# Package 'bcROCsurface'

July 22, 2025

**Version** 1.0-6

**Date** 2023-09-09

**Title** Bias-Corrected Methods for Estimating the ROC Surface of Continuous Diagnostic Tests

**Author** Duc-Khanh To, with contributions from Monica Chiogna and Gianfranco Adimari

**Maintainer** Duc-Khanh To <toduc@stat.unipd.it>

**Description** The bias-corrected estimation methods for the receiver operating characteristics ROC surface and the volume under ROC surfaces (VUS) under missing at random (MAR) assumption.

**License** GPL-3

**Depends** R(>= 3.5.0), nnet, rgl, boot, stats, utils, graphics, parallel

**Imports** Rcpp (>= 0.12.2)

**LinkingTo** Rcpp, RcppArmadillo

**LazyLoad** yes

**LazyData** TRUE

**ByteCompile** yes

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Suggests** knitr, rmarkdown, markdown

**URL** https://github.com/toduckhanh/bcROCsurface

**BugReports** https://github.com/toduckhanh/bcROCsurface/issues

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2023-09-09 08:10:10 UTC

# Contents



---

`bcROCsurface-package` *Bias-Corrected Methods for Estimating the ROC Surface of Continuous Diagnostic Tests*

---

## Description

This package provides tools for correcting verification bias in the evaluation of a continuous diagnostic test. More precisely, five bias-corrected methods for ROC surface and VUS inference are provided under MAR assumption, i.e., full imputation (FI), mean score imputation (MSI), inverse probability weighting (IPW), semiparametric efficient (SPE) and K nearest-neighbor (KNN) estimator.

## Details

| | |
|---|---|
| Package: | bcROCsurface |
| Type: | Package |
| Version: | 1.0-6 |
| Date: | 2023-09-09 |
| License: | GPL 2 \| GPL 3 |
| Lazy load: | yes |

Major functions are `rocs` and `vus_mar`.

## Author(s)

Duc-Khanh To, with contributions from Monica Chiogna and Gianfranco Adimari

Maintainer: Duc-Khanh To <toduc@stat.unipd.it>

## References

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*. **18**, 5, 697–720.

To Duc, K., Chiogna, M. and Adimari, G. (2016) Bias-corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, **10**, 3063-3113.

---

`asy_cov_tcf` *Asymptotic variance-covariance estimation for True Class Fractions (TCFs) at the cut point* $(c\_1, c\_2)$

---

## Description

`asy_cov_tcf` computes the asymptotic variance-covariance matrix of full data (FULL) and bias-corrected estimators (i.e. full imputation, mean score imputation, inverse probability weighting, semiparametric efficient and K nearest neighbor) of TCFs.

## Usage

```
asy_cov_tcf(
  obj_tcf,
  diag_test,
  dise_vec,
  veri_stat = NULL,
  rho_est = NULL,
  pi_est = NULL,
  boot = FALSE,
  n_boot = 250,
  parallel = FALSE,
  ncpus = ifelse(parallel, detectCores()/2, NULL)
)
```

## Arguments

| | |
|---|---|
| `obj_tcf` | a result of a call to `rocs.tcf`. |
| `diag_test` | a numeric vector containing the diagnostic test values. `NA` values of `diag_test` are not accepted. |
| `dise_vec` | a n * 3 binary matrix with three columns, corresponding to the three classes of the disease status. In row i, 1 in column j indicates that the i-th subject belongs to class j, with j = 1, 2, 3. A row of `NA` values indicates a non-verified subject. |
| `veri_stat` | a binary vector containing the verification status (1 verified, 0 not verified). |
| `rho_est` | a result of a call to `rho_mlogit` of `rho_knn` to fit the disease model. |
| `pi_est` | a result of a call to `psglm` to fit the verification model. |

| | |
|---|---|
| boot | a logical value. Default = FALSE. If set to TRUE, a bootstrap resampling is employed to estimate the asymptotic variance-covariance matrix of bias-corrected TCFs. |
| n_boot | the number of bootstrap replicates, used when boot = TRUE or for FULL estimator. Usually this will be a single positive integer. Default 250. |
| parallel | a logical value. If TRUE, a parallel computing is employed in the bootstrap resampling process. |
| ncpus | number of processes to be used in parallel computing. Default is half of available cores. |

## Details

For bias-corrected estimators of TCFs, the asymptotic variance-covariance matrix at a fixed cut point is estimated by using the Delta method. The function asy_cov_tcf implements the explicit forms presented in To Duc et al. (2016, 2020). In addition, the bootstrap procedure is also available.

For FULL estimator, the asymptotic variance-covariance matrix is computed via bootstrap only.

## Value

This function returns an estimated asymptotic variance-covariance matrix for FULL estimator and bias-corrected estimators of TCFs at a fixed cut point.

## References

To Duc, K., Chiogna, M. and Adimari, G. (2016) Bias-corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, **10**, 3063-3113.

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*. **18**, 5, 697–720.

