Package 'vlad'

March 6, 2018

Type Package

Title Variable Life Adjusted Display

Version 0.1.0

Depends R (>= 2.10)

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

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

Description Contains functions to set up risk-adjusted quality control charts in health care. For the variable life adjusted display (VLAD) proposed by Lovegrove et al. (1997) <doi:10.1016/S0140-6736(97)06507-0> and 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> the average run length and control limits can be computed.

License GPL (>= 2)

NeedsCompilation yes

Encoding UTF-8

LazyData true

Imports Rcpp (>= 0.12.12)

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

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

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

Repository CRAN

Date/Publication 2018-03-06 19:28:52 UTC

R topics documented:

`vlad-package` *Variable Life Adjusted Display*

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> and 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> the average run length and control limits can be computed.

Author(s)

Philipp Wittenberg and Sven Knoth

`calceo` *Compute Expected minus Observed value*

Description

Compute Expected minus Observed value.

Usage

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

Arguments

df	DataFrame. First column Parsonnet Score and second column outcome of each operation.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
yemp	boolean. If TRUE use observed outcome value, if FALSE use estimated binary logistc regression model.

Value

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

Author(s)

Philipp Wittenberg

References

Lovegrove J, Valencia O, Treasure T, Sherlaw-Johnson C and Gallivan S (1997). "Monitoring the results of cardiac surgery by variable life-adjusted display." *The Lancet*, **350**(9085), pp. 1128–1130. doi: 10.1016/S0140-6736(97)06507-0.

Poloniecki J, Valencia O and Littlejohns P (1998). "Cumulative risk adjusted mortality chart for detecting changes in death rate: observational study of heart surgery." *BMJ*, **316**(7146), pp. 1697–1700. doi: 10.1136/bmj.316.7146.1697.

Steiner S (2014). "Risk-Adjusted Monitoring of Outcomes in Health Care." In Lawless JF (ed.), *Statistics in Action*, pp. 225-242. Informa UK Limited. doi: 10.1201/b16597-15.

Examples

```
library("vlad")
# see Steiner (2014) p. 234
coeff <- c("(Intercept)"=-3.68, "Parsonnet"=0.077)
# penalty reward for death (E-O scores multiplied with -1 to get O-E scores)
calceo(df=data.frame(as.integer(0), 1), coeff=coeff)*-1
calceo(df=data.frame(as.integer(50), 1), coeff=coeff)*-1
# penalty reward for survival
calceo(df=data.frame(as.integer(0), 0), coeff=coeff)*-1
calceo(df=data.frame(as.integer(50), 0), coeff=coeff)*-1

# Plot a VLAD/CRAM chart
library("spcadjust")
data("cardiacsurgery")
cardiacsurgery <- dplyr::mutate(cardiacsurgery, phase=factor(ifelse(date < 2*365, "I", "II")))
S2 <- subset(cardiacsurgery, c(surgeon==2), c("phase", "Parsonnet", "status"))
S2I <- subset(S2, c(phase=="I"))
S2II <- subset(S2, c(phase=="II"))
coeff <- coef(glm(status ~ Parsonnet, data=S2I, family="binomial"))
EO <- sapply(1:nrow(S2), function(i) calceo(df=S2[i, c("Parsonnet", "status")], coeff=coeff))
```

```
df1 <- data.frame(cbind(subset(S2, select=c("phase")), "n"=1:nrow(S2), "cEO"=cumsum(EO)))
df2 <- tidyr::gather(df1, "variable", value, c(-n, -phase))

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

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

Description

Compute alarm threshold of the Bernoulli CUSUM control charts using simulation.

Usage

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

Arguments

`L0`	double. Prespecified in-control Average Run Length.
`df`	DataFrame. First column are Parsonnet Score values within a range of zero 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.
`verbose`	boolean. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

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

Value

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

Author(s)

Philipp Wittenberg

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

Description

Compute ARLs of the Bernoulli CUSUM control charts using simulation.

Usage

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

Arguments

`r`	Integer vector. Number of runs.
`h`	double. Control Chart limit for detecting deterioration/improvement.
`df`	DataFrame. First column are Parsonnet Score values within a range of zero 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

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

Description

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

Usage

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

Arguments

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

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

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

Taylor HM (1968). "The Economic Design of Cumulative Sum Control Charts." *Technometrics*, **10**(3), pp. 479-488. doi: 10.1080/00401706.1968.10490595.

Crosier R (1986). "A new two-sided cumulative quality control scheme." *Technometrics*, **28**(3), pp. 187-194. doi: 10.2307/1269074.

Examples

