Package 'Qindex'

November 14, 2024

Type Package

Title Continuous and Dichotomized Index Predictors Based on Distribution Quantiles

Version 0.1.7

Date 2024-11-14

Description Select optimal functional regression or dichotomized quantile predictors for survival/logistic/numeric outcome and perform optimistic bias correction for any optimally dichotomized numeric predictor(s), as in Yi, et. al. (2023) <doi:10.1016/j.labinv.2023.100158>.

RoxygenNote 7.3.2

Encoding UTF-8

License GPL-2

Depends R (>= 4.4),

Language en-US

Imports matrixStats, methods, mgcv, plotly, rpart, survival

Suggests knitr, boot, htmlwidgets, Qindex.data

NeedsCompilation no

Author Tingting Zhan [aut, cre, cph] (<https://orcid.org/0000-0001-9971-4844>), Misung Yi [aut, cph] (<https://orcid.org/0000-0002-4007-5408>), Inna Chervoneva [aut, cph] (<https://orcid.org/0000-0002-9104-4505>)

Maintainer Tingting Zhan <tingtingzhan@gmail.com>

Repository CRAN

Date/Publication 2024-11-14 20:10:02 UTC

Contents

`Qindex-package` *Continuous and Dichotomized Index Predictors Based on Distribution Quantiles*

Description

Continuous and dichotomized index predictors based on distribution quantiles.

Author(s)

Maintainer: Tingting Zhan <tingtingzhan@gmail.com> (ORCID) [copyright holder]

Authors:

- Misung Yi <misung.yi@dankook.ac.kr> (ORCID) [copyright holder]
- Inna Chervoneva <Inna.Chervoneva@jefferson.edu> (ORCID) [copyright holder]

References

Selection of optimal quantile protein biomarkers based on cell-level immunohistochemistry data. Misung Yi, Tingting Zhan, Amy P. Peck, Jeffrey A. Hooke, Albert J. Kovatich, Craig D. Shriver, Hai Hu, Yunguang Sun, Hallgeir Rui and Inna Chervoneva. BMC Bioinformatics, 2023. doi:10.1186/s12859023054088

Quantile index biomarkers based on single-cell expression data. Misung Yi, Tingting Zhan, Amy P. Peck, Jeffrey A. Hooke, Albert J. Kovatich, Craig D. Shriver, Hai Hu, Yunguang Sun, Hallgeir Rui and Inna Chervoneva. Laboratory Investigation, 2023. doi:10.1016/j.labinv.2023.100158