## Examples

```
data(EOC)

# FULL data estimator
dise_full <- pre_data(EOC$D.full, EOC$CA125)
dise_vec_full <- dise_full$dise_vec

full_tcf <- rocs.tcf("full", diag_test = EOC$CA125, dise_vec = dise_vec_full,
                     cps = c(2, 4))
full_var <- asy_cov_tcf(full_tcf, diag_test = EOC$CA125,
                        dise_vec = dise_vec_full)

# Preparing the missing disease status
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_vec_na <- dise_na$dise_vec
dise_fact_na <- dise_na$dise

rho_out <- rho_mlogit(dise_fact_na ~ CA125 + CA153 + Age, data = EOC,
```

```
                      test = TRUE)

## FI estimator
fi_tcf <- rocs.tcf("fi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
                   veri_stat = EOC$V, rho_est = rho_out, cps = c(2, 4))
fi_var <- asy_cov_tcf(fi_tcf, diag_test = EOC$CA125, dise_vec = dise_vec_na,
                      veri_stat = EOC$V, rho_est = rho_out)

## MSI estimator
msi_tcf <- rocs.tcf("msi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
                    veri_stat = EOC$V, rho_est = rho_out, cps = c(2, 4))
msi_var <- asy_cov_tcf(msi_tcf, diag_test = EOC$CA125,
                       dise_vec = dise_vec_na, veri_stat = EOC$V,
                       rho_est = rho_out)

## IPW estimator
pi_out <- psglm(V ~ CA125 + CA153 + Age, data = EOC, test = TRUE)

ipw_tcf <- rocs.tcf("ipw", diag_test = EOC$CA125, dise_vec = dise_vec_na,
                    veri_stat = EOC$V, pi_est = pi_out, cps = c(2, 4))
ipw_var <- asy_cov_tcf(ipw_tcf, diag_test = EOC$CA125,
                       dise_vec = dise_vec_na, veri_stat = EOC$V,
                       pi_est = pi_out)

## SPE estimator
spe_tcf <- rocs.tcf("spe", diag_test = EOC$CA125, dise_vec = dise_vec_na,
                    veri_stat = EOC$V, rho_est = rho_out, pi_est = pi_out,
                    cps = c(2, 4))
spe_var <- asy_cov_tcf(spe_tcf, diag_test = EOC$CA125,
                       dise_vec = dise_vec_na, veri_stat = EOC$V,
                       rho_est = rho_out, pi_est = pi_out)

## KNN estimators
x_mat <- cbind(EOC$CA125, EOC$CA153, EOC$Age)
rho_1nn <- rho_knn(x_mat = x_mat, dise_vec = dise_vec_na, veri_stat = EOC$V,
                   k = 1, type = "mahala")
knn_tcf <- rocs.tcf("knn", diag_test = EOC$CA125, dise_vec = dise_vec_na,
                    veri_stat = EOC$V, rho_est = rho_1nn, cps = c(2, 4))
knn_var <- asy_cov_tcf(knn_tcf, diag_test = EOC$CA125,
                       dise_vec = dise_vec_na, veri_stat = EOC$V,
                       rho_est = rho_1nn)
```

---

`asy_var_vus` *Asymptotic variance estimation for VUS*

---

## Description

`asy_var_vus` computes the asymptotic variance of full data (FULL) and bias-corrected estimators

(i.e. full imputation, mean score imputation, inverse probability weighting, semiparametric efficient and K nearest neighbor) of VUS.

## Usage

```
asy_var_vus(
  obj_vus,
  diag_test,
  dise_vec,
  veri_stat = NULL,
  rho_est = NULL,
  pi_est = NULL,
  boot = FALSE,
  n_boot = 250,
  parallel = FALSE,
  ncpus = ifelse(parallel, detectCores()/2, NULL)
)
```

## Arguments

| | |
|---|---|
| obj_vus | a result of a call to vus_mar. |
| diag_test | a numeric vector containing the diagnostic test values. NA values of diag_test are not accepted. |
| dise_vec | a n * 3 binary matrix with three columns, corresponding to the three classes of the disease status. In row i, 1 in column j indicates that the i-th subject belongs to class j, with j = 1, 2, 3. A row of NA values indicates a non-verified subject. |
| veri_stat | a binary vector containing the verification status (1 verified, 0 not verified). |
| rho_est | a result of a call to rho_mlogit of rho_knn to fit the disease model. |
| pi_est | a result of a call to psglm to fit the verification model. |
| boot | a logical value. Default = FALSE. If set to TRUE, a bootstrap resampling is employed to estimate the asymptotic variance of the bias-corrected VUS estimators. |
| n_boot | the number of bootstrap replicates, which is used for FULL or KNN estimators, or option boot = TRUE. The defaut is 250. |
| parallel | a logical value. If TRUE, a parallel computing is employed in the bootstrap resampling process. |
| ncpus | number of processes to be used in parallel computing. Default is half of available cores. |

## Details

For the FULL estimator, a bootstrap resampling process or Jackknife approach is used to estimate the asymptotic variance, whereas, a bootstrap resampling process is employed to obtain the asymptotic variance of K nearest neighbor estimator.

