Package 'fdclassify'

April 23, 2026

Title Supervised Classification for Functional Data via Signed Depth

Version 0.1.0

Description Provides a suite of supervised classifiers for functional data based on the concept of signed depth. The core pipeline computes Fraiman-Muniz (FM) functional depth in either its Tukey or Simplicial variant, derives a signed depth by comparing each curve to a reference median curve via the signed distance integral, and feeds the resulting scalar summary into several classifiers: the k-Ranked Nearest Neighbour (k-RNN) rule, a moving-average smoother, a kernel-density Bayes rule, logistic regression on signed depth and distance to the mode, and a generalised additive model (GAM) classifier. Cross-validation routines for tuning the neighbourhood size k and parametric bootstrap confidence intervals are also included.

License GPL-3

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.1

Depends R (>= 4.1.0)

Imports stats, graphics, mgcv, modeest

Suggests testthat (>= 3.0.0), spelling, knitr, rmarkdown

Config/testthat/edition 3

URL https://github.com/dapr12/fdclassify

BugReports https://github.com/dapr12/fdclassify/issues

NeedsCompilation no

Author Diego Andrés Pérez Ruiz [aut, cre] (ORCID: <https://orcid.org/0000-0002-9969-6065>), Peter Foster [ths]

Maintainer Diego Andrés Pérez Ruiz <diego.perezruiz@manchester.ac.uk>

Repository CRAN

Date/Publication 2026-04-23 20:10:03 UTC

Contents

`bayes_depth_classify` *Bayesian Kernel-Density Classifier on Signed Depth*

Description

Classifies functional curves via Bayes' rule applied to the signed depths. Class-conditional densities $f_g(\mathrm{sdp})$ are estimated by kernel density estimation; prior probabilities are estimated from class frequencies or supplied by the user.

Usage

```
bayes_depth_classify(
  X_train,
  y_train,
  X_test = NULL,
  priors = NULL,
  bw_method = "nrd0",
  grid = NULL,
  type = c("tukey", "simplicial")
)

## S3 method for class 'fd_bayes_fit'
print(x, ...)
```

Arguments

`X_train`	Numeric matrix ($n_{\text{train}} \times M$).
`y_train`	Integer vector of labels (0/1).
`X_test`	Numeric matrix of test curves. If `NULL` the training set is used.

`priors`	Named numeric vector with elements `"0"` and `"1"` giving prior probabilities. Defaults to empirical class proportions.
`bw_method`	Bandwidth selection method passed to `density`: `"nrd0"` (default), `"nrd"`, `"ucv"`, `"bcv"`, or `"SJ"`.
`grid`	Numeric vector of length M (time grid).
`type`	Depth variant.
`x`	A `"fd_bayes_fit"` object.
`...`	Ignored.

Details

The posterior probability that a new observation belongs to class g is

$$P(g \mid \mathrm{sdp}_0) = \frac{f_g(\mathrm{sdp}_0)\,\pi_g}{\sum_{g'} f_{g'}(\mathrm{sdp}_0)\,\pi_{g'}}.$$

Value

An object of class `"fd_bayes_fit"` with components:

predicted Predicted class labels.

prob Matrix with columns `prob_0` and `prob_1`: posterior probabilities.

log_odds Log-odds $\log(f_0\pi_0/f_1\pi_1)$.

sd_train The `"fd_signed_depth"` object for the training set.

Invisibly returns `x`.

