

Package: sisureg (via r-universe)

September 14, 2024

Title Sign-Simplicity-Regression-Solver

Version 1.1.1

Description Implementation of the SSR-Algorithm. The Sign-Simplicity-Regression model is a nonparametric statistical model which is based on residual signs and simplicity assumptions on the regression function. Goal is to calculate the most parsimonious regression function satisfying the statistical adequacy requirements. Theory and functions are specified in Metzner (2020, ISBN: 979-8-68239-420-3, ``Trendbasierte Prognostik") and Metzner (2021, ISBN: 979-8-59347-027-0, ``Adäquates Maschinelles Lernen").

License GPL (>= 2)

Encoding UTF-8

Imports zoo, raster, reticulate

RoxygenNote 7.1.1

NeedsCompilation yes

Author Lars Metzner [aut, cre]

Maintainer Lars Metzner <lars.metzner@ppi.de>

Repository CRAN

Date/Publication 2023-09-20 13:50:02 UTC

Contents

axe	2
axe_narch_model	3
axe_narch_predict	4
fii_model	5
fii_prediction	6
onnx_load	7
onnx_save	8
psplot	9
psplot3d	10

psplotnd	11
psvalid	12
runvalid	13
snarch	13
ssr	15
ssr3d	16
ssr3d_predict	17
ssrmlp_predict	18
ssrmlp_train	19
ssrnd	20
ssrnd_predict	21
ssr_predict	22
tauM	23

Index	24
--------------	-----------

axe	<i>Data model for the AxE-Model (Axiomatic Econometric Modeling Paradigm)</i>
-----	-------------------------------------------------------------------------------

Description

Calculation of the relevant data for the AxE-model from a financial time series: trend, volatility, change in quotes and risk level.

Usage

```
axe(quotes)
```

Arguments

quotes	financial time series
--------	-----------------------

Value

data frame	
quotes	the given time series
trend5	5-day trend
trend10	10-day trend
trend20	20-day trend
vol5	5-day volatility
vol10	10-day volatility
vol20	20-day volatility
chg5	5-day price change
chg10	10-day price change

chng20	20-day price change
risk5	5-day risk level
risk10	10-day risk level
risk20	20-day risk level

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2020) *Trendbasierte Prognostik*. Independently Published.

Examples

```
set.seed(1234)
s <- 13000 + cumsum(rnorm(100))
df_axe <- axe(s)
op <- par(mfrow=c(3,1))
plot(s, type = "l")
plot(df_axe$trend5, type = "l")
abline(a = 0, b = 0)
plot(df_axe$vola5, type = "l")
par(op)
```

axe_narch_model *implementation of the AxE model based on the ssr-MLP*

Description

Trains a 2-layer MLP with a given time series of quotes with price changes or volatility as target value. The coordinates (or independent factors) are given through the AxE model)

Usage

```
axe_narch_model(quotes, T, tgt)
```

Arguments

quotes	array with observations.
T	period: T = 5, 10 or 20.
tgt	target variable: tgt = 'trend' or 'vola'.

Value

model	the trained model for prediction.
-------	-----------------------------------

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
set.seed(1234)
n <- 250
s <- 13000 + cumsum(rnorm(n))
T = 20
# create model for 5-day trend
model <- axe_narch_model(s, T, 'trend')
# calculate prognosis for trend
s_ <- s[n] + cumsum(rnorm(20))
s_T <- axe_narch_predict(model, s_, 'trend')
# plot the results
plot(seq(1:20), s_, type = "l",
      xlim = c(0,21+T), ylim = c(min(s_, s_T)-5, max(s_, s_T)+5))
points(20+T, s_T, col='red', pch = 16)
# create model for 5-day vola
model <- axe_narch_model(s, T, 'vola')
r_T <- axe_narch_predict(model, s_, 'vola')
lines(c(20+T,20+T), c(s_T-r_T, s_T+r_T), col='orange')
```

axe_narch_predict *Prediction function for the AxE-NARCH model*

Description

Calculates the prediction for a given model

Usage

```
axe_narch_predict(model, quotes, tgt)
```

Arguments

model	previously calculated model.
quotes	20 days of history.
tgt	target variable: tgt = 'trend' or 'vola'.

Value

prediction	prediction based in the model and the given coordinates.
------------	----------------------------------------------------------

Author(s)

Dr. Lars Metzner

ReferencesDr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.**Examples**

