Package: NNbenchmark (via r-universe)

August 24, 2024

Title Datasets and Functions to Benchmark Neural Network Packages

Description Datasets and functions to benchmark (convergence, speed, ease of use) R packages dedicated to regression with neural networks (no classification in this version). The templates for the tested packages are available in the R, R Markdown and HTML formats at <https://github.com/pkR-pkR/NNbenchmarkTemplates> and <https://theairbend3r.github.io/NNbenchmarkWeb/index.html>. The submitted article to the R-Journal can be read at <https://www.inmodelia.com/gsoc2020.html>.

Depends R (>= 3.5.0)

Imports R6, pkgload, parallel

Suggests brnn, validann

Version 3.2.1

Author Patrice Kiener [aut, cre] (<https://orcid.org/0000-0002-0505-9920>), Christophe Dutang [aut] (<https://orcid.org/0000-0001-6732-1501>), Salsabila Mahdi [aut] (<https://orcid.org/0000-0002-2559-4154>), Akshaj Verma [aut] (<https://orcid.org/0000-0002-3936-0033>), Yifu Yan [ctb]

Maintainer Patrice Kiener <rpackages@inmodelia.com>

URL https://github.com/pkR-pkR/NNbenchmark

License GPL-2

Encoding UTF-8

LazyData true

NeedsCompilation no

RoxygenNote 7.2.3

Roxygen list(old_usage = TRUE)

Repository https://pkr-pkr.r-universe.dev

RemoteUrl https://github.com/pkr-pkr/nnbenchmark

RemoteRef HEAD

RemoteSha 504d48eaeae3d35025e7e2d42a607e69371bc51a

Contents

`NNbenchmark-package` *Package NNbenchmark*

Description

Datasets and functions to benchmark (convergence, speed, ease of use) R packages dedicated to regression with neural networks (no classification in this version). The templates for the tested packages are available at <https://github.com/pkR-pkR/NNbenchmarkTemplates> and <https://theairbend3r.github.io/NNbenchm
The submitted article to the R-Journal can be read at <https://www.inmodelia.com/gsoc2020.html>.

Examples

```
ds <- grep("^[m,u]", ls("package:NNbenchmark"), value = TRUE); ds
t(sapply(ds, function(x) dim(get(x))))
NNdataSummary(NNdatasets)
plot(uGauss2)
pairs(mIshigami)
ht(NNdatasets, n = 2, l = 6)
```

`add2csv` *Create or Append a data.frame to a csv File*

Description

Create or append a data.frame to a csv file. Column names are added at creation and ignored at the append steps.

Usage

```
add2csv(x, file = "results.csv", dir = ".")
```

Arguments

`x`	a data.frame or a matrix.
`file`	character. The filename.
`dir`	character. The directory in which the file is written. Default value `"."` is the current directory.

Value

Nothing in the console. A csv file on the disk.

Examples

```
results_csv <- tempfile("results", fileext = ".csv")
x <- data.frame(a = 1:3, b = 4:6)
add2csv(x, file = results_csv)
add2csv(x*10, file = results_csv)
add2csv(x*100, file = results_csv)
read.csv(file = results_csv)
```

`bWoodN1` *Dataset bWoodN1*

Description

A multivariate dataset (x1, x2, x3, x4, x5, x6, y) of class data.frame and dim 20000 x 7 to be fitted by a neural network with 5 hidden neurons (41 parameters).

References

Inspired by page 29 of Wood, S. N. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(1), 3-36. https://people.bath.ac.uk/man54/SAMBa/ITTs/ITT2/EDF/REMLWood2009.pdf

Examples

```
ht(bWoodN1)
```

`cc` *Concatenates List and Vectors into a List*

Description

An intermediate function between `c()` and `list()`. Combine all terms in one single list. The result can be used by `do.call()`.

Usage

```
cc(char, ...)
```

Arguments

`char`	a vector of named objects, except a list.
`...`	basic R objects: character, vectors, list, data.frame.

Value

A list with the objects concatenated.

Examples

```
lst <- list(yaxt = "s", side = 2, col = 1:3) ; lst
dfr <- data.frame(x = 5:9, y = 10:14) ; dfr

## With c(), the list is returned at its given position
c(lst, at = 7, labels = c("0", "0.5", "1"), dfr = dfr)
c(at = 7, labels = c("0", "0.5", "1"), lst, dfr = dfr)

## With cc(), the unnamed list is always returned in first position
cc(lst, at = 7, labels = c("0", "0.5", "1"), dfr = dfr)
cc(at = 7, dfr = dfr, labels = c("0", "0.5", "1"), lst)

## Some similarities and differences between c() and cc()
c( 1:5, y = 2:6, col = 2, lwd = 2)
cc(1:5, y = 2:6, col = 2, lwd = 2)

c( x = 1:5, y = 2:6, col = 2, lwd = 2)
cc(x = 1:5, y = 2:6, col = 2, lwd = 2)

## Regular function and do.call()
plot(x = 1:5, y = 2:6, col = 2, lwd = 2)
do.call( plot , cc(x = 1:5, y = 2:6, col = 3, lwd = 3, cex = 3))
do.call("plot", cc(x = 1:5, y = 2:6, col = 4, lwd = 4, cex = 4))
```

---

clearNN *Detach the Loaded Packages and the ZZ object*

Description

`clearNN` detachs `ZZ` and the packages loaded for the evaluation.