Examples

```
set.seed(11)
M <- 80; N <- 100
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(50, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(50, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 50)
fit <- bayes_depth_classify(X, y)
table(fit$predicted, y)
```

`fm_depth` *Fraiman-Muniz Functional Depth*

Description

Computes the Fraiman-Muniz (FM) depth for a collection of discretised functional observations. Two variants are supported: the Tukey-FM depth, which maps depth values to $[0, 1]$, and the Simplicial-FM depth, which maps to $[0.5, 1]$. The two are related by

$$\text{Tukey-FM}_i = 2\,(\text{Simplicial-FM}_i - 1/2).$$

Usage

```
fm_depth(X, grid = NULL, type = c("tukey", "simplicial"))
```

Arguments

`X`	Numeric matrix of dimension $N \times M$, where rows are curves and columns are time points. Missing values are not allowed.
`grid`	Numeric vector of length M giving the observation times (e.g. `seq(0, 1, length.out = M)`). If `NULL` (default) equally spaced points on $[0, 1]$ are assumed.
`type`	Character string, either `"tukey"` (default) or `"simplicial"`, selecting the univariate depth used at each time point.

Details

For a discretised dataset $\{x_i(t_j)\}$, $i = 1, \ldots, N$, $j = 1, \ldots, M$, the Simplicial-FM depth is

$$\text{FM}_i = \sum_{j=2}^{M} (t_j - t_{j-1}) \left[1 - \left| \frac{1}{2} - F_{N,t_j}(x_i(t_j)) \right| \right],$$

where $F_{N,t}$ is the empirical CDF of the sample at time t. The Tukey-FM depth substitutes the Tukey univariate depth $\min\{F_{N,t}(x), 1 - F_{N,t}(x)\}$ at each time point.

Value

A numeric vector of length N containing the FM depth value for each curve.

References

Fraiman, R. and Muniz, G. (2001). Trimmed means for functional data. *Test*, **10**(2), 419–440.

López-Pintado, S. and Romo, J. (2009). On the concept of depth for functional data. *Journal of the American Statistical Association*, **104**(486), 718–734.

Examples

```
set.seed(1)
N <- 50; M <- 100
X <- matrix(rnorm(N * M), nrow = N)
d <- fm_depth(X)
plot(d, xlab = "Curve index", ylab = "Tukey-FM depth")
```

`gam_depth_classify` *GAM Classifier on Signed Depth*

Description

Fits a generalised additive model (GAM) with a smooth term on the signed depth to estimate class membership probabilities. An optional iterative outlier down-weighting scheme is available (Section 3.8 of Perez Ruiz, 2020).

Usage

```
gam_depth_classify(
  X_train,
  y_train,
  X_test = NULL,
  covariates = c("sdp", "sdp+mode"),
  k_gam = 10L,
  n_pc = 10L,
  downweight = FALSE,
  max_iter = 10L,
  grid = NULL,
  type = c("tukey", "simplicial")
)

## S3 method for class 'fd_gam_fit'
print(x, ...)
```

Arguments

`X_train`	Numeric matrix ($N \times M$).
`y_train`	Integer vector of labels (0/1).
`X_test`	Numeric matrix. If `NULL` the training set is used.
`covariates`	Character: `"sdp"` (default) or `"sdp+mode"` to include a second smooth on the signed distance to the mode.
`k_gam`	Basis dimension for `s`. Default `10`.
`n_pc`	Number of PCs for mode estimation. Default `10`.
`downweight`	Logical; if `TRUE` iterate with outlier down-weighting. Default `FALSE`.

`max_iter`	Maximum down-weighting iterations. Default `10`.
`grid`	Numeric vector of length M.
`type`	Depth variant.
`x`	A `"fd_gam_fit"` object.
`...`	Ignored.

Value

An object of class `"fd_gam_fit"` with components:

predicted Predicted class labels.

prob Matrix with columns `prob_0` and `prob_1`.

gam_fit The fitted `gam` object.

sd_train The `"fd_signed_depth"` object for the training set.

covariates Character: covariates used.

Invisibly returns `x`.

Examples

```
set.seed(17)
M <- 80; N <- 100
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(50, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(50, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 50)
fit <- gam_depth_classify(X, y)
print(fit)
```

`krnn_cv` *Cross-Validation for k-RNN Neighbourhood Size*

Description

Selects the optimal half-neighbourhood size k by R-fold cross-validation, using either the minimum-error rule or the one-standard-error (1-SE) rule (Section 3.3.1 of Perez Ruiz, 2020).