```

set.seed(1234)
n <- 250
s <- 13000 + cumsum(rnorm(n))
T = 20
# create model for 5-day trend
model <- axe_narch_model(s, T, 'trend')
# calculate prognosis for trend
s_ <- s[n] + cumsum(rnorm(20))
s_T <- axe_narch_predict(model, s_, 'trend')
# plot the results
plot(seq(1:20), s_, type = "l",
      xlim = c(0,21+T), ylim = c(min(s_, s_T)-5, max(s_, s_T)+5))
points(20+T, s_T, col='red', pch = 16)
# create model for 5-day vola
model <- axe_narch_model(s, T, 'vola')
r_T <- axe_narch_predict(model, s_, 'vola')
lines(c(20+T,20+T), c(s_T-r_T, s_T+r_T), col='orange')

```

fii_model

*Factor-wise Influence Indicator (Model-fii) for a given ssrmlp model***Description**

The Model-fii depicts the overall influence of the input factors on the resulting trained ssrmlp model. For computation a unit matrix is used to accumulate the weights for each factor separately.

Usage

```
fii_model(W)
```

Arguments

W a trained ssrmlp model

Value

fii array of influence indicators

Author(s)

Dr. Lars Metzner

Examples

```
# generate data
set.seed(42)
x <- rnorm(300)
y <- rnorm(300)
z <- rnorm(300) + atan2(x, y)
# coordinates
X <- matrix(cbind(x,y), ncol = 2)
Y <- as.double(z)
# Training
W <- ssrmlp_train(X, Y)
fii_model(W)
```

fii_prediction

Factor-wise Influence Indicator (Prediction-fii) for a given ssrmlp model regarding a given input vector

Description

The Prediction-fii depicts the overall influence of the given input factors on the resulting prediction from a trained ssrmlp model. For computation the components of the input vectors are taken separately as input for the model.

Usage

```
fii_prediction(W, x)
```

Arguments

W	a trained ssrmlp model
x	a matrix of input vectors

Value

fii	array of influence indicators
-----	-------------------------------

Author(s)

Dr. Lars Metzner

Examples

```
# generate data
set.seed(42)
x <- rnorm(300)
y <- rnorm(300)
z <- rnorm(300) + atan2(x, y)
# coordinates
X <- matrix(cbind(x,y), ncol = 2)
Y <- as.double(z)
# Training
W <- ssrmlp_train(X, Y)
fii_prediction(W, X)
```

onnx_load

Loading a ssrmlp model from ONNX file

Description

Loading a ssrmlp model from ONNX file (also see [onnx.ai](#)). This function uses the onnx python implementation, hence a python environment including modules onnx and numpy is required.

Usage

```
onnx_load(filename)
```

Arguments

filename fully qualified file name

Value

W parameters of ssrmlp model

Author(s)

Dr. Lars Metzner

Examples

```
# generate data
set.seed(42)
x <- rnorm(300)
y <- rnorm(300)
z <- rnorm(300) + atan2(x, y)
# coordinates
X <- matrix(cbind(x,y), ncol = 2)
Y <- as.double(z)
```

```

# Training
ssrmlp_model <- ssrmlp_train(X, Y)
# only if python is available
if (reticulate::py_module_available('onnx')) {
  # save in ONNX format
  onnx_save(ssrmlp_model, 'file.onnx')
  # loading the file in ONNX format
  W <- onnx_load('file.onnx')
  # prediction with original implementation
  p <- t(c(0.25, 0.25))
  pred <- ssrmlp_predict(p, W)
  # predict with TensorFlow
  onnx <- reticulate::import("onnx")
  backend <- reticulate::import('onnx_tf.backend')
  model = onnx$load('file.onnx')
  tf_rep = backend$prepare(model)
  tf_y = tf_rep$run(p)
  print(tf_y$Y-pred)
  # cleanup
  file.remove('file.onnx')
  # to avoid NOTE in R CHECK
  tempfile <- reticulate::import("tempfile")
  tmp <- tempfile$gettempdir()
  if (dir.exists(file.path(tmp, "__pycache__"))) {
    unlink(file.path(tmp, "__pycache__"), recursive = TRUE, force = TRUE)
  }
  tmp_py_files <- list.files(tmp,
                             pattern = "^__autograph_generated_file.*py$", full.names = TRUE)
  file.remove(tmp_py_files)
}

```

onnx_save

Saving a ssrmlp model in onnx format to file

Description

Saving a ssrmlp model in onnx format to file (also see onnx.ai). This function uses the onnx python implementation, hence a python environment including modules onnx and numpy is required.

Usage