`detachNN` detachs `ZZ`.

Usage

```
clearNN(donotremove, donotdetach = NULL)

detachNN()
```

Arguments

`donotremove`	a vector of characters representing objects in `ls()`.
`donotdetach`	a vector of packages and environments that are not detached. `donotdetach = NULL` protects the packages and environments `".GlobalEnv"`, `"package:kableExtra"`, `"package:dplyr"`, `"package:stringr"`, `"package:NNbenchmark"`, `"package:rmarkdown"`, `"package:knitr"`, `"package:captioner"`, `"package:pkgload"`, `"package:R6"`, `"tools:rstudio"`, `"package:RWsearch"`, `"package:pacman"`, `"package:stats"`, `"package:graphics"`, `"package:grDevices"`, `"package:utils"`, `"package:datasets"`, `"package:methods"`, `"Autoloads"`, `"package:base"`.

createTimer — *Create a timer object. Get the data frame of a timer object*

Description

This is a modified version of the `timeR` package for an internal use. Full credit is to Yifu Yan, the author of the `timeR` package.

`createTimer` creates a timer object.

`getTimer` returns a data frame with all records saved by the timer object. Columns in the data.frame are: event, start, end, duration, RMSE, MAE, stars, params, comment.

Usage

```
createTimer(verbose = TRUE)

getTimer(object)
```

Arguments

`verbose`	A parameter to control whether to print messages while using methods. Default to `TRUE`.
`object`	The name for timer object.

Value

An (invisible) object of R6 class for `createTimer`. A data.frame for `getTimer`.

Examples

```
## Create a timer object. Record events. Get all records.
timeTT <- createTimer(FALSE) # print is disabled
timeTT <- createTimer()      # print is enabled
timeTT$start("event1")
Sys.sleep(1)
timeTT$stop("event1", RMSE = 1, MAE = 1.3, stars = "*",
            params = "maxiter=100, lr=0.01", comment = "OK for 1",
            printmsg = TRUE)
timeTT$start("event2")
Sys.sleep(2)
timeTT$stop("event2", RMSE = 2, MAE = 2.6, stars = "**",
            params = "maxiter=1000, lr=0.001", comment = "OK for 2",
            printmsg = FALSE)
getTimer(timeTT)
```

`funRMSE` *Calculate the RMSE, MSE, MAE, and WAE Rounded to 4 digits*

Description

Calculate the Root Mean Squared Error (RMSE), the Mean Squared Error (MSE), the Mean Absoluter Error (MAE), and the Worst Absolute Error (WAE). The result is rounded to 4 digits by default. Apply `na.rm = TRUE`

Usage

```
funRMSE(y_pred, y0, dgts = 4)

funMSE(y_pred, y0, dgts = 4)

funMAE(y_pred, y0, dgts = 4)

funWAE(y_pred, y0, dgts = 4)
```

Arguments

`y_pred`	numeric vector of the predicted values
`y0`	numeric vector of the observed values
`dgts`	integer value for how many digits to round to

Value

A numeric value, either the RMSE, MSE, MAE, or WAE.

Examples

```
y0 <- 1:19
y_pred <- y0 + rnorm(length(y0), sd = 0.3)
funRMSE(y_pred, y0)
funMSE( y_pred, y0)
funMAE( y_pred, y0)
funWAE( y_pred, y0)
```

`ht` *Concatenates head() and tail() in vector, list, matrix, data.frame, array*

Description

Concatenates `head(n)` and `tail(n)` rows and subset with m columns. Works also with array, list and matrix in data.frame. Keeps the data.table format (and add a timezone by default). See the matsindf package for matrix(ces) in tibble.

Usage

```
ht(x, n = 3, m = 4, p = 2, l = 2, names = TRUE, LTT = c("x", "L",
  "N", "M", "P", "Q", "R", "S", "T"))

ht9(x, n = 3, m = 9999, p = 2, l = 2, names = TRUE, LTT = c("x",
  "L", "N", "M", "P", "Q", "R", "S", "T"))
```

Arguments

`x`	vector, matrix, data.frame, array or list.
`n`	integer. Cut in the first dimension.
`m`	integer. Cut in the second dimension.
`p`	integer. Cut in the third and the next dimensions.
`l`	integer. Cut for a list and for data.frame with matrix inside.
`names`	logical. Provide names and numbers for undefined dimnames.
`LTT`	character. The letter used in each dimension (vector, list, array).

Value

An object of the same class than x but much shorter.