```
## Not run:
library("vlad")
library("spcadjust")
## Datasets
data("cardiacsurgery")
s5000 <- dplyr::sample_n(cardiacsurgery, size=5000, replace=TRUE)
df1 <- subset(cardiacsurgery, select=c(Parsonnet, status))
df2 <- subset(s5000, select=c(Parsonnet, status))
## estimate coefficients from logit model
coeff1 <- round(coef(glm(status~Parsonnet, data=df1, family="binomial")), 3)
coeff2 <- round(coef(glm(status~Parsonnet, data=df2, family="binomial")), 3)
## Number of simulation runs
m <- 10^3
## Number of cores
nc <- parallel::detectCores()
# steady state
RNGkind("L'Ecuyer-CMRG")
m <- 10^3
tau <- 50
kopt <- optimal_k(QA=2, parsonnetscores=df1$Parsonnet, coeff=coeff1)
# 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_adoc_sim, k=kopt, QS=2, h= 2.637854, df=df1,
                                m=i, coeff=coeff1, coeff2=coeff2, side="low", mc.cores=nc))
  list(data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m))))
} )
RES <- data.frame(cbind(M=0:(tau-1), do.call(rbind, res)))
ggplot2::qplot(x=M, y=ARL, data=RES, geom=c("line", "point")) +
ggplot2::theme_classic()

## End(Not run)
```

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

Description

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

Usage

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

Arguments

L0	double. Prespecified in-control Average Run Length.
k	double. Reference value of the CUSUM control chart.
df	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
coeff2	NumericVector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
m	integer. Number of simulation runs.
QS	double. Defines the performance of a surgeon with the odds ratio ratio of death Q.
side	character. Default is "low" to calculate ARL for the upper arm of the V-mask. If side = "up", calculate the lower arm of the V-mask.
nc	integer. Number of cores.
verbose	boolean. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

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

Value

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

Author(s)

Philipp Wittenberg

Examples

```
## Not run:
library("vlad")
library("spcadjust")
## Datasets
data("cardiacsurgery")
s5000 <- dplyr::sample_n(cardiacsurgery, size=5000, replace=TRUE)
df1 <- subset(cardiacsurgery, select=c(Parsonnet, status))
df2 <- subset(s5000, select=c(Parsonnet, status))
## estimate coefficients from logit model
coeff1 <- round(coef(glm(status~Parsonnet, data=df1, family="binomial")), 3)
coeff2 <- round(coef(glm(status~Parsonnet, data=df2, family="binomial")), 3)
## Number of simulation runs
m <- 10^3
## Number of cores
nc <- parallel::detectCores()
## Lower cusum (detecting deterioration)
## k = 0
eocusum_arloc_h_sim(L0=370, df=df1, k=0, m=m, side="low", coeff=coeff1, coeff2=coeff2, nc=nc)
## k = kopt
QA <- 2
# use package function optimal_k to determine k
kopt <- optimal_k(QA=QA, parsonnetscores=df1$Parsonnet, coeff=coeff1)
eocusum_arloc_h_sim(L0=370, df=df1, k=kopt, m=m, side="low", coeff=coeff1, coeff2=coeff2, nc=nc)
## Upper cusum (detecting improvement)
## k = 0
eocusum_arloc_h_sim(L0=370, df=df1, k=0, m=m, side="up", coeff=coeff1, coeff2=coeff2, nc=nc)
## k = kopt
QA <- 1/2
kopt <- optimal_k(QA=QA, parsonnetscores=df1$Parsonnet, coeff=coeff1)
eocusum_arloc_h_sim(L0=370, df=df1, k=kopt, m=m, side="up", coeff=coeff1, coeff2=coeff2, nc=nc)