Usage

```
krnn_cv(
  X,
  y,
  k_max = NULL,
  R = 10L,
```

```
  rule = c("min", "1se"),
  grid = NULL,
  type = c("tukey", "simplicial"),
  seed = NULL
)

## S3 method for class 'krnn_cv'
print(x, ...)
```

Arguments

`X`	Numeric matrix ($N \times M$).
`y`	Integer vector of class labels (0/1).
`k_max`	Maximum value of k to evaluate. Default `floor(nrow(X) / 4)`.
`R`	Number of CV folds. Default `10`.
`rule`	Character: `"min"` (default) or `"1se"`.
`grid`	Numeric vector of length M.
`type`	Depth variant.
`seed`	Optional integer seed for reproducibility.
`x`	A `"krnn_cv"` object.
`...`	Ignored.

Value

An object of class `"krnn_cv"` with components:

k_opt Selected optimal k.

cv_error Mean CV misclassification error for each k.

cv_se Standard error of CV error per k.

k_max Maximum k evaluated.

R Number of folds used.

rule The selection rule used.

Invisibly returns `x`.

Examples

```
set.seed(3)
M <- 80
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(50, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(50, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 50)
cv <- krnn_cv(X, y, k_max = 20, R = 5, seed = 1)
print(cv)
```

```
plot_krnn_cv(cv)
```

`krnn_fit` *k-Ranked Nearest Neighbour Classifier for Functional Data*

Description

Fits the k-RNN classifier of Perez Ruiz (2020). Curves are ranked by their signed depth and each new observation is assigned to the majority class among its $2k$ nearest ranked neighbours (k above and k below).

Usage

```
krnn_fit(
  X_train,
  y_train,
  X_test = NULL,
  k = 5L,
  grid = NULL,
  type = c("tukey", "simplicial"),
  sd_train = NULL
)

## S3 method for class 'krnn_fit'
print(x, ...)
```

Arguments

`X_train`	Numeric matrix ($n_{\text{train}} \times M$) of training curves (rows = curves, columns = time points).
`y_train`	Integer vector of class labels (0/1) of length n_{train}.
`X_test`	Numeric matrix of test curves. If `NULL` (default) the training set is used for in-sample evaluation.
`k`	Positive integer: half-neighbourhood size. The total number of neighbours per observation is $2k$. Use `krnn_cv` to select k.
`grid`	Numeric vector of length M giving observation times. Defaults to equally spaced points on $[0, 1]$.
`type`	Depth variant passed to `fm_depth`: `"tukey"` (default) or `"simplicial"`.
`sd_train`	Optional pre-computed `"fd_signed_depth"` object for the training set (avoids recomputation when called repeatedly).
`x`	A `"krnn_fit"` object.
`...`	Ignored.

Value

An object of class `"krnn_fit"` with components:

predicted Integer vector of predicted class labels.

prob Numeric vector of estimated probabilities $\hat{P}(y = 0|\mathrm{sdp})$.

sd_train The `"fd_signed_depth"` object for the training set.

y_train The training labels.

k The value of k used.

Invisibly returns `x`.

References

Perez Ruiz, D. A. (2020). *Supervised Classification for Functional Data*. PhD thesis, University of Manchester. Section 3.2.

See Also

`krnn_cv`, `predict.krnn_fit`

Examples

```
set.seed(7)
M <- 100; n <- 80
t <- seq(0, 1, length.out = M)
X0 <- t(replicate(n / 2, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(n / 2, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = n / 2)
idx_tr <- sample(n, 60)
fit <- krnn_fit(X[idx_tr, ], y[idx_tr], X[-idx_tr, ], k = 5)
print(fit)
```

`krnn_smoother` *k-RNN Moving-Average Smoother*

Description

Estimates the conditional probability $\hat{P}(y = 0 \mid \mathrm{sdp})$ using the running-mean smoother (moving average) with span $2k$. This is the regression interpretation of the k-RNN (Section 3.4 of Perez Ruiz, 2020).