Examples

```
### Vector, data.frame, array
ht(1:100, names = FALSE)
ht(1:100, LTT = "z")

ht9(mtcars); dim(mtcars)

arr4 <- array(1:1680, c(8,7,6,5))
ht(arr4, n=1, p=1, names = FALSE)
ht(arr4, n=1, p=1, names = TRUE, LTT = c("x","L","X","Y","Z","T"))

### List of matrices
lstmat <- rep(list(matrix(1:100, 10)), 8)
for (i in seq_along(lstmat)) lstmat[[i]] <- lstmat[[i]] *i
lstmat
```

```
ht(lstmat, n = 2, m = 2, l = 2, names = FALSE)
ht(lstmat, n = 2, m = 3, l = 1, names = TRUE)
ht(lstmat, n = 2, m = 3, l = 1, LTT = c("x","L","X","Y","Z"))

### Data.frame with matrices inside.
### See For instance data("gasoline", package = "pls")

## Colnames on matrix B but not on matrix C. Protected data.frame.
B <- matrix(101:160, 10); colnames(B) <- paste0(1:6, "b"); B
C <- matrix(101:160, 10); C
dfrmat <- data.frame(A = 1:10, B = I(B), C = I(C), D = 11:20); dfrmat
colnames(dfrmat)

## Matrix columns are controlled by m.
## Unnamed C matrix columns have old values but new names. Be aware!
ht(dfrmat, n = 2, m = 1, l = 2, names = FALSE) # Original C.6 is now C.2
ht(dfrmat, n = 2, m = 1, l = 2, names = TRUE)  # Names keep original ranks

## Data.frame columns are controlled by l.
ht(dfrmat, n = 3, m = 2, l = 1, names = TRUE)
```

`mDette` *Dataset mDette*

Description

A multivariate dataset (x1, x2, x3, y) of class matrix and dim 500 x 4 to be fitted by a neural network with 5 hidden neurons (26 parameters).

References

Dette, H., & Pepelyshev, A. (2010). Generalized Latin hypercube design for computer experiments. Technometrics, 52(4).

See also `https://www.sfu.ca/~ssurjano/detpep10curv.html`

Examples

```
ht(mDette)
pairs(mDette)
```

mFriedman *Dataset mFriedman*

Description

A multivariate dataset (x1, x2, x3, x4, x5, y) of class matrix and dim 500 x 6 to be fitted by a neural network with 5 hidden neurons (36 parameters).

References

Friedman, J. H., Grosse, E., & Stuetzle, W. (1983). Multidimensional additive spline approximation. SIAM Journal on Scientific and Statistical Computing, 4(2), 291-301.

See also `https://www.sfu.ca/~ssurjano/fried.html`

Examples

```
ht(mFriedman)
pairs(mFriedman)
```

mIshigami *Dataset mIshigami*

Description

A multivariate dataset (x1, x2, x3, y) of class matrix and dim 500 x 4 to be fitted by a neural network with 10 hidden neurons (51 parameters).

References

Ishigami, T., & Homma, T. (1990, December). An importance quantification technique in uncertainty analysis for computer models. In Uncertainty Modeling and Analysis, 1990. Proceedings., First International Symposium on (pp. 398-403). IEEE.

See also `https://www.sfu.ca/~ssurjano/ishigami.html`

Examples

```
ht(mIshigami)
pairs(mIshigami)
```

mRef153 *Dataset mRef153*

Description

A multivariate dataset (x1, x2, x3, x4, x5, y) of class matrix and dim 153 x 6 to be fitted by a neural network with 3 hidden neurons (22 parameters). This dataset was used to teach neural networks at ESPCI from 2003 to 2013 and is available in the software Neuro One.

References

Neuro One `https://www.inmodelia.com/software.html`

Examples

```
ht(mRef153)
pairs(mRef153)
```

NNbigdatasets *Big Datasets in One list (2020)*

Description

`NNbigdatasets` is a list with the big datasets presented in this package and the recommended number of hidden neurons for each neural network model.

- bWoodN1: 5 neurons.

Each item of the list is itself a list with 5 components:

- ds: character. The name of the dataset.
- neur: integer. The recommanded number of hidden neurons in the NN model and in `fmlaNN`.
- nparNN: integer. The number of parameters in `fmlaNN`.
- fmlaNN: the formula of the corresponding neural network, with tanh() as the activation function in the hidden layer.
- Z: matrix or data.frame. The dataset itself.

Using `attach()` and `detach()` gives a direct access to these items.

Examples

```
ht(NNbigdatasets)

NNdataSummary(NNbigdatasets)
```

NNdatasets *All Datasets in One List*

Description

NNdatasets is a list with the 12 datasets presented in this package and the recommended number of hidden neurons for each neural network model.

- mDette: 5 neurons.
- mFriedman: 5 neurons.
- mIshigami: 10 neurons.
- mRef153: 3 neurons.
- uDmod1: 6 neurons.
- uDmod2: 5 neurons.
- uDreyfus1: 3 neurons.
- uDreyfus2: 3 neurons.
- uGauss1: 5 neurons.
- uGauss2: 4 neurons.
- uGauss3: 4 neurons.
- uNeuroOne: 2 neurons.