## End(Not run)
```

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

Description

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

Usage

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

Arguments

`r`	int. Number of of simulation runs.

k	double. Reference value of the CUSUM control chart.
h	double. Decision interval (alarm limit, threshold) of the CUSUM control chart.
df	DataFrame. First column Parsonnet Score and second column outcome of each operation.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
coeff2	NumericVector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
QS	double. Defines the performance of a surgeon with the odds ratio ratio of death Q.
side	character. Default is "low" to calculate ARL for the upper arm of the V-mask. If side = "up", calculate the lower arm of the V-mask.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

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

Examples

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

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

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

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

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

L0	double. Prespecified in-control Average Run Length.
k	double. Reference value of the CUSUM control chart.
df	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model. For more information see details.
m	integer. Number of simulation runs.
yemp	boolean. Use emirical outcome value.
side	character. Default is "low" to calculate ARL for the upper arm of the V-mask. If side = "up", calculate the lower arm of the V-mask.
nc	integer. Number of cores.
verbose	boolean. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

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

Value

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

Author(s)

Philipp Wittenberg

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. doi: 10.2307/2985287.

Examples

```
## Not run:
library("vlad")
library("spcadjust")
set.seed(1234)
data("cardiacsurgery")
df1 <- subset(cardiacsurgery, select=c(Parsonnet, status))
## estimate coefficients from logit model
coeff1 <- round(coef(glm(status~Parsonnet, data=df1, family="binomial")), 3)
## Number of simulation runs
m <- 10^3
## Number of cores
nc <- parallel::detectCores()
## Detect deterioration
QA <- 2
kopt <- optimal_k(QA=QA, parsonnetscores=df1$Parsonnet, coeff=coeff1)
h <- eocusum_arl_h_sim(L0=370, df=df1, k=kopt, m=m, coeff=coeff1, side="low", nc=nc)
## V-Mask parameters
d <- h/kopt
theta <- atan(kopt)*180/pi
cbind(kopt, h, theta, d)

## 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, k, h, df, coeff, yemp = TRUE, side = "low")
```

Arguments

`r`	int. Number of of simulation runs.
`k`	double. Reference value of the CUSUM control chart.
`h`	double. Decision interval (alarm limit, threshold) of the CUSUM control chart.
`df`	DataFrame. First column Parsonnet Score and second column outcome of each operation.
`coeff`	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
`yemp`	boolean. 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

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

Examples

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

## Serial simulation
RNGkind("L'Ecuyer-CMRG")
m <- 10^3
kopt <- optimal_k(QA=2, parsonnetscores=df1$Parsonnet, coeff=coeff1)
#eocusum_arloc_h_sim(L0=370, df=df1, k=kopt, m=m, side="low", coeff=coeff1, coeff2=coeff2, nc=nc)
RLS <- do.call(c, lapply(1:m, eocusum_arloc_sim, h=2.626, k=kopt, df=df1, side="low", coeff=coeff1,
                         coeff2=coeff2))
data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m)))
## Parallel simulation (FORK)
RNGkind("L'Ecuyer-CMRG")
m <- 10^3
kopt <- optimal_k(QA=2, parsonnetscores=df1$Parsonnet, coeff=coeff1)
RLS <- simplify2array(parallel::mclapply(1:m, eocusum_arloc_sim, h=2.626, 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)
RNGkind("L'Ecuyer-CMRG")
no_cores <- parallel::detectCores()
cl <- parallel::makeCluster(no_cores)
side <- "low"
h_vec <- 2.626
QS_vec <- 1
m <- 10^3
```

```
k <- optimal_k(QA=2, parsonnetscores=df1$Parsonnet, coeff=coeff1)
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)
```

`gettherisk` *Compute Risk of death*

Description

Compute Risk of death.