Usage

```
krnn_smoother(
  X,
  y,
  k = 10L,
  grid_eval = NULL,
  boot = FALSE,
  B = 200L,
  alpha = 0.05,
  grid_fd = NULL,
  type = c("tukey", "simplicial")
)
```

Arguments

X	Numeric matrix ($N \times M$).
y	Integer vector of labels (0/1).
k	Half-neighbourhood size.
grid_eval	Optional numeric vector of signed-depth values at which to evaluate the smoother. Defaults to the training signed depths.
boot	Logical; if TRUE compute bootstrap-t confidence intervals. Default FALSE.
B	Number of bootstrap replicates (only used when boot = TRUE). Default 200.
alpha	Nominal level: coverage is $1 - \alpha$. Default 0.05.
grid_fd	Numeric vector of length M (time grid).
type	Depth variant.

Value

An object of class "krnn_smoother" with components:

sdp_eval Evaluation points (signed depths).

prob Estimated conditional probabilities.

ci_lower, ci_upper Bootstrap CI bounds (if boot = TRUE).

k Half-neighbourhood size used.

Examples

```
set.seed(5)
M <- 100; N <- 80
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(40, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(40, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 40)
sm <- krnn_smoother(X, y, k = 10)
plot_krnn_smoother(sm)
```

`logistic_depth_classify` *Logistic Regression Classifier on Signed Depth*

Description

Fits a logistic regression model with class label as response and signed depth (and optionally signed distance to the mode) as covariates (Section 3.7 of Perez Ruiz, 2020).

Usage

```
logistic_depth_classify(
  X_train,
  y_train,
  X_test = NULL,
  model = c("sdp", "sdp+mode", "sdp*mode"),
  n_pc = 10L,
  grid = NULL,
  type = c("tukey", "simplicial")
)

## S3 method for class 'fd_logistic_fit'
print(x, ...)
```

Arguments

`X_train`	Numeric matrix ($n_{\text{train}} \times M$).
`y_train`	Integer vector of labels (0/1).
`X_test`	Numeric matrix. If `NULL` the training set is used.
`model`	Character: `"sdp"` (default), `"sdp+mode"`, or `"sdp*mode"`.
`n_pc`	Number of principal components for mode estimation. Default 10.
`grid`	Numeric vector of length M.
`type`	Depth variant.
`x`	A `"fd_logistic_fit"` object.
`...`	Ignored.

Details

Three model formulae are supported:

`"sdp"` Model 1: signed depth only.

`"sdp+mode"` Model 2: signed depth plus signed distance to mode.

`"sdp*mode"` Model 3: Model 2 plus interaction term.

Value

An object of class "fd_logistic_fit" with components:

predicted Predicted class labels.

prob Matrix with columns prob_0 and prob_1.

glm_fit The fitted glm object.

sd_train The "fd_signed_depth" object for the training set.

model Character: model formula used.

Invisibly returns x.

Examples

```
set.seed(13)
M <- 80; N <- 100
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(50, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(50, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 50)
fit <- logistic_depth_classify(X, y, model = "sdp")
summary(fit$glm_fit)
```

---

plot_krnn_cv *Plot a krnn_cv Object*

Description

Plots the cross-validation error curve against the neighbourhood size k, with standard-error bars and a vertical line at the selected optimum.

Usage

```
plot_krnn_cv(x, ...)
```

Arguments

x	A "krnn_cv" object returned by krnn_cv.
...	Additional graphical parameters passed to plot.default.

Value

Invisibly returns x.

Examples

```
set.seed(3)
M <- 80
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(50, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(50, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 50)
cv <- krnn_cv(X, y, k_max = 20, R = 5, seed = 1)
plot_krnn_cv(cv)
```

---

plot_krnn_smoother *Plot a krnn_smoother Object*

Description

Plots the estimated conditional probability $\hat{P}(y = 0 \mid \mathrm{sdp})$ against the signed depth, with optional bootstrap confidence bands.