Each item of the list is itself a list with 5 components:

- ds: character. The name of the dataset.
- neur: integer. The recommanded number of hidden neurons in the NN model and in fmlaNN.
- nparNN: integer. The number of parameters in fmlaNN.
- fmlaNN: the formula of the corresponding neural network, with tanh() as the activation function in the hidden layer.
- Z: matrix or data.frame. The dataset itself.

Using attach() and detach() gives a direct access to these items.

Examples

```
ht(NNdatasets, n = 2, l = 6)

NNdataSummary(NNdatasets)
```

`NNdataSummary` *A Summary (in data.frame format) of NNdatasets*

Description

`NNdataSummary` summarizes the information of the 12 datasets listed in `NNdatasets`.

Usage

```
NNdataSummary(NNdatasets)
```

Arguments

`NNdatasets`	the NNdatasets list.

Value

A data.frame with 12 rows and 5 columns: (dataset) name, n_rows, n_inputs, n_neurons, n_parameters.

Examples

```
NNdataSummary(NNdatasets)
```

`NNsummary` *Summarize Calculations of RMSE, MSE, MAE, and WAE*

Description

Summarize measures of fit and time for a single training. Measures of fit include the Root Mean Squared Error (RMSE), the Mean Squared Error (MSE), the Mean Absolute Error (MAE), and the Worst Absolute Error (WAE) rounded by default to 4 digits and set to `na.rm = TRUE`. See more at `funRMSE`. The summary can also include the results of time from `getTimer` in NNbenchmark or the result of `timediff`.

Usage

```
NNsummary(y_pred, y0, time, dgts = 4)
```

Arguments

`y_pred`	numeric vector of the predicted values
`y0`	numeric vector of the observed values
`time`	numeric value of time
`dgts`	integer value for how many digits to round to, see `round`

Value

A vector of RMSE, MSE, MAE, WAE, and time values for a single iteration.

Examples

```
## With 2019 legacy code, no longer usable with 2020 trainPredict
old <- options("digits.secs" = 4)
timeTT <- createTimer()

timeTT$start("event")
y0 <- 1:19
y_pred <- y0 + rnorm(length(y0), sd = 0.3)
timeTT$stop("event")

time <- getTimer(timeTT)

NNsummary(y_pred, y0, time[,4], 4)

## With 2020 code
timestart()
y0 <- 1:19
y_pred <- y0 + rnorm(length(y0), sd = 0.3)
time <- timediff()

NNsummary(y_pred, y0, time, 4)

options(old)
```

NNtrainPredict *Generic Functions for Training and Predicting*

Description

An implementation with `do.call` so that any neural network function that fits the format can be tested.

In `trainPredict_1mth1data`, a neural network is trained on one dataset and then used for predictions, with several functionalities. Then, the performance of the neural network is summarized.

`trainPredict_1data` serves as a wrapper function for `trainPredict_1mth1data` for multiple methods.

`trainPredict_1pkg` serves as a wrapper function for `trainPredict_1mth1data` for multiple datasets.

Usage

```
trainPredict_1mth1data(dset, method, trainFUN, hyperparamFUN, predictFUN,
  summaryFUN, prepareZZ.arg = list(), nrep = 5, doplot = FALSE,
  plot.arg = list(col1 = 1:nrep, lwd1 = 1, col2 = 4, lwd2 = 3), pkgname,
  pkgfun, csvfile = FALSE, rdafile = FALSE, odir = ".", echo = FALSE,
  echoreport = FALSE, appendcsv = TRUE, ...)

trainPredict_1data(dset, methodlist, trainFUN, hyperparamFUN, predictFUN,
  summaryFUN, closeFUN, startNN = NA, prepareZZ.arg = list(), nrep = 5,
  doplot = FALSE, plot.arg = list(), pkgname = "pkg", pkgfun = "train",
  csvfile = FALSE, rdafile = FALSE, odir = ".", echo = FALSE,
  lib.loc = NULL, ...)

trainPredict_1pkg(dsetnum, pkgname = "pkg", pkgfun = "train", methodvect,
  prepareZZ.arg = list(), summaryFUN, nrep = 5, doplot = FALSE,
  plot.arg = list(), csvfile = FALSE, rdafile = FALSE, odir = ".",
  echo = FALSE, appendcsv = TRUE, parallel = "no", ncpus = 1,
  cl = NULL, lib.loc = NULL, ...)
```

Arguments

`dset`	a number or string indicating which dataset to use, see `NNdataSummary`
`method`	a method for a particular function
`trainFUN`	the training function used
`hyperparamFUN`	the function resulting in parameters needed for training
`predictFUN`	the prediction function used
`summaryFUN`	measure performance by observed and predicted y values, `NNsummary` is ready to use
`prepareZZ.arg`	list of arguments for `prepareZZ`
`nrep`	a number for how many times a neural network should be trained with a package/function
`doplot`	logical value, TRUE executes plots and FALSE does not
`plot.arg`	list of arguments for plots
`pkgname`	package name
`pkgfun`	name of the package function to train neural network
`csvfile`	logical value, adds summary to csv files per dataset if TRUE
`rdafile`	logical value, outputs rdafile of predictions and summary if TRUE
`odir`	output directory
`echo`	logical value, separates training between packages with some text and enables echoreport if TRUE
`echoreport`	logical value, detailed reports are printed (such as model summaries and str(data)) if TRUE, will not work if echo is FALSE
`appendcsv`	logical value, if `TRUE`, the csv output is appended to the csv file.
`...`	additional arguments
`methodlist`	list of methods per package/function
`closeFUN`	a function to detach packages or other necessary environment clearing

