Package 'deepImp'

June 10, 2026

Type Package

Title Imputation with Deep Learning Methods

Version 1.1.0

Description Imputation of mixed-type and compositional data with neural networks. The architecture (number and size of hidden layers, dropout, activation, optimiser) is user-configurable. See Templ (2021) <doi:10.1007/978-3-030-71175-7>.

License GPL-2

Encoding UTF-8

LazyData TRUE

ByteCompile TRUE

Depends R (>= 4.1)

Imports torch, luz, VIM, robCompositions, stats, utils, graphics

Suggests keras3, knitr, rmarkdown, testthat (>= 3.0.0)

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Matthias Templ [aut, cre] (ORCID: <https://orcid.org/0000-0002-8638-5276>)

Maintainer Matthias Templ <matthias.templ@gmail.com>

Repository CRAN

Date/Publication 2026-06-10 08:10:20 UTC

Contents

`beer` *Beer ageing volatile-compound data*

Description

Concentrations of 16 volatile compounds measured in beer samples, together with an indicator distinguishing fresh from aged beer. The volatile compounds are recognised markers of beer flavour and ageing (e.g. furfural, 5-hydroxymethylfurfural, hexanal, methional-related aldehydes). The data are used to illustrate imputation of mixed continuous measurements.

Usage

```
data(beer)
```

Format

A data frame with 86 rows and 17 variables:

v3MeBual 3-methylbutanal concentration.

v3MeBuon 3-methylbutanone concentration.

v2MeBual 2-methylbutanal concentration.

vHexanal hexanal concentration.

v2FurMeol 2-furanmethanol (furfuryl alcohol) concentration.

vHeptanal heptanal concentration.

v2AcFur 2-acetylfuran concentration.

v5Me2Fur 5-methyl-2-furaldehyde concentration.

vEssFuEst furanoic acid ethyl ester concentration.

v2Ac5MeFu 2-acetyl-5-methylfuran concentration.

v2PhEtal 2-phenylethanal (phenylacetaldehyde) concentration.

vNicEtEst nicotinic acid ethyl ester concentration.

v2PhEssEt 2-phenylethyl acetate concentration.

vgNonalac gamma-nonalactone concentration.

vFurfural furfural concentration.

vHMF 5-hydroxymethylfurfural (HMF) concentration.

vnewold2 indicator of beer condition (fresh vs. aged).

deepimp_arch *Architecture configuration for neural-network imputation*

Description

A backend-neutral description of the multilayer perceptron used by `impNNet()`. The same object is translated into a torch (and, later, keras) model.

Usage

```
deepimp_arch(
  hidden = c(256, 128, 64),
  dropout = 0.1,
  activation = "relu",
  batchnorm = TRUE,
  optimizer = "adam",
  learning_rate = 0.001
)

deepimp_arch_small(
  dropout = 0.1,
  activation = "relu",
  batchnorm = TRUE,
  optimizer = "adam",
  learning_rate = 0.001
)
```

Arguments

`hidden`	integer vector of hidden-layer widths; its length is the depth.
`dropout`	dropout rate applied after every hidden layer; a scalar in `[0, 1)`.
`activation`	hidden-layer activation: one of `"relu"`, `"tanh"`, `"sigmoid"`, `"elu"`, `"leaky_relu"`.
`batchnorm`	logical; apply batch normalisation after each hidden linear layer.
`optimizer`	one of `"adam"`, `"sgd"`, `"rmsprop"`.
`learning_rate`	positive learning rate for the optimiser.

Value

an object of class `"deepimp_arch"`.

See Also

`impNNet()`, `impNNetCoDa()`

Examples

```
deepimp_arch(hidden = c(128, 64), dropout = 0.2)
```

getImputed *Extract imputed data from a deepimp object*

Description

Extract imputed data from a deepimp object

Usage

```
getImputed(object, m = 1L, ...)
```

Arguments

object	a "deepimp" object from impNNet().
m	which completed dataset to return: an integer index, or "all" for the list of all imputations. With m > 1 the datasets are stochastic replicate completions, **not** valid multiple imputation.
...	unused.

Value

a data.frame, or a list of data.frames when m = "all".

See Also

impNNet()

guessType *Guess the measurement type of each variable*

Description

Classifies each column as "numeric", "mixed" (semi-continuous, i.e. a spike of repeated values such as zeros plus a continuous part), "binary", "nominal", or "count". Used by impNNet() to choose the model head, and by summary() of a "deepimp" object.

Usage

```
guessType(x)
```

Arguments

x	a data.frame, data.table, or tibble.

Value

a list with `indices` (a type-by-variable logical matrix) and `type` (a character vector, one entry per column).

Author(s)

Matthias Templ

Examples

```
data(sleep, package = "VIM")
guessType(sleep)
```

`impNNet` *Neural-network imputation for mixed-type data*

Description

Iterative, chained imputation of numeric, count, semi-continuous, binary and nominal variables with a configurable multilayer perceptron (torch backend).