Examples

```
### Data Preparation

library(survival)
data(Ki67, package = 'Qindex.data')
Ki67c = within(Ki67[complete.cases(Ki67), , drop = FALSE], expr = {
  marker = log1p(Marker); Marker = NULL
  PFS = Surv(RECFREESURV_MO, RECURRENCE)
})
(npt = length(unique(Ki67c$PATIENT_ID))) # 592

### Step 1: Cluster-Specific Sample Quantiles

Ki67q = clusterQp(marker ~ . - tissueID - inner_x - inner_y | PATIENT_ID, data = Ki67c)
```

```
stopifnot(is.matrix(Ki67q$marker))
head(Ki67q$marker, n = c(4L, 6L))

set.seed(234); id = sort.int(sample.int(n = npt, size = 480L))
Ki67q_0 = Ki67q[id, , drop = FALSE] # training set
Ki67q_1 = Ki67q[-id, , drop = FALSE] # test set

### Step 2 (after Step 1)

## Step 2a: Linear Sign-Adjusted Quantile Indices
(fr = Qindex(PFS ~ marker, data = Ki67q_0))
stopifnot(all.equal.numeric(c(fr), predict(fr)))
integrandSurface(fr)
integrandSurface(fr, newdata = Ki67q_1)

## Step 2b: Non-Linear Sign-Adjusted Quantile Indices
(nlfr = Qindex(PFS ~ marker, data = Ki67q_0, nonlinear = TRUE))
stopifnot(all.equal.numeric(c(nlfr), predict(nlfr)))
integrandSurface(nlfr)
integrandSurface(nlfr, newdata = Ki67q_1)

## view linear and non-linear sign-adjusted quantile indices together
integrandSurface(fr, nlfr)


### Step 2c: Optimal Dichotomizing
set.seed(14837); (m1 = optimSplit_dichotom(
  PFS ~ marker, data = Ki67q_0, nsplit = 20L, top = 2L))
predict(m1)
predict(m1, boolean = FALSE)
predict(m1, newdata = Ki67q_1)


### Step 3 (after Step 1 & 2)

Ki67q_0a = within.data.frame(Ki67q_0, expr = {
  FR = std_IQR(fr)
  nlFR = std_IQR(nlfr)
  optS = std_IQR(marker[,'0.27'])
})
Ki67q_1a = within.data.frame(Ki67q_1, expr = {
  FR = std_IQR(predict(fr, newdata = Ki67q_1))
  nlFR = std_IQR(predict(nlfr, newdata = Ki67q_1))
  optS = std_IQR(marker[,'0.27'])
})
# `optS`: use the best quantile but discard the cutoff identified by [optimSplit_dichotom]
# all models below can also be used on training data `Ki67q_0a`
# naive use
summary(coxph(PFS ~ NodeSt + Tstage + FR, data = Ki67q_1a))
summary(coxph(PFS ~ NodeSt + Tstage + nlFR, data = Ki67q_1a))
summary(coxph(PFS ~ NodeSt + Tstage + optS, data = Ki67q_1a))
# set.seed if necessary
summary(BBC_dichotom(PFS ~ NodeSt + Tstage ~ FR, data = Ki67q_1a))
```

```
# `NodeSt`, `Tstage`: predctors to be used as-is
# `FR` to be dichotomized
# set.seed if necessary
summary(BBC_dichotom(PFS ~ NodeSt + Tstage ~ nlFR, data = Ki67q_1a))
# set.seed if necessary
summary(BBC_dichotom(PFS ~ NodeSt + Tstage ~ optS, data = Ki67q_1a)) # statistically rigorous

# Option 1
summary(BBC_dichotom(PFS ~ NodeSt + Tstage ~ FR, data = Ki67q_1a))

# Option 2:
summary(tmp <- BBC_dichotom(PFS ~ NodeSt + Tstage ~ FR, data = Ki67q_0a))
#coxph(PFS ~ NodeSt + Tstage + I(FR > attr(tmp, 'apparent_cutoff')), data = Ki67q_1a)
coxph(PFS ~ NodeSt + Tstage + I(FR > matrixStats::colMedians(BBC_cutoff(tmp))), data = Ki67q_1a)


# Option 1 and 2 are also applicable to `nlFR` and `optS`
```

`BBC_dichotom` *Bootstrap-based Optimism Correction for Dichotomization*

Description

Multivariable regression model with bootstrap-based optimism correction on the dichotomized predictors.

Usage

```
BBC_dichotom(formula, data, ...)

optimism_dichotom(fom, X, data, R = 100L, ...)

coef_dichotom(fom, X., data)
```

Arguments

`formula`	formula, e.g., `y~z~x` or `y~1~x`. Response y may be double, logical and Surv. Predictors x's to be dichotomized may be one or more numeric vectors and/or one matrix. Additional predictors z's, if any, may be of any type.
`data`	data.frame
`...`	additional parameters, currently not in use
`fom`	formula, e.g., `y~z` or `y~1`, for helper functions, with the response y and additional predictors z's, if any
`X`	numeric matrix of k columns, numeric predictors $x_1, \cdots, x_k$ to be dichotomized
`R`	positive integer scalar, number of bootstrap replicates R, default `100L`
`X.`	logical matrix $\tilde{X}$ of k columns, dichotomized predictors $\tilde{x}_1, \cdots, \tilde{x}_k$