Usage

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

Arguments

`parsonnetscore`	int. Parsonnet Score.
`coeff`	NumericVector. Estimated coefficients α and β from the binary logistic regression model.

Value

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

Author(s)

Philipp Wittenberg

References

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

Steiner S (2014). "Risk-Adjusted Monitoring of Outcomes in Health Care." In Lawless JF (ed.), *Statistics in Action*, pp. 225-242. Informa UK Limited. doi: 10.1201/b16597-15.

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.

Examples

```
library("vlad")
# see Steiner et al. 2000 page 445 or Steiner (2014) p. 234
coeff <- c("(Intercept)"=-3.68, "Parsonnet"=0.077)
# low risk patient (Parsonnet score=0) has a risk of death 2.5%
gettherisk(0L, coeff=coeff)
# high risk patient (Parsonnet score=71) has a risk of death 86%
gettherisk(71L, coeff=coeff)
# high risk patient (Parsonnet score=50) has a risk of death 54%
gettherisk(50L, coeff=coeff)

# Get mortality and probability of death of a phase I dataset
library("spcadjust")
data("cardiacsurgery")
cardiacsurgery <- dplyr::mutate(cardiacsurgery, phase=factor(ifelse(date < 2*365, "I", "II")))
SI <- subset(cardiacsurgery, c(phase=="I"), c("Parsonnet", "status"))
GLM1 <- glm(status ~ Parsonnet, data=SI, family="binomial")
coeff1 <- coef(GLM1)
mprob <- as.numeric(table(SI$Parsonnet) / length(SI$Parsonnet))

# Use estimated model coefficients and parsonnet scores in gettherisk function
# or predicted values from a GLM
s <- sort(unique(SI$Parsonnet))
mort <- sapply(s, gettherisk, coeff=coeff1)
mort1 <- predict(GLM1, newdata=data.frame(Parsonnet=s), type="response")
all.equal(as.numeric(mort), as.numeric(mort1))
df1 <- data.frame(s, mprob, mort)
## Not run:
# Plot mortality and probability to die of phase I data
ggplot2::qplot(data=df1, s, mprob) + ggplot2::theme_classic()
library(ggplot2)
xx <- tapply(SI$status, SI$Parsonnet, sum)
nn <- tapply(SI$status, SI$Parsonnet, length)
ll <- binom::binom.confint(xx, nn, conf.level=0.99, methods="exact")$lower
uu <- binom::binom.confint(xx, nn, conf.level=0.99, methods="exact")$upper
ybar <- tapply(SI$status, SI$Parsonnet, mean)
ggplot(data=df1, aes(s, mort)) +
 geom_point(data=data.frame(s, ybar), aes(s, ybar), inherit.aes=FALSE) +
 geom_errorbar(aes(ymax=uu, ymin=ll), width=0.9, position="dodge", alpha=0.3) +
 geom_line(colour="red") + labs(x="Parsonnet score", y="Probability to die") +
 theme_classic()
```

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

`llr_score` *Compute the log-likelihood ratio score*

Description

Compute the log-likelihood ratio score.

Usage

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

Arguments

`df`	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
`coeff`	NumericVector. 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`	boolean. 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. doi: 10.1093/biostatistics/1.4.441.

Steiner S (2014). "Risk-Adjusted Monitoring of Outcomes in Health Care." In Lawless JF (ed.), *Statistics in Action*, pp. 225-242. Informa UK Limited. doi: 10.1201/b16597-15.

Examples

```
require(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)
```

---

optimal_k *Compute optimal k*

Description

Compute optimal k.

Usage

```
optimal_k(QA, parsonnetscores, coeff)
```

Arguments

QA	double. Defines the performance of a surgeon with the odds ratio ratio of death Q.
parsonnetscores	NumericVector. Vector of Parsonnet Scores.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model. For more information see details.