Usage

```
impNNet(
  data,
  arch = NULL,
  m = 1L,
  backend = c("torch", "keras"),
  vartypes = "guess",
  initialize = "knn",
  iterations = 3L,
  eps = 0.01,
  normalize = TRUE,
  epochs = 400L,
  patience = 40L,
  validation_split = 0.2,
  batch_size = 32L,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`data`	a data.frame (tibbles/data.tables are coerced).
`arch`	a `deepimp_arch()` object, or `NULL` to build one from `....` Supplying both `arch` and architecture scalars via `...` is an error.

`m`	number of stochastic replicate completions. NOTE: with `m > 1` these are stochastic replicate completions, **not** valid multiple imputation (no Rubin-rule inference).
`backend`	`"torch"` (default) or `"keras"` (optional; requires the keras3 package and a working Keras/TensorFlow backend).
`vartypes`	`"guess"` (use `guessType()`) or a length-`ncol(data)` vector.
`initialize`	starting values for NAs: `"knn"`, `"mean"`, or `"hotdeck"`.
`iterations`	maximum number of chained sweeps.
`eps`	convergence tolerance on the standardised mean change.
`normalize`	standardise numeric predictors.
`epochs`, `patience`, `validation_split`, `batch_size`	training controls.
`seed`	optional integer seed (sets the R and backend RNGs). Exact reproducibility is guaranteed when `dropout = 0`; with `dropout > 0` the backend's training-time mask sampling is not fully pinned by the seed in the current torch build, so repeated runs are close but may not be bit-identical.
`verbose`	print progress.
`...`	architecture scalar overrides forwarded to `deepimp_arch()`.

Value

a `"deepimp"` object; read the data with `getImputed()`.

Author(s)

Matthias Templ

See Also

`deepimp_arch()`, `getImputed()`, `guessType()`

Examples

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {

  data(sleep, package = "VIM")
  imp <- impNNet(sleep, arch = deepimp_arch_small(), epochs = 5, seed = 1)
  head(getImputed(imp))

}
```

`impNNetCoDa` *Neural-network imputation of rounded zeros in compositional data*

Description

Imputes rounded zeros (values below a detection limit) in compositional data with a configurable neural network, following Templ (2021). Each part with rounded zeros is pivoted to the front, transformed to pivot log-ratio coordinates, regressed on the remaining coordinates, censored at the detection limit, and back-transformed with preservation of the observed absolute values.

Usage

```
impNNetCoDa(
  x,
  dl = NULL,
  label = 0,
  coda = TRUE,
  correction = c("truncate", "expectation", "none"),
  initialize = "kNNa",
  arch = NULL,
  m = 1L,
  backend = c("torch", "keras"),
  iterations = 2L,
  eps = 0.01,
  normalize = TRUE,
  epochs = 400L,
  patience = 40L,
  validation_split = 0.2,
  batch_size = 32L,
  seed = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	a data.frame of compositional parts (positive values; rounded zeros marked by `label`).
`dl`	detection limits, a numeric vector of length `ncol(x)`. Required unless `correction = "none"`.
`label`	the value marking a rounded zero in `x` (converted to `NA` internally). Default `0`.
`coda`	if `TRUE` (default), work in pivot log-ratio coordinates; if `FALSE`, impute on the raw scale (ablation).
`correction`	censoring of imputed values: `"truncate"` (default; set values above the detection limit to the limit, per Templ 2021), `"expectation"` (truncated-normal mean; opt-in, beyond Templ 2021), or `"none"` (no censoring).

initialize	starting values for the rounded zeros: "kNNa" (default, compositional kNN), "knn", "hotdeck", or "mean".
arch	a deepimp_arch() object, or NULL to build one from ....
m	number of stochastic replicate completions (not valid multiple imputation; see impNNet()).
backend	"torch" (default) or "keras" (optional; requires keras3).
iterations	maximum number of chained sweeps.
eps	convergence tolerance on the standardised mean change.
normalize	standardise numeric predictors.
epochs, patience, validation_split, batch_size	training controls.
seed	optional integer seed (sets the R and backend RNGs). Exact reproducibility is guaranteed when dropout = 0; with dropout > 0 the backend's training-time mask sampling is not fully pinned by the seed in the current torch build, so repeated runs are close but may not be bit-identical.
verbose	print progress.
...	architecture scalar overrides forwarded to deepimp_arch().

Value

a "deepimp" object; read the data with getImputed().

Author(s)

Matthias Templ

References

Templ, M. (2021) Imputation of rounded zeros for high-dimensional compositional data.

See Also

impNNet(), deepimp_arch(), getImputed()

Examples

```
if (requireNamespace("torch", quietly = TRUE) && torch::torch_is_installed()) {

  set.seed(1)
  x <- data.frame(a = runif(50, 5, 10), b = runif(50, 5, 10), c = runif(50, 5, 10))
  x$a[1:5] <- 0
  imp <- impNNetCoDa(x, dl = c(1, 1, 1), label = 0,
                     arch = deepimp_arch_small(), epochs = 5, seed = 1)
  head(getImputed(imp))

}
```

`new_deepimp` *Construct a deepimp object*

Description

Low-level constructor for the object returned by `impNNet()`. Most users do not call this directly; use `getImputed()` to read the completed data.

Usage

```
new_deepimp(data, imputed, arch, info = list())
```

Arguments

`data`	data.frame with missing values (original input).
`imputed`	list of completed data.frames (one per imputation).
`arch`	a `deepimp_arch()` object describing the network.
`info`	list with training metadata (backend, vartypes, convergence, ...).

Value

an object of class `"deepimp"`.

See Also

`getImputed()`, `impNNet()`

Index