Details

Function BBC_dichotom obtains a multivariable regression model with bootstrap-based optimism correction on the dichotomized predictors. Specifically,

1. Obtain the dichotomizing rules $\mathcal{D}$ of predictors $x_1, \cdots, x_k$ based on response y (via m_rpartD). Multivariable regression (with additional predictors z, if any) with dichotomized predictors $(\tilde{x}_1, \cdots, \tilde{x}_k) = \mathcal{D}(x_1, \cdots, x_k)$ (via helper function coef_dichotom) is the **apparent performance**.
2. Obtain the bootstrap-based optimism based on R copies of bootstrap samples (via helper function optimism_dichotom). The median of bootstrap-based optimism over R bootstrap copies is the **optimism-correction** of the dichotomized predictors $\tilde{x}_1, \cdots, \tilde{x}_k$.
3. Subtract the optimism-correction (in Step 2) from the apparent performance estimates (in Step 1), only for $\tilde{x}_1, \cdots, \tilde{x}_k$. The apparent performance estimates for additional predictors z's, if any, are not modified. Neither the variance-covariance (vcov) estimates nor the other regression diagnostics, e.g., residuals, logLikelihood, etc., of the apparent performance are modified for now. This coefficient-only, partially-modified regression model is the **optimism-corrected performance**.

Value

Function BBC_dichotom returns a coxph, glm or lm regression model, with attributes,

`attr(,'optimism')` the returned object from optimism_dichotom

`attr(,'apparent_cutoff')` a double vector, cutoff thresholds for the k predictors in the apparent model

Details on Helper Functions

Bootstrap-Based Optimism:

Helper function optimism_dichotom computes the bootstrap-based optimism of the dichotomized predictors. Specifically,

1. R copies of bootstrap samples are generated. In the j-th bootstrap sample,
   (a) obtain the dichotomizing rules $\mathcal{D}^{(j)}$ of predictors $x_1^{(j)}, \cdots, x_k^{(j)}$ based on response $y^{(j)}$ (via m_rpartD)
   (b) multivariable regression (with additional predictors $z^{(j)}$, if any) coefficient estimates $\hat{\beta}^{(j)} = \left(\hat{\beta}_1^{(j)}, \cdots, \hat{\beta}_k^{(j)}\right)^t$ of the dichotomized predictors $\left(\tilde{x}_1^{(j)}, \cdots, \tilde{x}_k^{(j)}\right) = \mathcal{D}^{(j)}\left(x_1^{(j)}, \cdots, x_k^{(j)}\right)$ (via coef_dichotom) are the **bootstrap performance estimate**.
2. Dichotomize $x_1, \cdots, x_k$ in the *entire data* using each of the bootstrap rules $\mathcal{D}^{(1)}, \cdots, \mathcal{D}^{(R)}$. Multivariable regression (with additional predictors z, if any) coefficient estimates $\hat{\beta}^{[j]} = \left(\hat{\beta}_1^{[j]}, \cdots, \hat{\beta}_k^{[j]}\right)^t$ of the dichotomized predictors $\left(\tilde{x}_1^{[j]}, \cdots, \tilde{x}_k^{[j]}\right) = \mathcal{D}^{(j)}(x_1, \cdots, x_k)$ (via coef_dichotom) are the **test performance estimate**.
3. Difference between the bootstrap and test performance estimates, an $R \times k$ matrix of $\left(\hat{\beta}^{(1)}, \cdots, \hat{\beta}^{(R)}\right)$ minus another $R \times k$ matrix of $\left(\hat{\beta}^{[1]}, \cdots, \hat{\beta}^{[R]}\right)$, are the **bootstrap-based optimism**.

Multivariable Regression Coefficient Estimates of Dichotomized Predictors $\tilde{x}$'s:

Helper function coef_dichotom fits a multivariable Cox proportional hazards (coxph) model for Surv response, logistic (glm) regression model for logical response, or linear (lm) regression model for gaussian response, with the dichotomized predictors $\tilde{x}_1, \cdots, \tilde{x}_k$ as well as the additional predictors z's.

