

Package: fdclassify (via r-universe)

May 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 ($\geq 4.1.0$)

Imports stats, graphics, mgcv, modeest

Suggests testthat ($\geq 3.0.0$), 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 <https://cran.r-universe.dev>

Date/Publication 2026-04-23 22:05:20 UTC

RemoteUrl <https://github.com/cran/fdclassify>

RemoteRef HEAD

RemoteSha 76e1e87fbc10d2b32cd41f8863afd6c0ec90b53f

Contents

bayes_depth_classify	2
fm_depth	4
gam_depth_classify	5
knn_cv	6
knn_fit	8
knn_smoother	9
logistic_depth_classify	11
plot_knn_cv	12
plot_knn_smoother	13
predict.knn_fit	14
reference_curve	15
signed_depth	15
signed_distance_integral	17
simulate_fd	18

Index 20

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 $f_g(\text{sdp})$ 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

<code>X_train</code>	Numeric matrix ($n_{\text{train}} \times M$).
<code>y_train</code>	Integer vector of labels (0/1).
<code>X_test</code>	Numeric matrix of test curves. If NULL the training set is used.
<code>priors</code>	Named numeric vector with elements "0" and "1" giving prior probabilities. Defaults to empirical class proportions.
<code>bw_method</code>	Bandwidth selection method passed to <code>density</code> : "nrd0" (default), "nrd", "ucv", "bcv", or "SJ".
<code>grid</code>	Numeric vector of length M (time grid).
<code>type</code>	Depth variant.
<code>x</code>	A "fd_bayes_fit" object.
<code>...</code>	Ignored.

Details

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

$$P(g \mid \text{sd}p_0) = \frac{f_g(\text{sd}p_0) \pi_g}{\sum_{g'} f_{g'}(\text{sd}p_0) \pi_{g'}}.$$

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 $\{x_i(t_j)\}$, $i = 1, \dots, N$, $j = 1, \dots, M$, 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

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

<code>max_iter</code>	Maximum down-weighting iterations. Default 10.
<code>grid</code>	Numeric vector of length M .
<code>type</code>	Depth variant.
<code>x</code>	A "fd_gam_fit" object.
<code>...</code>	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

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

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 $\hat{P}(y = 0 | \text{sdp})$ 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

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

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

<code>object</code>	A "krnn_fit" object.
<code>newdata</code>	Numeric matrix of new curves ($n_{\text{new}} \times M$).
<code>y_true</code>	Optional integer vector of true labels for computing the misclassification rate.
<code>...</code>	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	<i>Reference (Median) Curve</i>
-----------------	---------------------------------

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

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

Description

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

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

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

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

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

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

<code>X</code>	Numeric matrix ($N \times M$).
<code>x_ref</code>	Numeric vector of length M : the reference curve. If NULL (default), reference_curve is called internally.
<code>grid</code>	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

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

bayes_depth_classify, 2

density, 3

fm_depth, 4, 8, 15

gam, 6

gam_depth_classify, 5

glm, 12

krnn_cv, 6, 8, 9, 12

krnn_fit, 8, 14

krnn_smoother, 9, 13

logistic_depth_classify, 11

plot.default, 12, 13

plot_krnn_cv, 12

plot_krnn_smoother, 13

predict.krnn_fit, 9, 14

print.fd_bayes_fit
 (bayes_depth_classify), 2

print.fd_gam_fit (gam_depth_classify), 5

print.fd_logistic_fit
 (logistic_depth_classify), 11

print.fd_signed_depth (signed_depth), 15

print.krnn_cv (krnn_cv), 6

print.krnn_fit (krnn_fit), 8

reference_curve, 15, 17

s, 5

signed_depth, 15, 17

signed_distance_integral, 17

simulate_fd, 18