```
onnx_save(ssrmlp_model, filename)
```

Arguments

ssrmlp_model	a trained ssrmlp model
filename	fully qualified file name

Author(s)

Dr. Lars Metzner

Examples

```

# generate data
set.seed(42)
x <- rnorm(300)
y <- rnorm(300)
z <- rnorm(300) + atan2(x, y)
# coordinates
X <- matrix(cbind(x,y), ncol = 2)
Y <- as.double(z)
# Training
ssrmlp_model <- ssrmlp_train(X, Y)
# prediction
p <- t(c(0.25, 0.25))
pred <- ssrmlp_predict(p, ssrmlp_model)
# only if python is available
if (reticulate::py_module_available('onnx')) {
  # save in ONNX format
  onnx_save(ssrmlp_model, 'file.onnx')
  # load and run with TensorFlow
  onnx <- reticulate::import("onnx")
  backend <- reticulate::import("onnx_tf.backend")
  model = onnx$load('file.onnx')
  tf_rep = backend$prepare(model)
  tf_y = tf_rep$run(p)
  print(tf_y$Y-pred)
  # cleanup
  file.remove('file.onnx')
  # to avoid NOTE in R CHECK
  tempfile <- reticulate::import("tempfile")
  tmp <- tempfile$gettempdir()
  if (dir.exists(file.path(tmp, "__pycache__"))) {
    unlink(file.path(tmp, "__pycache__"), recursive = TRUE, force = TRUE)
  }
  tmp_py_files <- list.files(tmp,
                             pattern = "^__autograph_generated_file.*py$", full.names = TRUE)
  file.remove(tmp_py_files)
}

```

psplot

*Partial Sum Plot***Description**

Plots the Partial Sums with their quantiles for a given set of observations und the corresponding regression function.

Usage

```
psplot(dat, mu, text = 'Sample')
```

Arguments

dat	observations.
mu	regression function.
text	title of the chart.

Value

No explicit return value: a plot is generated

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
psplot(sin(seq(-pi, pi, length.out = 200))+rnorm(200),  
       sin(seq(-pi, pi, length.out = 200)), text='Test')
```

psplot3d

Partial Sum Plot for 2-dimensional coordinates

Description

Plots the partial sum statistic for the 3-dimensional SSR model

Usage

```
psplot3d(koord, z, mu, text = "Sample")
```

Arguments

koord	data frame with coordinates.
z	vector of observations.
mu	vector of discrete regression function.
text	optional: title for the plot.

Value

No explicit return value: a plot is generated

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(1234)
x <- rnorm(900)
y <- rnorm(900)
xy <- data.frame(x=x, y=y)
z <- rnorm(900) + atan2(x, y)
# Training
df_model <- ssr3d(xy, z, k = 4, fn = 8)
# plot partial sum statistic
psplot3d(xy, z, df_model$mu, 'ssr3d')
```

psplotnd

Partial Sum Plot for the multidimensional coordinates

Description

plots the partial sum statistic for the general n-dimensional SSR-model

Usage

```
psplotnd(koord, dat, mu, text = "Sample")
```

Arguments

koord	data frame with coordinates.
dat	data frame of observations.
mu	list of discrete regression function.
text	optional: title for the plot.

Value

No explicit return value: a plot is generated

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(1234)
x <- rnorm(900)
y <- rnorm(900)
xy <- data.frame(x=x, y=y)
z <- data.frame(z=rnorm(900) + atan2(x, y))
# Training
df_model <- ssrnd(xy, z, k = 4, fn = 8)
# plot partial sum statistic
psplotnd(xy, z, df_model$mu, 'ssr3d')
```

psvalid

Partial Sum Validity Check

Description

Checks, if a given regression function is adequate with respect to the partial sum criterium.

Usage

```
psvalid(dat,mu)
```

Arguments

dat	obervations.
mu	discrete regression function.