For the full imputation, mean score imputation, inverse probability weighting and semiparametric efficient estimators of VUS, the asymptotic variances are computed by using the explicit form. Furthermore, a bootstrap procedure is also available, useful in case of small sample sizes.

## Value

asy_var_vus returns a estimated value of the asymptotic variance.

## References

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*. **18**, 5, 697–720.

To Duc, K., Chiogna, M. and Adimari, G. (2016) Bias-corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, **10**, 3063-3113.

Guangming, P., Xiping, W. and Wang, Z. (2013) Non-parameteric statistical inference for $P(X < Y < Z)$. *Sankhya A*, **75**, 1, 118-138.

## Examples

```
data(EOC)

# Preparing the missing disease status
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_vec_na <- dise_na$dise_vec
dise_fact_na <- dise_na$dise

rho_out <- rho_mlogit(dise_fact_na ~ CA125 + CA153 + Age, data = EOC,
                      test = TRUE)
vus_fi <- vus_mar("fi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
                  veri_stat = EOC$V, rho_est = rho_out, ci = FALSE)
var_fi <- asy_var_vus(vus_fi, diag_test = EOC$CA125, dise_vec = dise_vec_na,
                      veri_stat = EOC$V, rho_est = rho_out)
```

---

# cv_knn *Cross-validation for K nearest-neighbor regression*

---

## Description

This function calculates the estimated cross-validation prediction error for K nearest-neighbor regression and returns a suitable choice for K.

## Usage

```
cv_knn(x_mat, dise_vec, veri_stat, k_list = NULL, type = "eucli", plot = FALSE)
```

## Arguments

| | |
|---|---|
| x_mat | a numeric design matrix, which used in rho_knn to estimate probabilities of the disease status. |
| dise_vec | a n * 3 binary matrix with three columns, corresponding to the three classes of the disease status. In row i, 1 in column j indicates that the i-th subject belongs to class j, with j = 1, 2, 3. A row of NA values indicates a non-verified subject. |
| veri_stat | a binary vector containing the verification status (1 verified, 0 not verified). |
| k_list | a list of candidate values for K. If NULL(the default), the set $\{1, 2, ..., n.ver\}$ is employed, where, $n.ver$ is the number of verified subjects. |
| type | a type of distance, see rho_knn for more details. Default "eucli". |
| plot | if TRUE, a plot of cross-validation prediction error is produced. |

## Details

Data are divided into two groups, the first contains the data corresponding to veri_stat = 1, whereas the second contains the data corresponding to veri_stat = 0. In the first group, the discrepancy between the true disease status and the KNN estimates of the probabilities of the disease status is computed by varying k from 1 to the number of verification subjects, see To Duc et al. (2020). The optimal value of k is the value that corresponds to the smallest value of the discrepancy.

## Value

A suitable choice for k is returned.

## References

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*. **18**, 5, 697–720.

## Examples

```
data(EOC)
x_mat <- cbind(EOC$CA125, EOC$CA153, EOC$Age)
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_vec_na <- dise_na$dise_vec
cv_knn(x_mat, dise_vec_na, EOC$V, type = "mahala", plot = TRUE)
```

---

# EOC *A sub-data from Pre-PLCO Phase II Dataset*

---

## Description

A subset of the Pre-PLCO Phase II Dataset from the SPORE/Early Detection Network/Prostate, Lung, Colon, and Ovarian Cancer Ovarian Validation Study. This data deals with epithelial ovarian cancer (EOC).

## Usage

```
EOC
```

## Format

A data frame with 278 observations on the following 6 variables.

`D.full` a factor with 3 levels of disease status, 1, 2, 3. The levels correspond to benign disease, early stage (I and II) and late stage (III and IV).

`V` a binary vector containing the verification status. 1 or 0 indicates verified or non verified subject.

`D` a copy of `D.full` with the missing values. `NA` values correspond to non verified subjects.

`CA125` a numeric vector of biomarker CA125 (used as diagnostic test).

`CA153` a numeric vector of biomarker CA153 (used as covariate).

`Age` a numeric vector containing the age of patients.

## Details

The Pre-PLCO datasets contain some demographic variables (Age, Race, ect.) and 59 markers measured by 4 sites (Harvard, FHCRC, MD Anderson, and Pittsburgh). Some interest biomarkers are: CA125, CA153, CA19–9, CA72–4, Kallikrein 6 (KLK6), HE4 and Chitinase (YKL40). The original data set consist of control groups and three classes of EOC: benign disease, early stage (I and II) and late stage (III and IV). In the sub data set, the biomakers CA125 and CA153 (measured at Harvard laboratories), the age of patients, and three classes of EOC are collected. In addition, the verification status and the missing disease status are also added.

The verification status $V$ is generated by using the following selection process:

$$P(V = 1) = 0.05 + 0.35I(CA125 > 0.87) + 0.25I(CA153 > 0.3) + 0.35I(Age > 45).$$

This process leads to 63.4% patients selected to undergo disease verification.

The missing disease status D are the copies of the full disease status D.full, but some values corresponding to $V = 0$ are deleted (refered as `NA` values).