Usage

```
plot_krnn_smoother(x, ...)
```

Arguments

`x`	A `"krnn_smoother"` object returned by `krnn_smoother`.
`...`	Additional graphical parameters passed to `plot.default`.

Value

Invisibly returns `x`.

Examples

```
set.seed(5)
M <- 100; N <- 80
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(40, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(40, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = 40)
sm <- krnn_smoother(X, y, k = 10)
plot_krnn_smoother(sm)
```

`predict.krnn_fit` *Predict Method for krnn_fit Objects*

Description

Classifies new functional observations using a fitted `krnn_fit` object.

Usage

```
## S3 method for class 'krnn_fit'
predict(object, newdata, y_true = NULL, ...)
```

Arguments

`object`	A "`krnn_fit`" object.
`newdata`	Numeric matrix of new curves ($n_{\text{new}} \times M$).
`y_true`	Optional integer vector of true labels for computing the misclassification rate.
`...`	Ignored.

Value

A list with components `predicted`, `prob`, and (if `y_true` is supplied) `misclass`.

Examples

```
set.seed(7)
M <- 100; n <- 80
t <- seq(0, 1, length.out = M)
X0 <- t(replicate(n / 2, sin(2 * pi * t) + rnorm(M, sd = 0.4)))
X1 <- t(replicate(n / 2, cos(2 * pi * t) + rnorm(M, sd = 0.4)))
X  <- rbind(X0, X1)
y  <- rep(0:1, each = n / 2)
idx_tr <- sample(n, 60)
fit  <- krnn_fit(X[idx_tr, ], y[idx_tr], k = 5)
pred <- predict(fit, newdata = X[-idx_tr, ], y_true = y[-idx_tr])
pred$misclass
```

reference_curve *Reference (Median) Curve*

Description

Returns the curve (or average of curves) that attains the maximum Fraiman-Muniz depth over the combined sample — the functional analogue of the median.

Usage

```
reference_curve(X, depth = NULL, ...)
```

Arguments

X	Numeric matrix ($N \times M$) of curves.
depth	Numeric vector of length N of pre-computed FM depth values. If NULL (default), fm_depth is called internally with type = "tukey".
...	Additional arguments forwarded to fm_depth when depth is NULL.

Value

A numeric vector of length M.

Examples

```
set.seed(1)
X <- matrix(rnorm(50 * 100), nrow = 50)
ref <- reference_curve(X)
plot(ref, type = "l", xlab = "t", ylab = "x(t)",
     main = "Reference (median) curve")
```

signed_depth *Signed Depth*

Description

The core transformation of the k-RNN pipeline. For each curve computes

$$\mathrm{sdp}(x_i(t)) = \mathrm{sgn}\left\{ \int_I (x_i(t) - x_{\mathrm{ref}}(t))\, dt \right\} \times D(x_i(t)),$$

where $D(\cdot)$ is the Fraiman-Muniz depth and the sign is derived from the signed distance integral (equation 3.2 of Perez Ruiz, 2020). Curves above the reference receive a positive signed depth; curves below receive a negative one.

Usage

```
signed_depth(
  X,
  grid = NULL,
  type = c("tukey", "simplicial"),
  x_ref = NULL,
  depth = NULL
)

## S3 method for class 'fd_signed_depth'
print(x, ...)
```

Arguments

X	Numeric matrix ($N \times M$) with curves in rows.
grid	Numeric vector of length M. Defaults to equally spaced points on $[0, 1]$.
type	Depth variant: "tukey" (default) or "simplicial".
x_ref	Optional pre-computed reference curve (numeric vector of length M). Computed internally if NULL.
depth	Optional pre-computed FM depth vector of length N. Computed internally if NULL.
x	A "fd_signed_depth" object.
...	Ignored.

Value

An object of class "fd_signed_depth" with components:

sdp Numeric vector of length N: the signed depths.

depth Numeric vector of length N: raw FM depths.

sdi Numeric vector of length N: signed distance integrals.

x_ref Numeric vector of length M: reference curve.

grid Numeric vector of length M: time grid used.

type Character: depth variant used.