Value

valid	function is valid?
-------	--------------------

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
psvalid(sin(seq(-pi, pi, length.out = 200))+rnorm(200),
        sin(seq(-pi, pi, length.out = 200)))
```

runvalid	<i>Maximum Run Validity Check</i>
----------	-----------------------------------

Description

Checks, if a given regression function is adequate with respect to the maximum run criterium.

Usage

```
runvalid(dat, mu, k=NULL)
```

Arguments

dat	obervations.
mu	discrete regression function.
k	optional: maximum run length.

Value

valid	function is valid?
-------	--------------------

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
runvalid(sin(seq(-pi, pi, length.out = 200))+rnorm(200)/2,  
         sin(seq(-pi, pi, length.out = 200)))
```

snarch	<i>S-NARCH Model</i>
--------	----------------------

Description

Calculates the long-, middle- and short-term trends and vola for a financial time series.

Usage

```
snarch(dat)
```

Arguments

dat financial time series.

Value

data frame

tr20 long-term trend

v120 long-term vola

tr10 middle-term trend

v110 middle-term vola

tr5 short-term trend

v15 short-term vola

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2019) *Finanzmathematische Zeitreihenanalyse*. Independently Published.

Examples

```
# generate test data
set.seed(1234)
x <- seq(1:250)
dat <- 13000 + cumsum(rnorm(250))
# calculate the S-NARCH model
df <- snarch(dat)
# plot the results
op <- par(mfrow=c(1,3))
plot(x,dat)
lines(x,df$tr20)
lines(x,df$tr20 - df$v120, lty = 'dotted')
lines(x,df$tr20 + df$v120, lty = 'dotted')
plot(x,dat)
lines(x,df$tr10)
lines(x,df$tr10 - df$v110, lty = 'dotted')
lines(x,df$tr10 + df$v110, lty = 'dotted')
plot(x,dat)
lines(x,df$tr5)
lines(x,df$tr5 - df$v15, lty = 'dotted')
lines(x,df$tr5 + df$v15, lty = 'dotted')
par(op)
```

ssr *Onedimensional SSR-model calculation*

Description

Calculates L1- and L2-functions satisfying the partial sum criterium.

Usage

```
ssr(df, y1=NULL, yn=NULL, fn=0, iter=10000,  
    minStat=FALSE, ne=TRUE, l1=TRUE, ps=TRUE)
```

Arguments

df	data frame with two-dimensional data.
y1	optional: fixed value left.
yn	optional: fixed value right.
fn	optional: partial-sum-quantile (standard: generic calculation from data).
iter	optional: maximum number of iterations.
minStat	optional: boolean value for the minimum statistic.
ne	optional: boolean value for non-equidistant observations.
l1	optional: boolean value for function type.
ps	optional: sign criterium (partial sum or run).

Value

mu	SSR-function as array.
----	------------------------

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate equidistant data  
set.seed(1234)  
x <- seq(0, 2*pi, length.out = 200)  
y <- 4*sin(x) + rnorm(200)  
df <- data.frame(x=x, y=y)  
# calculate regression functions  
l1 <- ssr(df, ne=FALSE, ps=FALSE)  
l2 <- ssr(df, ne=FALSE, l1=FALSE)
```

```

lmin <- ssr(df, ne=FALSE, minStat=TRUE, ps=FALSE)
# plot results
plot(x, y, main = 'Sign-Simplicity-Regression',
      xlab = 't', ylab = 'sin(t)+noise')
lines(x, l1, col = 'blue')
lines(x, l2, col = 'red')
lines(x, lmin, col = 'purple')
legend("topleft", inset=c(0.01,0.01),
      legend=c("L1 run-crit.", "L2 ps-crit.", "L1 min-stat."),
      col=c("blue", "red", "purple"), lty=1:1)

# generate nonequidistant data
df <- data.frame(x=runif(500, min=-1, max=1)*pi)
df$y <- sin(df$x)*20 + rnorm(nrow(df), mean=0, sd=10)
# calculate regression function
df11 <- ssr(df, fn = 5)
# plot results
plot(df)
lines(df11, col = 'red')

```

ssr3d

3-dimensional SSR model

Description

Calculates the regression function for the 3-dimensional SSR-model.

Usage

```
ssr3d(koord, dat, k = NULL, fn = NULL, iter = 1000)
```

Arguments

koord	data frame with 2-dimensional coordinates.
dat	vector with observations.
k	optional: maximum run length for the model.
fn	optional: quantile for partial sums.
iter	optional: number of iterations for the numeric solver.

Value

df data frame with the relevant model data.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(1234)
x <- rnorm(900)
y <- rnorm(900)
xy <- data.frame(x=x, y=y)
z <- rnorm(900) + atan2(x, y)
# Training
df_model <- ssr3d(xy, z)
```

ssr3d_predict	<i>3-dimensional SSR model prediction</i>
---------------	-------------------------------------------

Description

Calculates the prediction for a given 3-dimensional SSR model.