It is almost inevitable to have duplicates among the dichotomized predictors $\tilde{x}_1, \cdots, \tilde{x}_k$. In such case, the multivariable model is fitted using the unique $\tilde{x}$'s.

Returns of Helper Functions

Of helper function optimism_dichotom:

Helper function optimism_dichotom returns an $R \times k$ double matrix of bootstrap-based optimism, with attributes

`attr(,'cutoff')` an $R \times k$ double matrix, the R copies of bootstrap cutoff thresholds for the k predictors. See attribute `'cutoff'` of function m_rpartD

Of helper function coef_dichotom:

Helper function coef_dichotom returns a double vector of the regression coefficients of dichotomized predictors $\tilde{x}$'s, with attributes

`attr(,'model')` the coxph, glm or lm regression model

In the case of duplicated $\tilde{x}$'s, the regression coefficients of the unique $\tilde{x}$'s are duplicated for those duplicates in $\tilde{x}$'s.

References

For helper function optimism_dichotom:

Ewout W. Steyerberg (2009) Clinical Prediction Models. doi:10.1007/9780387772448

Frank E. Harrell Jr., Kerry L. Lee, Daniel B. Mark. (1996) Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. doi:10.1002/(SICI)10970258(19960229)15:4<361::AIDSIM168>3.0.CO;24

Examples

```
library(survival)
data(flchain, package = 'survival') # see more details from ?survival::flchain
head(flchain2 <- within.data.frame(flchain, expr = {
  mgus = as.logical(mgus)
}))
dim(flchain3 <- subset(flchain2, futime > 0)) # required by ?rpart::rpart
dim(flchain_Circulatory <- subset(flchain3, chapter == 'Circulatory'))

m1 = BBC_dichotom(Surv(futime, death) ~ age + sex + mgus ~ kappa + lambda,
 data = flchain_Circulatory, R = 1e2L)
summary(m1)
matrixStats::colMedians(BBC_cutoff(m1)) # median bootstrap cutoff
attr(m1, 'apparent_cutoff')
```

clusterQp *Cluster-Specific Sample Quantiles*

Description

Sample quantiles in each cluster of observations.

Usage

```
clusterQp(
  formula,
  data,
  f_sum_ = mean.default,
  probs = seq.int(from = 0.01, to = 0.99, by = 0.01),
  ...
)
```

Arguments

formula	formula, including response y, cluster(s) c's, cluster-specific covariate(s) x's to be retained, and cluster-specific covariate(s) z's to be removed from data, e.g., y ~ 1 | c1 cluster c_1, without cluster-specific covariate y ~ 1 | c1/c2 cluster c_1, and cluster c_2 nested in c_1, without cluster-specific covariate y ~ x1 + x2 | c1 cluster c_1, and cluster-specific covariates x_1 and x_2 y ~ . | c1 cluster c_1, and all (supposedly cluster-specific) covariates from data y ~ . - z1 - z2 | c1 cluster c_1, and all (supposedly cluster-specific) covariates, except for z_1 and z_2, from data
data	data.frame
f_sum_	function to summarize the sample quantiles from lower-level cluster c_2 (if present), such as mean.default (default), median.default, max, min, etc.
probs	double vector, probabilities $\mathbf{p} = (p_1, \cdots, p_N)'$ shared across all clusters, where the cluster-specific sample quantiles of response y are calculated. Default seq(.01, .99, by = .01)
...	additional parameters of function quantile

Value

Function clusterQp returns an aggregated data.frame, in which

- the highest cluster c_1 and cluster-specific covariate(s) x's are retained.
  - If the input formula takes form of y ~ . \| c1 or y ~ . - z1 \| c1, then all covariates (except for z_1) are considered cluster-specific;
  - Sample quantiles from lower-level clusters (e.g., c_2) are point-wise summarized using function f_sum_.