Details

Formula deterioration:

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

Formula improvement:

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

Value

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

Author(s)

Philipp Wittenberg

Examples

```
library("vlad"); library("spcadjust")
data("cardiacsurgery")
cardiacsurgery <- dplyr::mutate(cardiacsurgery, phase=factor(ifelse(date < 2*365, "I", "II")))
S2I <- subset(cardiacsurgery, c(surgeon==2 & phase=="I"), c("Parsonnet", "status"))
coeff <- coef(glm(status ~ Parsonnet, data=S2I, family="binomial"))
kopt <- optimal_k(QA=2, parsonnetscores=S2I$Parsonnet, coeff=coeff)
kopt ## (Deterioration)
# manually find optimal k for detecting improvement
QA <- 2
pbar <- mean(sapply(S2I[, 1], gettherisk, coef=coeff))
kopt <- pbar * ( QA - 1 - log(QA) ) / log(QA)
all.equal(kopt, optimal_k(QA=2, parsonnetscores=S2I$Parsonnet, coeff=coeff) )
kopt <- optimal_k(QA=1/2, parsonnetscores=S2I$Parsonnet, coeff=coeff)
kopt ##(Improvement)
## k_opt = 0.02555328
### k = kopt
QA <- 1/2
# manually find optimal k for detecting improvement
pbar <- mean(sapply(S2I[, 1], gettherisk, coef=coeff))
kopt <- pbar * ( 1 - QA + log(QA) ) / log(QA)
all.equal(kopt, optimal_k(QA=1/2, parsonnetscores=S2I$Parsonnet, coeff=coeff) )
```

---

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

Description

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

Usage

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

Arguments

r	Integer vector. Number of runs.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
coeff2	NumericVector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
h	double. Control Chart limit for detecting deterioration/improvement.
df	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
R0	double. Odds ratio of death under the null hypotheses.
RA	double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio RA=2. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death RA=1/2.
RQ	double. Defines the performance of a surgeon with the odds ratio ratio of death Q.
m	Integer. Simulated in-control observations.
type	character. Default argument is "cond" for computation of conditional steady-state. Other option is the cyclical steady-state "cycl".

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

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

Taylor HM (1968). "The Economic Design of Cumulative Sum Control Charts." *Technometrics*, **10**(3), pp. 479-488. doi: 10.1080/00401706.1968.10490595.

Crosier R (1986). "A new two-sided cumulative quality control scheme." *Technometrics*, **28**(3), pp. 187-194. doi: 10.2307/1269074.

Examples

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

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

# simulation of conditional steady state
m <- 10^3
tau <- 50
res <- sapply(0:(tau-1), function(i){
 RLS <- do.call(c, parallel::mclapply( 1:m, racusum_adoc_sim, RQ=2, h=2.0353, df=df1, m=i,
                                       coeff=coeff1, coeff2=coeff1,
                                       mc.cores=parallel::detectCores()) )
 list(data.frame(cbind(ARL=mean(RLS), ARLSE=sd(RLS)/sqrt(m))))
} )

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

`L0`	double. Prespecified in-control Average Run Length.
`df`	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
`coeff`	NumericVector. Estimated coefficients α and β from the binary logistic regression model.

coeff2	NumericVector. Estimated coefficients α and β from the binary logistic regression model of a resampled dataset.
R0	double. Odds ratio of death under the null hypotheses.
RA	double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio RA=2. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death RA=1/2.
RQ	double. Defines the performance of a surgeon with the odds ratio ratio of death Q.
m	integer. Number of simulation runs.
nc	integer. Number of cores.
verbose	boolean. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

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

Value

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

Author(s)

Philipp Wittenberg

Examples

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

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

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

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

# 2. Determine ARL and Standard Error
RLS <- do.call(c, parallel::mclapply(1:m, racusum_arloc_sim, h=2.035, df=df1, RA=2, coeff=coeff1,
                                     coeff2=coeff2, mc.cores=6))
data.frame(cbind("ARL"=mean(RLS), "ARLSE"=sd(RLS)/sqrt(m)))
# ARL=371.125; ARLSE=11.36053

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

RA — double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio RA=2. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death RA=1/2.