Usage

```
ssr3d_predict(df_model, xy, ms = FALSE)
```

Arguments

df_model	data frame with model coordinates.
xy	data frame with coordinates for prediction.
ms	optional: boolean value to use the minimal surface algorithm.

Value

z	array with predictions.
---	-------------------------

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(1234)
x <- rnorm(900)
y <- rnorm(900)
xy <- data.frame(x=x, y=y)
z <- rnorm(900) + atan2(x, y)
# Training
df_model <- ssr3d(xy, z)
# Prediction
xx <- c(c(0,1), c(-1,1), c(1,-1))
xx <- matrix(xx, ncol = 2)
yy <- ssr3d_predict(df_model, xx)
```

ssrmlp_predict

Prediction function for the ssrMLP

Description

Calculates the prediction for a given ssrMLP

Usage

```
ssrmlp_predict(X, W)
```

Arguments

X matrix of coordinates.
W the weight matrices from ssrmlp_train method.

Value

Yp array with predictions.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(42)
x <- rnorm(300)
y <- rnorm(300)
z <- rnorm(300) + atan2(x, y)
# coordinates
X <- matrix(cbind(x,y), ncol = 2)
Y <- as.double(z)
# Training
W <- ssrmlp_train(X, Y)
Yp <- ssrmlp_predict(X, W)
```

ssrmlp_train

2-layer MLP with partial sum optimization

Description

Calculates the weights of a 2-layer MLP with respect to the partial sums criterion

Usage

```
ssrmlp_train(X, Y, std=TRUE, opt='ps', h1 = NULL, W = NULL,
             k=10, fn=4, eta=0.75, maxIter=1000)
```

Arguments

X	matrix with n-dimensional coordinates.
Y	array with observations.
std	optional: standardizing values if TRUE.
opt	optional: optimizing function ('l2' or 'ps').
h1	optional: array tuple with number of perceptrons in each layer.
W	optional: previously calculates weights for refining the model.
k	optional: number of neighbors per quadrant.
fn	optional: quantile for partial sums.
eta	optional: constant factor of the gradient algorithm.
maxIter	optional: number of iterations for the numeric solver.

Value

W List with weight matrices.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(42)
x <- rnorm(300)
y <- rnorm(300)
z <- rnorm(300) + atan2(x, y)
# coordinates
X <- matrix(cbind(x,y), ncol = 2)
Y <- as.double(z)
# Training
W <- ssrmlp_train(X, Y)
```

ssrnd

Multi-dimensional SSR model

Description

Calculates the multi-dimensional SSR model

Usage

```
ssrnd(koord, dat, k = NULL, fn = NULL, iter = 1000)
```

Arguments

<code>koord</code>	data frame with n-dimensional coordinates.
<code>dat</code>	data frame with observations.
<code>k</code>	optional: maximum run length for the model.
<code>fn</code>	optional: quantile for partial sums.
<code>iter</code>	optional: number of iterations for the numeric solver.

Value

`df` data frame with the relevant model data.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(1234)
x <- rnorm(300)
y <- rnorm(300)
xy <- data.frame(x=x, y=y)
z <- data.frame(z=rnorm(300) + atan2(x, y))
# Training
df_model <- ssrnd(xy, z)
```

ssrnd_predict

Prediction function for the multi-dimensional SSR model

Description

Calculates the prediction for a given multi-dimensional SSR model

Usage

```
ssrnd_predict(df_model, xx)
```

Arguments

df_model data frame with model coordinates.
xx data frame with coordinates for prediction.