- response y is removed; instead, a double matrix of N columns stores the cluster-specific sample quantiles. This matrix
  - is named after the parsed expression of response y in `formula`;
  - colnames are the probabilities $\mathbf{p}$, for the ease of subsequent programming.

Examples

```
# see ?`Qindex-package` for examples
```

`integrandSurface` *Integrand Surface(s) of Sign-Adjusted Quantile Indices Qindex*

Description

An interactive **htmlwidgets** of the perspective plot for Qindex model(s) using package **plotly**.

Usage

```
integrandSurface(
  ...,
  newdata = data,
  proj_Q_p = TRUE,
  proj_S_p = TRUE,
  proj_beta = TRUE,
  n = 501L,
  newid = seq_len(min(50L, .row_names_info(newdata, type = 2L))),
  qlim = range(X, newX),
  axis_col = c("dodgerblue", "deeppink", "darkolivegreen"),
  beta_col = "purple",
  surface_col = c("white", "lightgreen")
)
```

Arguments

`...`	one or more Qindex models based on *a same training set*.
`newdata`	data.frame, with at least the response y^{new} and the double matrix of functional predictor values X^{new} of the *test set*. The predictor X^{new} are tabulated on the same p-grid as the training functional predictor values X. If missing, the training set will be used.
`proj_Q_p`	logical scalar, whether to show the projection of $\hat{S}\big(p, Q_i(p)\big)$ (see sections **Details** and **Value**) to the (p, q)-plain, default `TRUE`
`proj_S_p`	logical scalar, whether to show the projection of $\hat{S}\big(p, Q_i(p)\big)$ to the (p, s)-plain, default `TRUE`
`proj_beta`	logical scalar, whether to show $\hat{\beta}(p)$ on the (p, s)-plain when applicable, default `TRUE`

`n`	integer scalar, fineness of visualization, default `501L`. See parameter `n.grid` of function vis.gam.
`newid`	integer scalar or vector, row indices of `newdata` to be visualized. Default `1:2`, i.e., the first two test subjects. Use `newid = NULL` to disable visualization of `newdata`.
`qlim`	length-2 double vector, range on q-axis. Default is the range of X and X^{new} combined.
`axis_col`	length-3 character vector, colors of the (p, q, s) axes
`beta_col`	character scalar, color of $\hat{\beta}(p)$
`surface_col`	length-2 character vector, color of the integrand surface(s), for lowest and highest surface values

Value

Function integrandSurface returns a pretty **htmlwidgets** created by **R** package **plotly** to showcase the perspective plot of the estimated sign-adjusted integrand surface $\hat{S}(p, q)$.

If a set of training/test subjects is selected (via parameter `newid`), then

- the estimated **sign-adjusted line integrand curve** $\hat{S}\big(p, Q_i(p)\big)$ of subject i is displayed on the surface $\hat{S}(p, q)$;
- the quantile curve $Q_i(p)$ is projected on the (p, q)-plain of the 3-dimensional (p, q, s) cube, if `proj_Q_p=TRUE` (default);
- the user-specified $\tilde{p}$ is marked on the (p, q)-plain of the 3D cube, if `proj_Q_p=TRUE` (default);
- $\hat{S}\big(p, Q_i(p)\big)$ is projected on the (p, s)-plain of the 3-dimensional (p, q, s) cube, if one and only one Qindex model is provided in in put argument `...` and `proj_S_p=TRUE` (default);
- the estimated *linear functional coefficient* $\hat{\beta}(p)$ is shown on the (p, s)-plain of the 3D cube, if one and only one *linear* Qindex model is provided in input argument `...` and `proj_beta=TRUE` (default).