## Source

SPORE/EDRN/PRE-PLCO Ovarian Phase II Validation Study: https://edrn-labcas.jpl.nasa.gov/labcas-ui/c/index.html?collection_id=Pre-PLCO_Phase_II_Dataset.

---

`pre_data` *Preparing monotone ordered disease classes*

---

## Description

`pre_data` is used to check and make a suitable monotone increasing ordering of the disease classes. In addition, this function also creates a binary matrix format of disease status to pass into the functions of `bcROCsurface` package.

## Usage

```
pre_data(dise, diag_test, plot = TRUE)
```

## Arguments

| | |
|---|---|
| dise | a numeric vector/factor containing the disease status. |
| diag_test | a numeric vector containing the diagnostic test values. NA values are not admitted. |
| plot | if TRUE (the default) then a boxplot of diagnostic test based on three ordered groups is produced. |

## Details

The ROC surface analysis implemented in the package is coherent when the ordering of the diagnostic classes is monotone increasing. That is, for a diagnostic test $T$ and three disease classes, 1, 2 and 3, the monotone increasing ordering of interest is $T_1 < T_2 < T_3$. Here, $T_1$, $T_2$ and $T_3$ are the measurements of diagnostic test $T$ corresponding to class 1, 2 and 3, respectively. Note that, if an umbrella or tree ordering is of interest, then the results of ROC surface analysis is not reliable.

In order to find out the monotone ordering, we compute the medians of $T_1$, $T_2$ and $T_3$, and then sort the three medians in ascending order. After that, the three disease classes are reordered corresponding to the order of medians.

To be used in the functions of package bcROCsurface, the vector of disease status must be presented as a n * 3 binary matrix with the three columns, corresponding to the three classes.

With real data, the application of this function is the first step in the use of ROC analysis. Note that, if the user is sure that the disease classes follow a monotone increasing ordering and the disease matrix is available, then the use of pre_data is not necessary.

## Value

This function returns a list containting a factor dise of ordered disease status and a binary matrix dise_vec of the disease status and a vector order containing the sequence of class labels.

## Examples

```
data(EOC)
dise_full <- pre_data(EOC$D.full, EOC$CA125)
```

## print.vus_mar *Print summary results of VUS*

### Description

print.vus_mar prints the results for the output of function vus_mar.

### Usage

```
## S3 method for class 'vus_mar'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

### Arguments

| | |
|---|---|
| x | an object of class "vus_mar", a result of a call to vus_mar. |
| digits | minimal number of significant digits, see print.default. |
| ... | further arguments passed to print method. |

### Details

print.vus_mar shows a nice format of the summary table for the VUS estimate results. Some information on the diagnostic test, the fitted values of VUS, and confidence intervals are shown.

### See Also

vus_mar

## psglm *Fitting verification models*

### Description

psglm is used to fit generalized linear models to the verification process. This function requires a symbolic formula of the linear predictor, and a specified regression model.

### Usage

```
psglm(formula, data, model = "logit", test = FALSE, trace = TRUE, ...)
```

## Arguments

| | |
|---|---|
| formula | an object of class "formula": a symbolic description of the model to be fitted. |
| data | an optional data frame containing the variables in the model. |
| model | a specified model to be used in the fitting. The suggestion regression models are logit, probit and threshold. If model is ignored, then psglm use a default model as logit. |
| test | a logical value indicating whether p-values of the regression coefficients should be returned. |
| trace | switch for tracing estimation process. Default TRUE. |
| ... | optional arguments to be passed to glm. |

## Details

psglm estimates the verification probabilities of the patients. The suggestion model is designed as a list containing: logit, probit and threshold.

## Value

psglm returns a list containing the following components:

| | |
|---|---|
| coeff | a vector of estimated coefficients. |
| values | fitted values of the model. |
| Hess | the Hessian of the measure of fit at the estimated coefficients. |
| x | a design model matrix. |
| formula | the formula supplied. |
| model | the model object used. |

## See Also

glm

## Examples

```
data(EOC)
out <- psglm(V ~ CA125 + CA153 + Age, data = EOC, test = TRUE)
```

rho_knn *K nearest-neighbor (KNN) regression*

## Description

rho_knn uses the KNN approach to estimate the probabilities of the disease status in case of three categories.

## Usage