Value

z list with predictions.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
# generate data
set.seed(1234)
x <- rnorm(300)
y <- rnorm(300)
xy <- data.frame(x=x, y=y)
z <- data.frame(z=rnorm(300) + atan2(x, y))
# Training
df_model <- ssrnd(xy, z)
```

```
# Prediction
xx <- c(c(0,1), c(-1,1), c(1,-1))
xx <- matrix(xx, ncol = 2)
yy <- ssrnd_predict(df_model, xx)
```

ssr_predict

SSR model Prediction

Description

Calculates the prediction for a given SSR model.

Usage

```
ssr_predict(df, xx)
```

Arguments

df dataframe containing two series with x- und y-values.
xx array containing locations for predictions.

Value

yy array containing the predicted values.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2021) *Adäquates Maschinelles Lernen*. Independently Published.

Examples

```
set.seed(1234)
df <- data.frame(x=runif(500, min=-1, max=1)*pi)
df$y <- sin(df$x)*20 + rnorm(nrow(df), mean=0, sd=10)
plot(df, xlim=c(-4, 4))
df11 <- ssr(df)
lines(df11)
xx <- c(-4, -1, 0, 1, 4)
yy <- ssr_predict(df11, xx)
points(xx,yy, pch='+', col='red', cex=2)
```

tauM	<i>Trend-based Correlation</i>
------	--------------------------------

Description

Calculates the trend-based correlation of two time series based on the trend function (Metzner's Tau)

Usage

```
tauM(x, y)
```

Arguments

x	time series.
y	time series.

Value

trend-based correlation.

Author(s)

Dr. Lars Metzner

References

Dr. Lars Metzner (2020) *Trendbasierte Prognostik*. Independently Published.

Examples

```
set.seed(1234)
s <- seq(-pi, pi, length.out = 200)
x <- s + rnorm(200)
y <- exp(s) + 5*rnorm(length(s))
op <- par(mfrow=c(1,2))
plot(x)
plot(y)
par(op)

p <- cor(x,y) # 0.5037
t <- cor(x,y, method = 'kendall') # 0.2959
tm <- tauM(x, y) # 0.0867
```

Index

* models

- fii_model, 5
- fii_prediction, 6
- onnx_load, 7
- onnx_save, 8

* model

- axe, 2
- axe_narch_model, 3
- axe_narch_predict, 4
- psplot, 9
- psplot3d, 10
- psplotnd, 11
- psvalid, 12
- runvalid, 13
- snarch, 13
- ssr, 15
- ssr3d, 16
- ssr3d_predict, 17
- ssr_predict, 22
- ssrmlp_predict, 18
- ssrmlp_train, 19
- ssrnd, 20
- ssrnd_predict, 21
- tauM, 23

* nonparametric

- axe, 2
- axe_narch_model, 3
- axe_narch_predict, 4
- fii_model, 5
- fii_prediction, 6
- onnx_load, 7
- onnx_save, 8
- psplot, 9
- psplot3d, 10
- psplotnd, 11
- psvalid, 12
- runvalid, 13
- snarch, 13
- ssr, 15

- ssr3d, 16

- ssr3d_predict, 17
- ssr_predict, 22
- ssrmlp_predict, 18
- ssrmlp_train, 19
- ssrnd, 20
- ssrnd_predict, 21
- tauM, 23

* regression

- axe_narch_model, 3
- axe_narch_predict, 4
- psplot, 9
- psplot3d, 10
- psplotnd, 11
- psvalid, 12
- runvalid, 13
- ssr, 15
- ssr3d, 16
- ssr3d_predict, 17
- ssr_predict, 22
- ssrmlp_predict, 18
- ssrmlp_train, 19
- ssrnd, 20
- ssrnd_predict, 21

* ts

- axe, 2
- fii_model, 5
- fii_prediction, 6
- onnx_load, 7
- onnx_save, 8
- snarch, 13
- tauM, 23

- axe, 2
- axe_narch_model, 3
- axe_narch_predict, 4

- fii_model, 5
- fii_prediction, 6

onnx_load, [7](#)
onnx_save, [8](#)

psplot, [9](#)
psplot3d, [10](#)
psplotnd, [11](#)
psvalid, [12](#)

runvalid, [13](#)

snarch, [13](#)
ssr, [15](#)
ssr3d, [16](#)
ssr3d_predict, [17](#)
ssr_predict, [22](#)
ssrmlp_predict, [18](#)
ssrmlp_train, [19](#)
ssrnd, [20](#)
ssrnd_predict, [21](#)

tauM, [23](#)