Integrand Surface

The quantile index (QI),

$$\text{QI} = \int_0^1 \beta(p) \cdot Q(p)\, dp$$

with a linear functional coefficient $\beta(p)$ can be estimated by fitting a functional generalized linear model (FGLM, James, 2002) to exponential-family outcomes, or by fitting a linear functional Cox model (LFCM, Gellar et al., 2015) to survival outcomes. More flexible non-linear quantile index (nlQI)

$$\text{nlQI} = \int_0^1 F\big(p, Q(p)\big)\, dp$$

with a bivariate twice differentiable function $F(\cdot, \cdot)$ can be estimated by fitting a functional generalized additive model (FGAM, McLean et al., 2014) to exponential-family outcomes, or by fitting an additive functional Cox model (AFCM, Cui et al., 2021) to survival outcomes.

The estimated **integrand surface** of quantile indices and non-linear quantile indices, defined on $p \in [0,1]$ and $q \in \text{range}\big(Q_i(p)\big)$ for all training subjects $i = 1, \cdots, n$, is

$$\hat{S}_0(p,q) = \begin{cases} \hat{\beta}(p) \cdot q & \text{for QI} \\ \hat{F}(p,q) & \text{for nlQI} \end{cases}$$

Sign-Adjustment

Ideally, we would wish that, *in the training set*, the estimated linear and/or non-linear quantile indices

$$\widehat{\text{QI}}_i = \int_0^1 \hat{S}_0\big(p, Q_i(p)\big)\, dp$$

be *positively correlated* with a more intuitive quantity, e.g., quantiles $Q_i(\tilde{p})$ at a user-specified $\tilde{p}$, for the interpretation of downstream analysis, Therefore, we define the sign-adjustment term

$$\hat{c} = \text{sign}\left(\text{corr}\left(Q_i(\tilde{p}), \widehat{\text{QI}}_i\right)\right), \quad i = 1, \cdots, n$$

as the sign of the correlation between the estimated quantile index $\widehat{\text{QI}}_i$ and the quantile $Q_i(\tilde{p})$, for training subjects $i = 1, \cdots, n$.

The estimated **sign-adjusted integrand surface** is $\hat{S}(p,q) = \hat{c} \cdot \hat{S}_0(p,q)$.

The estimated **sign-adjusted quantile indices** $\int_0^1 \hat{S}\big(p, Q_i(p)\big)\, dp$ are positively correlated with subject-specific sample medians (default $\tilde{p} = .5$) in the training set.

Note

The maintainer is not aware of any functionality of projection of arbitrary curves in package **plotly**. Currently, the projection to (p,q)-plain is hard coded on $(p,q,s = \min(s))$-plain.

References

James, G. M. (2002). *Generalized Linear Models with Functional Predictors*, doi:10.1111/1467-9868.00342

Gellar, J. E., et al. (2015). *Cox regression models with functional covariates for survival data*, doi:10.1177/1471082X14565526

Mathew W. M., et al. (2014) *Functional Generalized Additive Models*, doi:10.1080/10618600.2012.729985

Cui, E., et al. (2021). *Additive Functional Cox Model*, doi:10.1080/10618600.2020.1853550

Examples

```
# see ?`Qindex-package`
```

optimSplit_dichotom *Optimal Dichotomizing Predictors via Repeated Sample Splits*

Description

To identify the optimal dichotomizing predictors using repeated sample splits.

Usage

```
optimSplit_dichotom(
  formula,
  data,
  include = quote(p1 > 0.15 & p1 < 0.85),
  top = 1L,
  nsplit,
  ...
)

split_dichotom(y, x, id, ...)

splits_dichotom(y, x, ids = rSplit(y, ...), ...)

## S3 method for class 'splits_dichotom'
quantile(x, probs = 0.5, ...)
```

