `0` of the beta distribution.
`g0`	Double. Estimated intercept coefficient from a binary logistic regression model.
`g1`	Double. Estimated slope coefficient 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`. Odds ratio of death under the null hypotheses is `1`.
`RQ`	Double. Defines the performance of a surgeon with the odds ratio ratio of death.
`r`	Integer. Number of runs.
`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.
`N`	Integer. Number of quadrature nodes, dimension of the resulting linear equation system is equal to `N`.
`pw`	Logical. If `FALSE` full collocation is applied. If `TRUE` a piece-wise collocation method is used.
`rs`	Integer. Maximum risk score.

Value

Returns a single value which is the Average Run Length for `"racusum_beta_arl_mc"` and `"racusum_beta_arl_int"`, and the Run Length for `"racusum_beta_arl_sim"`.

Author(s)

Philipp Wittenberg

References

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

Examples

```
## Not run:
library(vlad)
## Markov Chain
racusum_beta_arl_mc(h=4.5, shape1=1, shape2=6, g0=-3.6798, g1=0.0768*71, RA=2, r=1e4)
## Full collocation
racusum_beta_arl_int(h=4.5, shape1=1, shape2=6, g0=-3.6798, g1=0.0768*71, RA=2, RQ=1, N=150,
 pw=FALSE)
## Piece-wise collocation
racusum_beta_arl_int(h=4.5, shape1=1, shape2=6, g0=-3.6798, g1=0.0768*71, RA=2, RQ=1, N=49,
 pw=TRUE)
## Monte Carlo simulation
m <- 1e3
RLS <- sapply(1:m, racusum_beta_arl_sim, h=4.5, shape1=1, shape2=6, g0=-3.6798, g1=0.0768,
RA = 2, RQ = 1, rs = 71)
data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m)))

## End(Not run)
```

`racusum_beta_crit` *Alarm thresholds of Beta RA-CUSUM charts*

Description

Compute alarm threshold of risk-adjusted CUSUM charts assuming a beta distributed patient mix.

Usage

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

racusum_beta_crit_sim(L0, shape1, shape2, g0, g1, RA = 2, RQ = 1, nc = 1,
  rs = 71, hmax = 30, jmax = 4, m = 10000, verbose = FALSE)
```

Arguments

`L0`	Double. Prespecified Average Run Length.
`shape1`	Double. Shape parameter $\alpha > 0$ of the beta distribution.
`shape2`	Double. Shape parameter $\beta > 0$ of the beta distribution.
`g0`	Double. Estimated intercept coefficient from a binary logistic regression model.
`g1`	Double. Estimated slope coefficient 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`. 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.
`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.
`r`	Double. Matrix system dimension.
`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 (see details) is included.
`nc`	Integer. Number of cores used for parallel processing. Value is passed to `parSapply`.
`rs`	Integer. Maximum risk score.
`hmax`	Integer. Maximum value of `h` for the grid search.
`m`	Integer. Number of simulation runs.

Details

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

Value

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

References

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

Examples

```
## Not run:
library(vlad)
## Markov Chain
racusum_beta_crit_mc(L0=7500, shape1=.61, shape2=4.09, g0=-3.6798, g1=0.0768*71, RA=2, RQ=1,
 r=1e3)
## Monte Carlo simulation
racusum_beta_crit_sim(L0=7500, shape1=.61, shape2=4.09, g0=-3.6798, g1=0.0768, RA = 2, RQ = 1,
rs = 71, verbose=TRUE, m=1e3)

## End(Not run)
```

`racusum_crit` *Alarm thresholds of RA-CUSUM charts*

Description

Compute alarm threshold of risk-adjusted CUSUM charts.

Usage

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

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 Average Run Length.
`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`. Use `RQ = 1` to compute the in-control ARL and other values to compute the out-of-control ARL.
`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.

`yemp`	Logical. If `TRUE`, use emirical outcome values, else use 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.

Details

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

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

Knoth S, Wittenberg P and Gan FF (2019). Risk-adjusted CUSUM charts under model error. *Statistics in Medicine*, **38**(12), pp. 2206–2218.

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.

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")

## Markov Chain
## 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)

## Monte Carlo simulation
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)
h <- racusum_crit_sim(pmix = pmix, L0 = 370, RA = 2, nc = 4, verbose = TRUE)

## End(Not run)
```

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

Description

Compute ARLs of RA-CUSUM control charts using simulation.

Usage

```
racusum_discretebeta_arl_sim(r, shape1, shape2, coeff, h, RA = 2, rs = 72,
  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 intercept and slope coefficients from a 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. 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

```
## 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)
```

`racusum_discretebeta_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_discretebeta_crit_sim(L0, shape1, shape2, coeff, rs = 72, 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 α > `0` of the beta distribution.
`shape2`	Double. Shape parameter β > `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

```
## Not run:
library(vlad)
racusum_discretebeta_crit_sim(L0=7500, shape1=.61, shape2=4.09, rs=(71+1),
coeff=c(-3.6798, .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

```
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

References

Knoth S, Wittenberg P and Gan FF (2019). Risk-adjusted CUSUM charts under model error. *Statistics in Medicine*, **38**(12), pp. 2206–2218..

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)
```

`surgery` *Surgical outcome data.*

Description

A data set with the risk scores and surgical outcomes of 2,500 patients.

Usage

```
surgery
```

Format

A data frame with 2500 rows and 2 variables:

s Risk scores

y Binary operation outcome (0=survival, 1=death)

`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

`VMASK3` *Vmask3*

Description

Helper function to compute truncated symeterical/asymetrical vmask

Usage

```
VMASK3(A, B, d1, d2, theta1, theta2, Sn, seg)
```

Arguments

`A`	...
`B`	...
`d1`	Double. For the XYZ CUSUM Distance d from vertex of V-Mask. d=h/k
`d2`	Double. For the XYZ CUSUM Distance d from vertex of V-Mask. d=h/k
`theta1`	Double. Angle ...
`theta2`	Double. Angle ...
`Sn`	...
`seg`	Logical. ...

Value

...

Author(s)

Philipp Wittenberg

Index