`startNN`	a function to start needed outside libraries, for example, h2o
`lib.loc`	A character vector describing the location of R library trees to search through, or `NULL`. The default value of `NULL` corresponds to all libraries currently known to `.libPaths()`. Non-existent library trees are silently ignored.
`dsetnum`	a vector of numbers indicating which dataset to use in `NNdataSummary`
`methodvect`	vector of methods per package/function
`parallel`	The type of parallel operation to be used (if any). If missing, the default is `"no"`.
`ncpus`	integer: number of processes to be used in parallel operation: typically one would chose this to the number of available CPUs.
`cl`	An optional parallel or snow cluster for use if `parallel = "snow"`. If not supplied, a cluster on the local machine is created for the duration of the call.

Value

An array with values as in NNsummary including each repetition, with options for plots and output files

Examples

```
nrep <- 2
odir <- tempdir()

### Package with one method/optimization algorithm
library("brnn")
brnn.method <- "gaussNewton"
hyperParams.brnn <- function(optim_method, ...) {
  return(list(iter = 200))
  }
brnn.prepareZZ <- list(xdmv = "m", ydmv = "v", zdm = "d", scale = TRUE)

NNtrain.brnn   <- function(x, y, dataxy, formula, neur, optim_method, hyperParams,...) {
  hyper_params <- do.call(hyperParams.brnn, list(brnn.method))
  iter  <- hyper_params$iter

  NNreg <- brnn::brnn(x, y, neur, normalize = FALSE, epochs = iter, verbose = FALSE)
  return(NNreg)
  }
NNpredict.brnn <- function(object, x, ...) { predict(object, x) }
NNclose.brnn <- function(){
  if("package:brnn" %in% search())
    detach("package:brnn", unload=TRUE)
  }


res <- trainPredict_1pkg(1:2, pkgname = "brnn", pkgfun = "brnn", brnn.method,
                         prepareZZ.arg = brnn.prepareZZ, nrep = nrep, doplot = TRUE,
                         csvfile = FALSE, rdafile = FALSE, odir = odir, echo = FALSE)

### Package with more than one method/optimization algorithm
library(validann)
```

```
validann.method <- c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN")
hyperParams.validann <- function(optim_method, ...) {
  if(optim_method == "Nelder-Mead")  { maxiter <- 10000 }
  if(optim_method == "BFGS")         { maxiter <- 200   }
  if(optim_method == "CG")           { maxiter <- 1000  }
  if(optim_method == "L-BFGS-B")     { maxiter <- 200   }
  if(optim_method == "SANN")         { maxiter <- 1000  }
  return(list(iter = maxiter, method = optim_method, params))
  }
validann.prepareZZ <- list(xdmv = "m", ydmv = "m", zdm = "d", scale = TRUE)

NNtrain.validann <- function(x, y, dataxy, formula, neur, optim_method, hyperParams, ...) {
  hyper_params <- do.call(hyperParams, list(optim_method, ...))
  iter <- hyper_params$iter
  method <- hyper_params$method

  NNreg <- validann::ann(x, y, size = neur, method = method, maxit = iter)
  return (NNreg)
  }
NNpredict.validann <- function(object, x, ...) { predict(object, x) }
NNclose.validann <- function() {
  if("package:validann" %in% search())
  detach("package:validann", unload=TRUE)
  }

res <- trainPredict_1pkg(1:2, pkgname = "validann", pkgfun = "ann", validann.method,
                         repareZZ.arg = validann.prepareZZ, nrep = nrep, doplot = FALSE,
                         csvfile = TRUE, rdafile = TRUE, odir = odir, echo = FALSE)
```

---

plotNN *Create a Plot, Add Lines or Points Depending The Context*

Description

`plotNN` uses the parameter `uni` to launch the `plot()` function either for a univariate dataset (x, y_pred) or for a multivariate dataset (y, y_pred).

`lipoNN` uses the parameter `uni` to launch either `lines()` for an univariate dataset or `points()` for a multivariate dataset.

See the examples in `prepareZZ`.

Usage

```
plotNN(xory, y0, uni, TF = TRUE, ...)

lipoNN(xory, y_pred, uni, TF = TRUE, ...)
```

Arguments

`xory`	vector of numeric. The original x values for an univariate dataset or the original y values for a multivariate dataset.
`y0`	vector of numeric. The original y values.
`uni`	logicial. TRUE for an univariate dataset. FALSE for a multivariate dataset. ... fails or if you call the function from another function.
`TF`	logical. TRUE executes the instruction. FALSE ignores the instruction. Equivalent to `if (TRUE/FALSE) plot()`.
`...`	parameters passed to `plot()`, `lines()` or `points()`.
`y_pred`	vector of numeric. The values returned by the `predict()` function.