Arguments

formula, y, x	formula, e.g., y~X or y~x1+x2. Response y may be double, logical and Surv. Candidate numeric predictors x's may be specified as the columns of one matrix column, e.g., y~X; or as several vector columns, e.g., y~x1+x2. In helper functions, x is a numeric vector.
data	data.frame
include	(optional) language, inclusion criteria. Default (p1>.15 & p1<.85) specifies a user-desired range of p_1 for the candidate dichotomizing predictors. See explanation of p_1 in section **Returns of Helper Functions**.
top	positive integer scalar, number of optimal dichotomizing predictors, default 1L
nsplit, ...	additional parameters for function rSplit
id	logical vector for helper function split_dichotom, indices of training (TRUE) and test (FALSE) subjects
ids	(optional) list of logical vectors for helper function splits_dichotom, multiple copies of indices of repeated training-test sample splits.
probs	double scalar for helper function quantile.splits_dichotom, see quantile

Details

Function optimSplit_dichotom identifies the optimal dichotomizing predictors via repeated sample splits. Specifically,

1. Generate multiple, i.e., repeated, training-test sample splits (via rSplit)
2. For each candidate predictor x_i, find the **median-split-dichotomized regression model** based on the repeated sample splits, see details in section **Details on Helper Functions**
3. Limit the selection of the candidate predictors x's to a user-desired range of p_1 of the split-dichotomized regression models, see explanations of p_1 in section **Returns of Helper Functions**
4. Rank the candidate predictors x's by the decreasing order of the absolute values of the regression coefficient estimate of the median-split-dichotomized regression models. On the top of this rank are the **optimal dichotomizing predictors**.

Value

Function optimSplit_dichotom returns an object of class `'optimSplit_dichotom'`, which is a list of dichotomizing functions, with the input `formula` and `data` as additional attributes.

Details on Helper Functions

Split-Dichotomized Regression Model:

Helper function split_dichotom performs a univariable regression model on the test set with a dichotomized predictor, using a dichotomizing rule determined by a recursive partitioning of the training set. Specifically, given a training-test sample split,

1. find the *dichotomizing rule* $\mathcal{D}$ of the predictor x_0 given the response y_0 in the training set (via rpartD);
2. fit a univariable regression model of the response y_1 with the dichotomized predictor $\mathcal{D}(x_1)$ in the test set.

Currently the Cox proportional hazards (coxph) regression for Surv response, logistic (glm) regression for logical response and linear (lm) regression for gaussian response are supported.

Split-Dichotomized Regression Models based on Repeated Training-Test Sample Splits:

Helper function splits_dichotom fits multiple split-dichotomized regression models split_dichotom on the response y and predictor x, based on each copy of the repeated training-test sample splits.

Quantile of Split-Dichotomized Regression Models:

Helper function quantile.splits_dichotom is a method dispatch of the S3 generic function quantile on splits_dichotom object. Specifically,

1. collect the univariable regression coefficient estimate from each one of the split-dichotomized regression models;
2. find the nearest-even (i.e., `type = 3`) quantile of the coefficients from Step 1. By default, we use the median (i.e., `prob = .5`);
3. the split-dichotomized regression model corresponding to the selected coefficient quantile in Step 2, is returned.

Returns of Helper Functions

Helper function split_dichotom returns a split-dichotomized regression model, which is either a Cox proportional hazards (coxph), a logistic (glm), or a linear (lm) regression model, with additional attributes

`attr(,'rule')` function, dichotomizing rule $\mathcal{D}$ based on the training set

`attr(,'text')` character scalar, human-friendly description of $\mathcal{D}$

`attr(,'p1')` double scalar, $p_1 = \Pr(\mathcal{D}(x_1) = 1)$

`attr(,'coef')` double scalar, univariable regression coefficient estimate of $y_1 \sim \mathcal{D}(x_1)$

Helper function splits_dichotom returns a list of split-dichotomized regression models (split_dichotom).

Helper function quantile.splits_dichotom returns a split-dichotomized regression model (split_dichotom).

Examples

```
# see ?`Qindex-package`
```

---

predict.optimSplit_dichotom

Regression Models with Optimal Dichotomizing Predictors

Description

Regression models with optimal dichotomizing predictor(s), used either as boolean or continuous predictor(s).

