

# Package ‘ExtremesBinaryClassifier’

November 5, 2021

**Type** Package

**Description** Provides functions for the empirical risk estimation of binary classifiers for extremes, as proposed in Legrand, J., Naveau, P., and Oesting, M. (2021) <add doi here>.

**Title** Evaluation of binary classifiers for extremes

**Version** 1.0.1

**Date** 2021-10-22

**Author** Juliette Legrand [aut, cre]

**Maintainer** Juliette Legrand <juliette.legrand@lsce.ipsl.fr>

**Depends** R (>= 3.1.0)

**Imports** graphicalExtremes

**Remotes** github::sebastian-engelke/graphicalExtremes

**Suggests** glmnet, randomForest, rpart, e1071

**License** to add (ex. GPL (>= 2))

**Encoding** UTF-8

**URL** to add

**RoxygenNote** 7.1.2

**LazyData** true

## R topics documented:

dataDanube . . . . .	2
EmpiricalRisk . . . . .	2
FrechetMargin . . . . .	4
LinearClassifier . . . . .	4

<b>Index</b>	<b>6</b>
--------------	----------

---

dataDanube	<i>Danube river discharges</i>
------------	--------------------------------

---

### Description

Daily river discharges, measured in  $m^3/s$ , at 31 stations spread over the upper Danube basin. The data set covers the period from 1960 to 2010 but only the months of June, July, and August are retained. These data are already declustered following Mhalla et al.(2020) methodology.

### Usage

```
graphicalExtremes::danube
```

### Source

Bavarian Environmental Agency (<http://www.gkd.bayern.de>)

### References

Asadi, P., Davison, A.C., and Engelke, S. (2015). Extremes on river networks. The Annals of Applied Statistics, 9(4), 2023-2050.

### Examples

```
graphicalExtremes::danube
```

---

EmpiricalRisk	<i>A Risk estimation function</i>
---------------	-----------------------------------

---

### Description

Compute the empirical risk defined  $\hat{R}_n(g)$  given the output of a classifier  $g$  and the true binary outcomes  $Y$ . If  $\epsilon > 0$ , supply the values of the trained classifier on the thresholded data  $g_\epsilon$  and the true binary outcomes  $Y_\epsilon := +1$  if  $H > \epsilon_u$  and  $-1$  otherwise.

### Usage

```
EmpiricalRisk(Y, Y.eps = NULL, g, g.eps = NULL, epsilon = 0)
```

### Arguments

<code>Y</code>	vector of the true binary outcomes
<code>Y.eps</code>	vector of the true binary outcomes in the extreme region
<code>g</code>	vector of the predicted binary outcomes from a given classifier
<code>g.eps</code>	vector of the predicted binary outcomes in the extreme region from the same classifier
<code>epsilon</code>	single numeric between 0 and 1 giving the amount of data we want to remove

**Details**

to add

**References**

Legrand et al.

**See Also**

[LinearClassifier](#)

**Examples**

```
require(rpart)

set.seed(123)
## Reproduce the simulation example from Legrand et al.
nsim <- 1e4
X1 <- 1/(runif(nsim)^(1/3))
X2 <- 1/(runif(nsim)^(1/2))
P <- 1/(runif(nsim)^(1/2))
H <- X1 + P
## Compute the two thresholds u and epsilon_u
u <- quantile(H,probs=0.97)
eps <- 0.7
eps_u <- u*eps
## Split data between training and testing sets
ii <- sample.int(length(H), size = 0.7*length(H), replace=F)
Xtrain <- cbind(X1[ii], X2[ii])
Xtest <- cbind(X1[-ii], X2[-ii])
Htrain <- H[ii]
Htest <- H[-ii]

## Linear classifier
## Train the linear classifier
init <- lm(H ~ X1 + X2, data = data.frame(H = H, X1 = X1, X2 = X2))$coefficients[2:3]
linclass <- LinearClassifier(X = Xtrain, thresh = eps_u, H = Htrain, initials = init)
## Compute the predicted binary outcome on the test set from all the data and only with the extreme region data
glin <- 2*(as.vector(linclass$theta %*% t(Xtest)) > u) - 1
glineps <- 2*(as.vector(linclass$theta %*% t(Xtest)) > eps_u) - 1
## Compute the true binary outcome on the test set from all the data and only with the extreme region data
Ytesteps <- 2*(Htest > eps_u) - 1
Ytest <- 2*(Htest > u) - 1
EmpiricalRisk(Y = Ytest, Y.eps = Ytesteps, g = glin, g.eps = glineps, epsilon = eps)

## Comparison with regression tree
Ytraineps <- 2*(Htrain > eps_u) - 1
treeclass <- rpart(y~., data=data.frame(x=Xtrain, y = as.factor(Ytraineps)), method = "class")
gtree <- as.numeric(as.character(predict(treeclass, newdata = data.frame(x = Xtest, y = Ytest), type="class"))))
gtreeeps <- as.numeric(as.character(predict(treeclass,
newdata = data.frame(x = Xtest, y = Ytesteps), type = 'class'))))
EmpiricalRisk(Y = Ytest, Y.eps = Ytesteps, g = gtree, g.eps = gtreeeps, epsilon = eps)
```

---

FrechetMargin	<i>Unit-Frechet transformation</i>
---------------	------------------------------------

---

**Description**

Transforms data to unit-Frechet scale using rank transformation

**Usage**

```
FrechetMargin(X)
```

**Arguments**

X	a numeric vector
---	------------------

**References**

Legrand et al.

**Examples**

```
FrechetMargin()
```

---

LinearClassifier	<i>Optimal linear classifier</i>
------------------	----------------------------------

---

**Description**

Compute the optimal linear classifier as defined in section ??? by minimizing the empirical risk (defined by `emp.risk.lin`). Initial values must be provided which can be estimated by performing a classical linear regression (`lm`) for example.

**Usage**

```
LinearClassifier(X, thresh, H, initials)
```

**Arguments**

X	numeric matrix corresponding to the input data we want to classify
thresh	single numeric giving the threshold over which an extreme event is defined
H	numeric vector corresponding to the latent variable that we wish to predict
initials	initial values for the parameters of the linear classifier to be optimized over

**Value**

theta value of the linear classifier

theta	the optimal parameters for the linear classifier
Risk	the value of the risk corresponding theta

**References**

Legrand et al.

**Examples**

```
set.seed(123)
## Reproduce the simulation example from Legrand et al.
nsim <- 1e4
X1 <- 1/(runif(nsim)^(1/3))
X2 <- 1/(runif(nsim)^(1/2))
P <- 1/(runif(nsim)^(1/2))
H <- X1 + P
u <- quantile(H,probs=0.97)
init <- lm(H ~ X1 + X2, data=data.frame(H = H, X1 = X1, X2 = X2))$coefficients[2:3]
LinearClassifier(X = cbind(X1, X2), thresh = u, H = H, initials = init)
```

# Index

dataDanube, [2](#)

EmpiricalRisk, [2](#)

FrechetMargin, [4](#)

LinearClassifier, [3](#), [4](#)