```
rho_knn(
  x_mat,
  dise_vec,
  veri_stat,
  k,
  type = c("eucli", "manha", "canber", "lagran", "mahala"),
  trace = FALSE
)
```

## Arguments

| | |
|---|---|
| x_mat | a numeric design matrix. |
| dise_vec | a n * 3 binary matrix with three columns, corresponding to the three classes of the disease status. In row i, 1 in column j indicates that the i-th subject belongs to class j, with j = 1, 2, 3. A row of NA values indicates a non-verified subject. |
| veri_stat | a binary vector containing the verification status (1 verified, 0 not verified). |
| k | an integer value/vector, which indicates the number of nearest neighbors. It should be less than the number of the verification subjects. |
| type | a distance measure. |
| trace | switch for tracing estimation process. Default FALSE. |

## Details

type should be selected as one of "eucli", "manha", "canber", "lagran", "mahala" corresponding to Euclidean, Manhattan, Canberra, Lagrange and Mahalanobis distance. In practice, the selection of a suitable distance is typically dictated by features of the data and possible subjective evaluations. For example, if the covariates are heterogeneous with respect to their variances (which is particularly true when the variables are measured on heterogeneous scales), the choice of the Mahalanobis distance may be a good choice.

For the number of nearest neighbors, a small value of k, within the range 1-3, may be a good choice. In general, the choice of k may depend on the dimension of the feature space, and propose to use cross–validation to find k in case of high–dimensional covariate. See cv_knn.

## Value

rho_knn returns a list containing the following components:

| | |
|---|---|
| values | estimates of the probabilities. |
| X | a design model matrix. |
| K | the number of nearest neighbors. |
| type | the chosen distance. |

## References

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*. **18**, 5, 697–720.

## Examples

```
data(EOC)
x_mat <- cbind(EOC$CA125, EOC$CA153, EOC$Age)
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_vec_na <- dise_na$dise_vec

## Euclidean distance, k = 1
out_ecul_1nn <- rho_knn(x_mat, dise_vec_na, EOC$V, k = 1, type = "eucli")

## Manhattan distance, k = 1
out_manh_1nn <- rho_knn(x_mat, dise_vec_na, EOC$V, k = 1, type = "manha")

## Canberra distance, k = 3
out_canb_1nn <- rho_knn(x_mat, dise_vec_na, EOC$V, k = 3, type = "canber")

## Lagrange distance, k = 3
out_lagr_1nn <- rho_knn(x_mat, dise_vec_na, EOC$V, k = 3, type = "lagran")

## Mahalanobis distance, k = c(1,3)
out_maha_13nn <- rho_knn(x_mat, dise_vec_na, EOC$V, k = c(1, 3),
                         type = "mahala")
```

---

# rho_mlogit *Fitting disease models via multinomial logistic models*

---

## Description

rho_mlogit is used to fit multinomial logistic models to the disease process in the verified subjects.

## Usage

```
rho_mlogit(formula, data, test = FALSE, maxit = 500, trace = FALSE)
```

## Arguments

| | |
|---|---|
| formula | an object of class "formula": a symbolic description of the model to be fitted. |
| data | an optional data frame containing the variables in the model. |
| test | a logical value indicating whether p-values of the regression coefficients should be returned. Default FALSE. |
| maxit | maximum number of iterations. Default 500. |
| trace | switch for tracing estimation process. Default FALSE. |

## Details

In the formula, the response must be a result of pre_data, a factor with three levels, say 1, 2, 3. These levels correspond to three classes of disease status, e.g., non-dieseased, intermediate, diseased. The last class (class 3) is considered as the reference level in multinomal logistic model. In presence of verification bias, the missing (NA) values correspond to non verified subjects.

## Value

rho_mlogit returns a list containing the following components:

| | |
|---|---|
| coeff | a vector of estimated coefficients. |
| values | fitted values of the model. |
| Hess | the Hessian of the measure of fit at the estimated coefficients. |
| D | the disease status vector used. |
| X | a design model matrix. |
| formula | the fomular supplied. |

## References

To Duc, K., Chiogna, M. and Adimari, G. (2016) Bias-corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, **10**, 3063-3113.

## See Also

multinom, nnet

## Examples

```
data(EOC)
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_fact_na <- dise_na$dise
out <- rho_mlogit(dise_fact_na ~ CA125 + CA153 + Age, data = EOC,
                  test = TRUE, trace = TRUE)
```

ROCsurface *Receiver operating characteristics surface for a continuous diagnostic test*

## Description

`rocs.tcf` is used to obtain bias-corrected estimates of the true class fractions (TCFs) for evaluating the accuracy of a continuous diagnostic test for a given cut point $(c_1, c_2)$, with $c_1 < c_2$.

`rocs` provides bias-corrected estimates of the ROC surfaces of the continuous diagnostic test by using TCF.

## Usage

```
rocs.tcf(
  method = "full",
  diag_test,
  dise_vec,
  veri_stat = NULL,
  rho_est = NULL,
  pi_est = NULL,
  cps
)