N Number of curves.

M Number of time points.

Invisibly returns x.

References

Perez Ruiz, D. A. (2020). *Supervised Classification for Functional Data*. PhD thesis, University of Manchester.

Examples

```
set.seed(42)
N <- 60; M <- 100
t  <- seq(0, 1, length.out = M)
X0 <- t(replicate(N / 2, sin(2 * pi * t) + rnorm(M, sd = 0.3)))
X1 <- t(replicate(N / 2, cos(2 * pi * t) + rnorm(M, sd = 0.3)))
X  <- rbind(X0, X1)
sd_obj <- signed_depth(X)
plot(sd_obj$sdp, col = rep(1:2, each = N / 2),
     xlab = "Curve index", ylab = "Signed depth",
     main = "Signed depth by group")
abline(h = 0, lty = 2)
```

---

signed_distance_integral

Signed Distance Integral

Description

For each curve $x_i(t)$, computes

$$\int_I \big(x_i(t) - x_{\text{ref}}(t)\big)\, dt,$$

which is positive when the curve lies predominantly above the reference and negative when it lies below. This integral assigns a sign to the depth of each curve (equation 3.1 of Perez Ruiz, 2020).

Usage

```
signed_distance_integral(X, x_ref = NULL, grid = NULL)
```

Arguments

`X`	Numeric matrix ($N \times M$).
`x_ref`	Numeric vector of length M: the reference curve. If `NULL` (default), `reference_curve` is called internally.
`grid`	Numeric vector of length M with observation times. Defaults to equally spaced points on $[0, 1]$.

Value

A numeric vector of length N.

See Also

`signed_depth`, `reference_curve`

Examples

```
set.seed(1)
X <- matrix(rnorm(40 * 80), nrow = 40)
sdi <- signed_distance_integral(X)
hist(sdi, main = "Signed distance integrals", xlab = "SDI")
```

`simulate_fd` *Simulate a Two-Group Functional Dataset*

Description

Generates a labelled functional dataset from two Gaussian processes, useful for illustrating and testing the classifiers in `fdclassify`. The two groups differ in their mean function:

$$x_i(t) = \mu_g(t) + \varepsilon_i(t), \quad \varepsilon_i(t) \sim \mathcal{GP}(0, \sigma^2),$$

where $\mu_0(t) = A_0 \sin(2\pi f_0 t)$ and $\mu_1(t) = A_1 \cos(2\pi f_1 t)$ by default.

Usage

```
simulate_fd(
  n0 = 50L,
  n1 = 50L,
  M = 100L,
  sigma = 0.4,
  A0 = 1,
  A1 = 1,
  f0 = 1,
  f1 = 1,
  seed = NULL
)
```

Arguments

`n0`	Number of curves in group 0. Default 50.
`n1`	Number of curves in group 1. Default 50.
`M`	Number of time points. Default 100.
`sigma`	Standard deviation of the noise. Default 0.4.
`A0`, `A1`	Amplitudes of the mean functions. Both default to 1.
`f0`, `f1`	Frequencies of the mean functions. Both default to 1.
`seed`	Optional integer seed.

Value

A list with:

`X` Numeric matrix $(n_0 + n_1) \times M$.

`y` Integer vector of labels (0 or 1).

`grid` Numeric vector of length M on $[0, 1]$.

Examples

```
dat <- simulate_fd(n0 = 40, n1 = 40, seed = 1)
matplot(t(dat$X[dat$y == 0, ]), type = "l", col = "steelblue",
        lty = 1, xlab = "t", ylab = "x(t)", main = "Simulated curves")
matlines(t(dat$X[dat$y == 1, ]), col = "firebrick", lty = 1)
legend("topright", legend = c("Group 0", "Group 1"),
       col = c("steelblue","firebrick"), lty = 1)
```

Index