Value

NULL in the console. An initial plot or some added lines/points.

`prepareZZ` *Prepare a Dataset For All Possible Formats*

Description

This function modifies a dataset to the format required by a training function: data.frame, matrix or vector (numeric), pre-normalization.

Usage

```
prepareZZ(Z, xdmv = "m", ydmv = "v", zdm = "d", scale = FALSE)
```

Arguments

`Z`	a matrix or a data.frame representing a dataset.
`xdmv`	character, either "d", "m" or "v". The prefered output format for x: data.frame, matrix, vector (numeric).
`ydmv`	character, either "d", "m" or "v". The prefered output format for y: data.frame, matrix, vector (numeric).
`zdm`	character, either "d" or "m". The prefered output format for Zxy: data.frame or matrix.
`scale`	logical. Scale x, y and Zxy with their respective means and standard deviations.

Value

The output is a list, usually named ZZ, with the following items:

- Zxy: the original or scaled Z in the desired format (data.frame, matrix).
- x: the original or scaled x in the desired format (data.frame, matrix, vector).
- y: the original or scaled y in the desired format (data.frame, matrix, vector).
- xory: the original x or y (as vector).
- y0: the original y (as vector).
- xm0: the mean(s) of the original x.
- ym0: the mean of the original y.
- xsd0: the standard deviation(s) of the original x.
- ysd0: the standard deviation of the original y.
- uni: the univariate (TRUE) or multivariate (FALSE) status of x (Z).
- fmla: the formula `y ~ x` or `y ~ x1 + x2 + .. + xn` where n is the number of inputs variables.

The use of `attach()` and `detach()` gives direct access to the modified values of ZZ. See the examples.

Examples

```
library("brnn")
library("validann")

maxit <- 200  # increase this number to get more accurate results with validann:ann
TF    <- TRUE # display the plots

### UNIVARIATE DATASET
Z     <- uGauss2
neur  <- 4

## brnn
ZZ <- prepareZZ(Z, xdmv = "m", ydmv= "v", scale = FALSE) ; ht(ZZ)
attach(ZZ)
y_pred <- ym0 + ysd0*predict(brnn(x, y, neur))
plotNN(xory, y0, uni, TF)
lipoNN(xory, y_pred, uni, TF, lwd = 4, col = 2)
ym0 ; ysd0
detach(ZZ) ; rm(y_pred)


## validann
ZZ <- prepareZZ(Z, xdmv = "m", ydmv= "v", scale = TRUE) ; ht(ZZ)
attach(ZZ)
y_pred <- ym0 + ysd0*predict(validann::ann(x, y, neur, maxit = maxit))
lipoNN(xory, y_pred, uni, TF, lwd = 4, col = 3)
ym0 ; ysd0
detach(ZZ) ; rm(y_pred)
```

```
### UNIVARIATE DATASET + LOOP
nruns  <- 10

## brnn
ZZ <- prepareZZ(Z, xdmv = "m", ydmv= "v", scale = FALSE) ; ht(ZZ)
attach(ZZ)
Zreg <- list() ; Zreg
for (i in 1:nruns) Zreg[[i]] <- brnn::brnn(x, y, neur)
m      <- matrix(sapply(Zreg, function(x) x$Ed) , ncol=1) ; m
best   <- which(min(m) == m)[1] ; best
y_pred <- ym0 + ysd0*predict(Zreg[[best]])
plotNN(xory, y0, uni, TF)
lipoNN(xory, y_pred, uni, TF, lwd = 4, col = 2)
detach(ZZ) ; rm(y_pred)

## validann
ZZ <- prepareZZ(Z, xdmv = "m", ydmv= "v", scale = TRUE) ; ht(ZZ)
attach(ZZ)
Zreg <- list()
for (i in 1:nruns) Zreg[[i]] <- validann::ann(x, y, size = neur, maxit = maxit)
m      <- matrix(sapply(Zreg, function(x) x$value), ncol=1) ; m
best   <- which(min(m) == m)[1] ; best
y_pred <- ym0 + ysd0*predict(Zreg[[best]])
lipoNN(xory, y_pred, uni, TF, lwd = 4, col = 4)
detach(ZZ) ; rm(y_pred)


### MULTIVARIATE DATASET
Z    <- mDette
neur <- 5

## brnn
ZZ <- prepareZZ(Z, xdmv = "m", ydmv= "v", scale = FALSE) ; ht(ZZ)
attach(ZZ)
y_pred <- ym0 + ysd0*predict(brnn::brnn(x, y, neur))
plotNN(xory, y0, uni, TF)
lipoNN(xory, y_pred, uni, TF, lwd = 4, col = 2)
ym0 ; ysd0
detach(ZZ) ; rm(y_pred)

## validann
ZZ <- prepareZZ(Z, xdmv = "m", ydmv= "v", scale = TRUE) ; ht(ZZ)
attach(ZZ)
y_pred <- ym0 + ysd0*predict(validann::ann(x, y, neur, maxit = maxit))
lipoNN(xory, y_pred, uni, TF, lwd = 4, col = 3)
ym0 ; ysd0
detach(ZZ) ; rm(y_pred)


### INSIDE A FUNCTION
plotds <- function(Z, xdmv = "m", ydmv = "v", scale = FALSE, neurons = 3, col = 2) {
    ZZ <- prepareZZ(Z, xdmv = xdmv, ydmv= ydmv, scale = scale)
    attach(ZZ) ; on.exit(detach(ZZ))
    y_pred <- ym0 + ysd0*predict(brnn::brnn(x, y, neurons))
    plotNN(xory, y0, uni, TF)
    lipoNN(xory, y_pred, uni, TF, lwd = 4, col = col)
    print(ht(x))
    print(ht(y))
}
plotds(uNeuroOne, scale = FALSE, neurons = 2, col = 2)
plotds(uNeuroOne, scale = TRUE,  neurons = 3, col = 3)

plotds(mFriedman, scale = TRUE,  neurons = 5, col = 4)
```