rocs(
  method = "full",
  diag_test,
  dise_vec,
  veri_stat,
  rho_est = NULL,
  pi_est = NULL,
  ncp = 100,
  plot = TRUE,
  ellipsoid = FALSE,
  cpst = NULL,
  ci_level = 0.95,
  surf_col = c("gray40", "green"),
  boot = FALSE,
  n_boot = 250,
  parallel = FALSE,
  ncpus = ifelse(parallel, detectCores()/2, NULL),
  ...
)
```

## Arguments

| | |
|---|---|
| `method` | a estimation method to be used for estimating the true class fractions in presence of verification bias. See 'Details'. |

| | |
|---|---|
| diag_test | a numeric vector containing the diagnostic test values. NA values are not allowed. |
| dise_vec | a n * 3 binary matrix with the three columns, corresponding to three classes of the disease status. In row i, 1 in column j indicates that the i-th subject belongs to class j, with j = 1, 2, 3. A row of NA values indicates a non-verified subject. |
| veri_stat | a binary vector containing the verification status (1 verified, 0 not verified). |
| rho_est | a result of a call to rho_mlogit of rho_knn to fit the disease model. |
| pi_est | a result of a call to psglm to fit the verification model. |
| cps | a cut point $(c_1, c_2)$, with $c_1 < c_2$, which used to estimate TCFs. If m estimates of TCFs are required, cps must be matrix with m rows and 2 columns. |
| ncp | the dimension of cut point grid. It is used to determine the cut points (see 'Details'). Default 100. |
| plot | if TRUE(the default), a 3D plot of ROC surface is produced. |
| ellipsoid | a logical value. If TRUE, adds an ellipsoidal confidence region for TCFs at a specified cut point to current plot of ROC surface. |
| cpst | a specified cut point, which used to construct the ellipsoid confidence region. If m ellipsoid confidence regions are required, cpst must be matrix with m rows and 2 columns. Default NULL. |
| ci_level | an confidence level to be used for constructing the ellipsoid confidence region; default 0.95. |
| surf_col | color to be used for plotting ROC surface and ellipsoid. |
| boot | a logical value. Default = FALSE. If set to TRUE, a bootstrap resampling is employed to estimate the asymptotic variance-covariance matrix of TCFs at the cut point cpst. See more details in asy_cov_tcf. |
| n_boot | the number of bootstrap replicates, which is used for FULL estimator, or option boot = TRUE. Usually this will be a single positive integer. Default 250. |
| parallel | a logical value. If TRUE, a parallel computing is employed to the bootstrap resampling process. |
| ncpus | number of processes to be used in parallel computing. Default is half of of available cores. |
| ... | optional arguments to be passed to plot3d, surface3d. |

## Details

In a three-class diagnostic problem, quantities used to evaluate the accuracy of a diagnostic test are the true class fractions (TCFs). For a given pair of cut points $(c_1, c_2)$ such that $c_1 < c_2$, subjects are classified into class 1 ($D_1$) if $T < c_1$; class 2 ($D_2$) if $c_1 \leq T < c_2$; class 3 ($D_3$) otherwise. The true class fractions of the test $T$ at $(c_1, c_2)$ are defined as

$$TCF_1(c_1) = P(T < c_1|D_1 = 1) = 1 - P(T \geq c_1|D_1 = 1),$$

$$TCF_2(c_1, c_2) = P(c_1 \leq T < c_2|D_2 = 1) = P(T \geq c_1|D_2 = 1) - P(T \geq c_2|D_2 = 1),$$

$$TCF_3(c_2) = P(T > c_2|D_3 = 1) = P(T \geq c_2|D_3 = 1).$$

The receiver operating characteristic (ROC) surface is the plot of $TCF_1, TCF_2$ and $TCF_3$ by varying the cut point $(c_1, c_2)$ in the domain of the diagnostic test. The cut points $(c_1, c_2)$ are produced

by designing a cut point grid with ncp dimension. In this grid, the points satisfying $c_1 < c_2$ are selected as the cut points. The number of the cut points are obtained as $ncp(ncp-1)/2$, for example, the default is 4950.

These functions implement the bias-corrected estimators in To Duc et al (2016, 2020) for estimating TCF of a three-class continuous diagnostic test in presence of verification bias. The estimators work under MAR assumption. Five methods are provided, namely:

- Full imputation (FI): uses the fitted values of the disease model to replace the true disease status (both of missing and non-missing values).
- Mean score imputation (MSI): replaces only the missing values by the fitted values of the disease model.
- Inverse probability weighted (IPW): weights each observation in the verification sample by the inverse of the sampling fraction (i.e. the probability that the subject was selected for verification).
- Semiparametric efficient (SPE): replaces the true disease status by the double robust estimates.
- K nearest-neighbor (KNN): uses K nearest-neighbor imputation to obtain the missing values of the true disease status.

The argument `method` must be selected from the collection of the bias-corrected methods, i.e., `"full"`, `"fi"`, `"msi"`, `"ipw"`, `"spe"` and `"knn"`.

The ellipsoidal confidence region of TCFs at a given cut point can be constructed by using a normal approximation and plotted in the ROC surface space. The confidence level (default) is 0.95.

Note that, before using the functions `rocs` and `rocs.tcf`, the use of `pre_data` might be needed to check the monotone ordering disease classes and to create the matrix format for disease status.

## Value

`rocs` returns a list, with the following components:

| | |
|---|---|
| `vals` | the estimates of TCFs at all cut points. |
| `cpoint` | the cut points are used to construct the ROC surface. |
| `ncp` | dimension of the cut point grid. |
| `cpst` | the cut points are used to construct the ellipsoidal confidence regions. |
| `tcf` | the estimates of TCFs at the cut points `cpst`. |
| `message` | an integer code or vector. 1 indicates the ellipsoidal confidence region is available. |

`rocs.tcf` returns a vector having estimates of TCFs at a cut point when `cps` is a vector with two elements, or a list of estimates of TCFs at `m` cut points when `cps` is a `m*2` matrix. In addition, some attributes called `theta`, `beta`, `cp` and `name` are given. Here, `theta` is a probability vector, with 3 element, corresponding to the disease prevalence rates of three classes. `beta` is also a probability vector having 4 components, which are used to compute TCFs, see To Duc el al. (2016, 2020) for more details. `cp` is the specified cut point that is used to estimate TCFs. `name` indicates the method used to estimate TCFs. These attributes are required to compute the asymptotic variance-covariance matrix of TCFs at the given cut point.

## References

To Duc, K., Chiogna, M. and Adimari, G. (2016) Bias-corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, **10**, 3063-3113.

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*, **18**, 5, 697–720.

## See Also

psglm, rho_mlogit, plot3d.

## Examples