RQ — double. Defines the performance of a surgeon with the odds ratio ratio of death Q.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

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

Examples

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

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

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

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

```
df3 <- data.frame(cbind(RQ, "ARL"=apply(RLS, 2, mean), "ARLSE"=apply(RLS, 2, mean)/sqrt(m) ))
ggplot(df3, aes(RQ, ARL)) + geom_line() + theme_classic()

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

`L0`	double. Prespecified in-control Average Run Length.
`df`	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
`coeff`	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
`R0`	double. Odds ratio of death under the null hypotheses.
`RA`	double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio RA=2. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death RA=1/2.
`m`	integer. Number of simulation runs.
`yemp`	boolean. If TRUE, use emirical outcome values, else use model.

nc	integer. Number of cores used for parallel processing.
verbose	boolean. If TRUE verbose output is included, if FALSE a quiet calculation of h is done.

Details

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

Value

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

Author(s)

Philipp Wittenberg

Examples

```
## Not run:
library("vlad")
library("spcadjust")
data("cardiacsurgery")
cardiacsurgery <- dplyr::mutate(cardiacsurgery, phase=factor(ifelse(date < 2*365, "I", "II")))
S2 <- subset(cardiacsurgery, c(surgeon==2), c("phase", "Parsonnet", "status"))
# subset phase I (In-control) of surgeons 2
S2I <- subset(S2, c(phase=="I"), c("Parsonnet", "status"))
# estimate coefficients from logit model
coeff1 <- round(coef(glm(status~Parsonnet, data=S2I, family="binomial")), 3)

racusum_arl_h_sim(L0=740, df=S2I, coeff=coeff1, m=10^2, nc=4)

## End(Not run)
```

---

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

Description

Compute ARLs of RA-CUSUM control charts using simulation.

Usage

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

Arguments

r	Integer vector. Number of runs.
coeff	NumericVector. Estimated coefficients α and β from the binary logistic regression model.
h	double. Control Chart limit for detecting deterioration/improvement.
df	DataFrame. First column are Parsonnet Score values within a range of zero to 100 representing the preoperative patient risk. The second column are binary (0/1) outcome values of each operation.
R0	double. Odds ratio of death under the null hypotheses.
RA	double. Odds ratio of death under the alternative hypotheses. Detecting deterioration in performance with increased mortality risk by doubling the odds Ratio RA=2. Detecting improvement in performance with decreased mortality risk by halving the odds ratio of death RA=1/2.
yemp	boolean. If TRUE use observed outcome value, if FALSE use estimated binary logistc regression model.

Value

Returns a single value which is the Run Length.

Author(s)

Philipp Wittenberg

References

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

Examples

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

## Parallel Simulation 1: y = random (10^4 runs, RA=2)
m <- 10^4; h_vec <- 2.7; yemp <- FALSE
no_cores <- parallel::detectCores()
cl <- parallel::makeCluster(no_cores)
parallel::clusterExport(cl, c("h_vec", "racusum_arl_sim", "coeff1", "df1", "yemp"))
time <- system.time( {
  ARL <- array(NA, dim=c( length(h_vec), m))
  for (h in h_vec) {
```

```
   ARL[which(h_vec==h), ] <- parallel::parSapply(cl, 1:m, racusum_arl_sim, h=h, coeff=coeff1,
                                                 df=df1, yemp=yemp, USE.NAMES=FALSE) }
} )
simMean <- apply(ARL, c(1), mean)
simSE <- sqrt(apply(ARL, c(1), var)/m)
print(list(simMean, simSE, time))
parallel::stopCluster(cl)
df.sim1 <- data.frame("RA"=2, "h"=h, "ARL"=simMean, "ARLSE"=simSE, "nsim"=m)

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

rbind(df.sim1, df.sim2)

## End(Not run)
```

Index