`timeR` *A R6 Class to represent a timer.*

Description

timer is a R6 Class that represent a timer. This is a modified version of the `timeR` package for an internal use. Full credit is to Yifu Yan, the author of the `timeR` package.

Value

`getTimer` returns a data frame with all records saved by the timer object. Columns in the data.frame are: event, start, end, duration, RMSE, MAE, stars, params, comment.

Public Methods

`initialize(time,event,verbose,eventTable)` Initialize a timer object. You can also use `createTimer()` function to initialize a timer object.

`start(eventName)` Start timing for a event, `eventName` should be a string

`stop(eventName)` Stop timing for a event.

`getTimer()` Get/Print a data.frame with all records.

`removeEvent(eventName)` Remove an given row in the eventTable.

`toggleVerbose()` Toggle between `TRUE` and `FALSE` for `verbose`

`getStartTime()` Get start time for a selected event.

`getStopTime()` Get stop time for a selected event.

`getDuration()` Get duration for a selected event.

`getRMSE()` Get the RMSE for a selected event.

`getMAE()` Get the MAE for a selected event.

`getStars()` Get stars for a selected event.

`getParams()` Get params for a selected event.

`getComment()` Get comment for a selected event.

`getEventf()` Get entire row for a selected event.

`print()` Custom print method for timer class. However, you don't need to use this function to generate custom printing. Custom printing is triggered by default.

Private Methods

`slprint(msg, flag = self$verbose)` A function that controls whether to print extra message.

Public fields

`time` A POSIXct/POSIXlt value of your latest timing.

`event` A string of your latest timing.

`eventTable` A data frame that stores all timings.

`verbose` A printing setting that controls whether to print messages.

Active bindings

`time` A POSIXct/POSIXlt value of your latest timing.

`event` A string of your latest timing.

Methods

Public methods:

- `timeR$new()`
- `timeR$start()`
- `timeR$stop()`
- `timeR$getTimer()`
- `timeR$removeEvent()`
- `timeR$toggleVerbose()`
- `timeR$getStartTime()`
- `timeR$getStopTime()`
- `timeR$getDuration()`
- `timeR$getRMSE()`
- `timeR$getMAE()`
- `timeR$getStars()`
- `timeR$getParams()`
- `timeR$getComment()`
- `timeR$getEvent()`
- `timeR$print()`
- `timeR$clone()`

Method `new()`:

Usage:

```
timeR$new(verbose = TRUE)
```

Method `start()`:

Usage:

```
timeR$start(eventName)
```

Method `stop()`:

Usage:

```
timeR$stop(eventName, RMSE = NA_real_, MAE = NA_real_,
  stars = NA_character_, params = NA_character_, comment = NA_character_,
  printmsg = TRUE)
```

Method `getTimer()`:

Usage:

```
timeR$getTimer(...)
```

Method `removeEvent()`:

Usage:

```
timeR$removeEvent(eventName)
```

Method `toggleVerbose()`:

Usage:

```
timeR$toggleVerbose(...)
```

Method `getStartTime()`:

Usage:

```
timeR$getStartTime(eventName)
```

Method `getStopTime()`:

Usage:

```
timeR$getStopTime(eventName)
```

Method `getDuration()`:

Usage:

```
timeR$getDuration(eventName)
```

Method `getRMSE()`:

Usage:

```
timeR$getRMSE(eventName)
```

Method `getMAE()`:

Usage:

```
timeR$getMAE(eventName)
```

Method `getStars()`:

Usage:

```
timeR$getStars(eventName)
```

Method `getParams()`**:**

Usage:

```
timeR$getParams(eventName)
```

Method `getComment()`**:**

Usage:

```
timeR$getComment(eventName)
```

Method `getEvent()`**:**

Usage:

```
timeR$getEvent(eventName)
```

Method `print()`**:**

Usage:

```
timeR$print(...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
timeR$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
timer <- createTimer()
timer$start("event1")
## put some codes in between, for instance
Sys.sleep(1)
timer$stop("event1", RMSE = 1, MAE = 1.3, stars = "*",
           params = "maxiter=100, lr=0.01", comment = "OK for 1",
           printmsg = TRUE)

timer$start("event2")
## put some codes in between, for instance
Sys.sleep(2)
timer$stop("event2", RMSE = 2, MAE = 2.6, stars = "**",
           params = "maxiter=1000, lr=0.001", comment = "OK for 2",
           printmsg = FALSE)

table1 <- getTimer(timer)
timer$toggleVerbose() # set verbose to FALSE as default is TRUE
table1 # print all records in a data frame

## get attributes for selected events
timer$getStartTime("event1")
timer$getStopTime("event1")
timer$getDuration("event1")
```

```
timer$getComment("event1")
timer$getEvent("event1")
```

timestart *Collect the difftime between two events*

Description

`timestart` starts the timer and saved the value in an object named `time0` stored in `.GlobalEnv`.

`timediff` stops the timer, remove the `time0` objet from `.GlobalEnv` and prints the duration in seconds between the two events.

`timestart` and `timediff` are fully independant from the R6 class `timeR` and the objects `createTimer` or `getTimer`. They use `proc.time` instead.

Usage

```
timestart()

timediff()
```

Value

A single numeric value that represents a duration in seconds.

Examples

```
timestart()
Sys.sleep(2)
timediff()
```

uDmod1 *Dataset uDmod1*

Description

An univariate dataset (x, y) of class data.frame and dim 51 x 2 to be fitted by a neural network with 6 hidden neurons (19 parameters). The parameters are highly correlated and singular Jacobian matrices often appear. A difficult dataset.

Examples

```
ht(uDmod1)
plot(uDmod1)
```

uDmod2 *Dataset uDmod2*

Description

An univariate dataset (x, y) of class data.frame and dim 51 x 2 to be fitted by a neural network with 5 hidden neurons (16 parameters).

Examples

```
ht(uDmod2)
plot(uDmod2)
```

uDreyfus1 *Dataset uDreyfus1*

Description

An univariate dataset (x, y) of class data.frame and dim 51 x 2 to be fitted by a neural network with 3 hidden neurons (10 parameters). This dataset was used to teach neural networks at ESPCI from 1991 to 2013. It usually appeared in the very first slides. This is a combination of 3 pure tanh() functions without noise. The Jacobian matrix is singular at the target parameter values and many algorithms could fail.

References

Dreyfus, G., ESPCI `https://www.neurones.espci.fr`

Examples

```
ht(uDreyfus1)
plot(uDreyfus1)
```

uDreyfus2 *Dataset uDreyfus2*

Description

An univariate dataset (x, y) of class data.frame and dim 51 x 2 to be fitted by a neural network with 3 hidden neurons (10 parameters). This dataset was used to teach neural networks at ESPCI from 1991 to 2013. It usually appeared in the very first slides. This is a combination of 3 pure tanh() functions with a small noise. Due to the noise, the Jacobian matrix is not singular at the target parameter values. All algorithms should find the target parameter values.

References

Dreyfus, G., ESPCI https://www.neurones.espci.fr

Examples

```
ht(uDreyfus2)
plot(uDreyfus2)
```

---

uGauss1 *Dataset uGauss1*

Description

An univariate dataset (x, y) of class data.frame and dim 250 x 2 to be fitted by a neural network with 5 hidden neurons (16 parameters).

References

Rust, B., NIST (1996) https://www.inmodelia.com/gsoc2020-redirection.html with redirection to the (slow) NIST page.

Examples

```
ht(uGauss1)
plot(uGauss1)
```

---

uGauss2 *Dataset uGauss2*

Description

An univariate dataset (x, y) of class data.frame and dim 250 x 2 to be fitted by a neural network with 4 hidden neurons (13 parameters).

References

Rust, B., NIST (1996) https://www.inmodelia.com/gsoc2020-redirection.html with redirection to the (slow) NIST page.

Examples

```
ht(uGauss2)
plot(uGauss2)
```

uGauss3 *Dataset uGauss3*

Description

An univariate dataset (x, y) of class data.frame and dim 250 x 2 to be fitted by a neural network with 4 hidden neurons (13 parameters).

References

Rust, B., NIST (1996) https://www.inmodelia.com/gsoc2020-redirection.html with redirection to the (slow) NIST page.

Examples

```
ht(uGauss3)
plot(uGauss3)
```

uNeuroOne *Dataset uNeuroOne*

Description

An univariate dataset (x, y) of class data.frame and dim 51 x 2 to be fitted by a neural network with 2 hidden neurons (7 parameters). This dataset was used to teach neural networks at ESPCI from 1991 to 2013 and is available in the software Neuro One.

References

Dreyfus, G., ESPCI https://www.neurones.espci.fr

Neuro One https://www.inmodelia.com/software.html

Examples

```
ht(uNeuroOne)
plot(uNeuroOne)
```

Index