```
data(EOC)
head(EOC)

## Not run:
# FULL data estimator
dise_full <- pre_data(EOC$D.full, EOC$CA125)
dise_vec_full <- dise_full$dise_vec
if(requireNamespace("webshot2", quietly = TRUE)){
   rocs("full", diag_test = EOC$CA125, dise_vec = dise_vec_full, ncp = 30,
        ellipsoid = TRUE, cpst = c(-0.56, 2.31))
}

## End(Not run)

## Not run:
# Preparing the missing disease status
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_vec_na <- dise_na$dise_vec
dise_fact_na <- dise_na$dise

# FI estimator
rho_out <- rho_mlogit(dise_fact_na ~ CA125 + CA153 + Age, data = EOC,
                      test = TRUE)
if (requireNamespace("webshot2", quietly = TRUE)) {
   rocs("fi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_out, ncp = 30)
}

# Plot ROC surface and add ellipsoid confidence region
if (requireNamespace("webshot2", quietly = TRUE)) {
   rocs("fi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_out, ncp = 30,
        ellipsoid = TRUE, cpst = c(-0.56, 2.31))
}

# MSI estimator
if (requireNamespace("webshot2", quietly = TRUE)) {
   rocs("msi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
```

```
        veri_stat = EOC$V, rho_est = rho_out, ncp = 30,
        ellipsoid = TRUE, cpst = c(-0.56, 2.31))
}

# IPW estimator
pi_out <- psglm(V ~ CA125 + CA153 + Age, data = EOC, test = TRUE)
if (requireNamespace("webshot2", quietly = TRUE)) {
   rocs("ipw", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, pi_est = pi_out, ncp = 30,
        ellipsoid = TRUE, cpst = c(-0.56, 2.31))
}

# SPE estimator
if (requireNamespace("webshot2", quietly = TRUE)) {
   rocs("spe", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_out, ncp = 30,
        pi_est = pi_out, ellipsoid = TRUE, cpst = c(-0.56, 2.31))
}

# NN estimator
x_mat <- cbind(EOC$CA125, EOC$CA153, EOC$Age)
k_opt <- cv_knn(x_mat = x_mat, dise_vec = dise_vec_na, veri_stat = EOC$V,
                type = "mahala", plot = TRUE)
rho_k_opt <- rho_knn(x_mat = x_mat, dise_vec = dise_vec_na,
                     veri_stat = EOC$V, k = k_opt, type = "mahala")
if (requireNamespace("webshot2", quietly = TRUE)) {
   rocs("knn", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_k_opt, ncp = 30,
        ellipsoid = TRUE, cpst = c(-0.56, 2.31))
}

## Compute TCFs at three cut points
cutps <- rbind(c(0, 0.5), c(0, 1), c(0.5, 1))
rocs.tcf("spe", diag_test = EOC$CA125, dise_vec = dise_vec_na,
         veri_stat = EOC$V, rho_est = rho_out, ncp = 30,
         pi_est = pi_out, cps = cutps)

## End(Not run)
```

---

vus_mar *Estimation methods for volume under ROC surface (VUS) under MAR*

---

## Description

vus_mar computes bias-corrected estimates of the volume under the ROC surface for evaluating the accuracy of a continuous diagnostic test.

## Usage