Usage

```
## S3 method for class 'optimSplit_dichotom'
predict(
  object,
  formula = attr(object, which = "formula", exact = TRUE),
  newdata = attr(object, which = "data", exact = TRUE),
  boolean = TRUE,
  ...
)
```

Arguments

`object`	an optimSplit_dichotom object
`formula`	(optional) formula to specify the response in test data. If missing, the model formula of training data is used
`newdata`	(optional) test data.frame, candidate numeric predictors x's must have the same name and dimension as the training data. If missing, the training data is used
`boolean`	logical scalar, whether to use the *dichotomized* predictor (default, `TRUE`), or the continuous predictor (`FALSE`)
`...`	additional parameters, currently not in use

Value

Function predict.optimSplit_dichotom returns a list of regression models, coxph model for Surv response, glm for logical response, and lm model for numeric response.

Examples

```
# see ?`Qindex-package`
```

`predict.Qindex` *Predicted Sign-Adjusted Quantile Indices*

Description

To predict sign-adjusted quantile indices of a test set.

Usage

```
## S3 method for class 'Qindex'
predict(object, newdata = object@gam$data, ...)
```

Arguments

`object`	an Qindex object based on the training set.
`newdata`	test data.frame, with at least the response y^{new} and the double matrix of functional predictor values X^{new} of the test set, tabulated on the same p-grid as the training set X. If missing, the training set `object@gam$data` will be used.
`...`	additional parameters, currently not in use.

Details

Function predict.Qindex computes the predicted sign-adjusted quantile indices on the test set, which is the product of function predict.gam return and the correlation sign based on training set (`object@sign`, see Step 3 of section **Details** of function Qindex). Multiplication by `object@sign` is required to ensure that the predicted sign-adjusted quantile indices are positively associated with the **training** functional predictor values at the selected tabulating grid.

Value

Function predict.Qindex returns a double vector, which is the predicted sign-adjusted quantile indices on the test set.

`Qindex` *Sign-Adjusted Quantile Indices*

Description

Sign-adjusted quantile indices based on linear and/or nonlinear functional predictors.

Usage

```
Qindex(formula, data, sign_prob = 0.5, ...)

Qindex_prefit_(formula, data, family, nonlinear = FALSE, ...)
```

Arguments

`formula`	formula, e.g., `y~X`. Response y may be double, logical and Surv. Functional predictor X is a tabulated double matrix; the rows of X correspond to the subjects, while the columns of X correspond to a *common tabulating grid* shared by all subjects. The numeric values of the grid are in the colnames of X
`data`	data.frame, must be a returned object from function clusterQp
`sign_prob`	double scalar between 0 and 1, user-specified probability $\tilde{p}$ for the nearest-even quantile in the grid, which is used to determine the sign-adjustment. Default is `.5`, i.e., the nearest-even median of the grid
`...`	additional parameters for functions s and ti, most importantly `k`
`family`	family object, see function gam. Default values are • `mgcv::cox.ph()` for Surv response y; • `binomial(link = 'logit')` for logical response y; • `gaussian(link = 'identity')` for double response y
`nonlinear`	logical scalar, whether to use nonlinear or linear functional model. Default `FALSE`

Value

Function Qindex returns an Qindex object, which is an instance of an S4 class. See section **Slots** for details.

Slots

`.Data` double vector, sign-adjusted quantile indices, see section **Details** of function integrandSurface

`formula` see section **Arguments**, parameter `formula`

`gam` a gam object

`gpf` a `'gam.prefit'` object, which is the returned object from function gam with argument `fit = FALSE`

`p.value` numeric scalar, p-value for the test of significance of the functional predictor, based on slot `@gam`

`sign` double scalar of either 1 or -1, sign-adjustment, see section **Details** of function integrandSurface

`sign_prob` double scalar, section **Arguments**, parameter `sign_prob`

Examples

```
# see ?`Qindex-package`
```

Index