```
vus_mar(
  method = "full",
  diag_test,
  dise_vec,
  veri_stat,
  rho_est = NULL,
  pi_est = NULL,
  ci = TRUE,
  ci_level = ifelse(ci, 0.95, NULL),
  boot = FALSE,
  n_boot = ifelse(ci, 250, NULL),
  parallel = FALSE,
  ncpus = ifelse(parallel, detectCores()/2, NULL),
  trace = TRUE
)
```

## Arguments

| | |
|---|---|
| method | name of bias-corrected estimation method to be used for estimating the VUS in presence of verification bias. See rocs for more details. |
| diag_test | a numeric vector containing the diagnostic test values. NA values are not admitted. |
| dise_vec | a n * 3 binary matrix with the three columns, corresponding to three classes of the disease status. In row i, 1 in column j indicates that the i-th subject belongs to class j, with j = 1, 2, 3. A row of NA values indicates a non-verified subject. |
| veri_stat | a binary vector containing the verification status (1 verified, 0 not verified). |
| rho_est | a result of a call to rho_mlogit of rho_knn to fit the disease model. |
| pi_est | a result of a call to psglm to fit the verification model. |
| ci | a logical value. If TRUE (default), computes an confidence interval of VUS and tests the null hypothesis H0: VUS = 1/6. |
| ci_level | an confidence level to be used for constructing the confidence interval; default 0.95. |
| boot | a logical value. Default = FALSE. If set to TRUE, a bootstrap resampling is employed to estimate the asymptotic variance of bias-corrected VUS estimates. See asy_var_vus. |
| n_boot | the number of bootstrap replicates, which is used for FULL or KNN estimator, or option boot = TRUE. Usually this will be a single positive integer. |
| parallel | a logical value. If TRUE, a parallel computing is employed to the bootstrap resampling process. |
| ncpus | number of processes to be used in parallel computing. Default is a half of available cores. |
| trace | a logical value. If TRUE, tracing information on the progress of the estimation is produced. |

## Details

The function implements five bias-corrected estimation methods in To Duc et al. (2016, 2020) for estimating VUS of a three-class continuous diagnostic test in presence of verification bias. The estimators are full imputation (FI), mean score imputation (MSI), inverse probability weighted (IPW), semiparametric efficient (SPE) and K nearest-neighbor (KNN), see `rocs`. These estimators work under MAR assumption.

The standard error of the estimates are obtained through the function `asy_var_vus`. In particular, the standard error of the FULL estimate is computed by bootstrap resampling method or by Jackknife approach proposed in Guangming et al. (2013). For the bias-corrected estimates, the standard errors are computed by using asymptotic theory (with respect to FI, MSI, IPW and SPE estimator) or bootstrap resampling method (with respect to KNN estimator). A confidence interval for VUS also is given. A logit transformation is also applied for obtaining the confidence interval.

The default value of the number of bootstrap replicates is 250.

Note that, before apply the functions `vus_mar`, the use of `pre_data` might be needed to check the monotone ordering disease classes and to create the matrix format for disease status.

## Value

`vus_mar` returns an object of class inheriting from "vus_mar" class.

The function `print.vus_mar` can be used to print a summary of the results.

An object of class "vus_mar" is a list containing at least the following components:

| | |
|---|---|
| `vus_fit` | the estimate of VUS. |
| `std` | the standard error, obtained by using asymptotic theory or bootstrap resampling method. |
| `call` | the matched call. |
| `t_stat` | t-statistic. |
| `p_val_norm` | p-value correspond to normal-test. |
| `ci_norm` | the confidence interval of VUS by using normal approximation. |
| `ci_logit` | the confidence interval of VUS via logit transform. |
| `ci_level` | the confidence level used. |
| `boot` | the value of `boot`. |
| `n_boot` | the number of bootstrap replicates used. |

In addition, the name of method used to estimate VUS also is given as the attribute of `vus_fit`.

## References

To Duc, K., Chiogna, M. and Adimari, G. (2020) Nonparametric estimation of ROC surfaces in presence of verification bias. *REVSTAT-Statistical Journal*, **18**, 5, 697–720.

To Duc, K., Chiogna, M. and Adimari, G. (2016) Bias-corrected methods for estimating the receiver operating characteristic surface of continuous diagnostic tests. *Electronic Journal of Statistics*, **10**, 3063-3113.

Guangming, P., Xiping, W. and Wang, Z. (2013) Non-parameteric statistical inference for $P(X < Y < Z)$. *Sankhya A*, **75**, 1, 118-138.

## Examples

```
data(EOC)
head(EOC)


## Not run:
# FULL data estimator
dise_full <- pre_data(EOC$D.full, EOC$CA125)
dise_vec_full <- dise_full$dise_vec
vus_mar("full", diag_test = EOC$CA125, dise_vec = dise_vec_full)

## End(Not run)

## Not run:
# Preparing the missing disease status
dise_na <- pre_data(EOC$D, EOC$CA125)
dise_vec_na <- dise_na$dise_vec
dise_fact_na <- dise_na$dise
# FI estimator
rho_out <- rho_mlogit(dise_fact_na ~ CA125 + CA153 + Age, data = EOC,
                      test = TRUE)

vus_mar("fi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_out)

# MSI estimator
vus_mar("msi", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_out)

# IPW estimator
pi_out <- psglm(V ~ CA125 + CA153 + Age, data = EOC, test = TRUE)
vus_mar("ipw", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, pi_est = pi_out)

# SPE estimator
vus_mar("spe", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_out, pi_est = pi_out)

# KNN estimator, K = 1, Mahalanobis distance
x_mat <- cbind(EOC$CA125, EOC$CA153, EOC$Age)
rho_maha_1nn <- rho_knn(x_mat = x_mat, dise_vec = dise_vec_na,
                        veri_stat = EOC$V, k = 1, type = "mahala")
vus_mar("knn", diag_test = EOC$CA125, dise_vec = dise_vec_na,
        veri_stat = EOC$V, rho_est = rho_maha_1nn)

## End(Not run)
```

# Index