Package: soilVAE (via r-universe)

May 18, 2026

Type Package

Title Supervised Variational Autoencoder Regression via 'reticulate'

Version 0.1.9

Description Supervised latent-variable regression for high-dimensional predictors such as soil reflectance spectra. The model uses an encoder-decoder neural network with a stochastic Gaussian latent representation regularized by a Kullback-Leibler term, and a supervised prediction head trained jointly with the reconstruction objective. The implementation interfaces R with a 'Python' deep-learning backend and provides utilities for training, tuning, and prediction.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports reticulate, stats

Suggests knitr, rmarkdown, prospectr, pls

VignetteBuilder knitr

SystemRequirements Python (>= 3.9); TensorFlow (>= 2.13); Keras (>= 3)

URL https://hugomachadorodrigues.github.io/soilVAE/, https://github.com/HugoMachadoRodrigues/soilVAE/

BugReports https://github.com/HugoMachadoRodrigues/soilVAE/issues

Config/pak/sysreqs libpng-dev python3

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

Date/Publication 2026-03-12 21:05:21 UTC

RemoteUrl https://github.com/hugomachadorodrigues/soilvae

RemoteRef HEAD

RemoteSha a9b8f4dc224e59e931b3f092d5487d8b7fae6e44

Contents

datsoilspc 2
select_best_from_grid 3
tune_vae_train_val 4
vae_build 5
vae_configure 6
vae_encode 6
vae_fit 7
vae_predict 8

Index **9**

`datsoilspc` *Soil spectroscopy example dataset used in the soilVAE vignettes*

Description

A small soil spectroscopy dataset packaged with **soilVAE** for demonstrating typical spectral preprocessing (reflectance → absorbance, resampling, SNV, smoothing) and for comparing a classic PLS baseline model against supervised VAE regression via **soilVAE**.

Format

A data.frame or list containing at minimum:

spc Numeric matrix/data.frame of reflectance spectra (samples × wavelengths).

TotalCarbon Numeric vector of total carbon values.

Details

The object `datsoilspc` contains:

- `spc`: a numeric matrix (or data.frame) of reflectance spectra, with rows as samples and columns as wavelengths (nm). Column names should be interpretable as numeric wavelengths.
- `TotalCarbon`: a numeric vector with the soil total carbon content for each sample.

Depending on the original source, additional columns may be present (e.g., sample identifiers or other soil properties).

The dataset is intended for examples and unit-sized demonstrations. It is not meant to be a comprehensive soil spectral library.

Examples

```
data("datsoilspc", package = "soilVAE")
str(datsoilspc)

# basic plot of reflectance spectra
spc <- as.matrix(datsoilspc$spc)
wav <- as.numeric(colnames(spc))
matplot(wav, t(spc), type = "l", lty = 1,
        xlab = "Wavelength / nm", ylab = "Reflectance")
```

`select_best_from_grid` *Select the best configuration from a tuning table*

Description

Select the best configuration from a tuning table

Usage

```
select_best_from_grid(
  tuning_df,
  selection_metric = c("euclid", "rmse", "r2", "rpiq")
)
```

Arguments

`tuning_df`	Data frame containing `RMSE_val`, `R2_val`, and `RPIQ_val`.
`selection_metric`	One of: `"euclid"`, `"rmse"`, `"r2"`, `"rpiq"`.

Value

List with `best` (one-row data frame) and `best_score`.

Examples

```
tuning_df <- data.frame(
  cfg_id   = 1:3,
  RMSE_val = c(0.5, 0.3, 0.4),
  R2_val   = c(0.8, 0.9, 0.85),
  RPIQ_val = c(2.0, 3.0, 2.5)
)
best <- select_best_from_grid(tuning_df, selection_metric = "rmse")
best$best
```

tune_vae_train_val *Tune VAEReg on a train/validation split*

Description

Tune VAEReg on a train/validation split

Usage

```
tune_vae_train_val(X_tr, y_tr, X_va, y_va, seed = 123, grid_vae)
```

Arguments

X_tr	Train predictors matrix.
y_tr	Train response numeric.
X_va	Validation predictors matrix.
y_va	Validation response numeric.
seed	Integer seed.
grid_vae	Data frame with required columns: hidden_enc (list), hidden_dec (list), latent_dim, dropout, lr, beta_kl, alpha_y, epochs, batch_size, patience.

Value

A list with grid (input grid) and tuning_df (metrics per config).

Examples

```
## Not run:
  vae_configure()
  X <- matrix(rnorm(500), nrow = 50, ncol = 10)
  y <- rnorm(50)
  grid <- data.frame(
    latent_dim = 4L, dropout = 0.1, lr = 0.001,
    beta_kl = 1, alpha_y = 1, epochs = 5L,
    batch_size = 16L, patience = 3L
  )
  grid$hidden_enc <- list(c(32L, 16L))
  grid$hidden_dec <- list(c(16L, 32L))
  result <- tune_vae_train_val(
    X_tr = X[1:40, ], y_tr = y[1:40],
    X_va = X[41:50, ], y_va = y[41:50],
    grid_vae = grid
  )

## End(Not run)
```

vae_build *Build a supervised VAE regression model (VAEReg)*

Description

Build a supervised VAE regression model (VAEReg)

Usage

```
vae_build(
  input_dim,
  hidden_enc = c(512L, 256L),
  hidden_dec = c(256L, 512L),
  latent_dim = 32L,
  dropout = 0.1,
  lr = 0.001,
  beta_kl = 1,
  alpha_y = 1
)
```

Arguments

input_dim	integer
hidden_enc	integer vector
hidden_dec	integer vector
latent_dim	integer
dropout	numeric
lr	numeric learning rate
beta_kl	numeric
alpha_y	numeric

Value

A fitted 'Python' 'Keras' model object (VAEReg).

Examples

```
## Not run:
  vae_configure()
  model <- vae_build(input_dim = 100L)

## End(Not run)
```

vae_configure *Configure Python / reticulate for soilVAE*

Description

Configure Python / reticulate for soilVAE

Usage

```
vae_configure(python = NULL, venv = NULL, conda = NULL)
```

Arguments

python	Path to python executable (optional).
venv	Name of virtualenv to use (optional).
conda	Name of conda env to use (optional).

Value

TRUE invisibly.

Examples

```
## Not run:
  # Let reticulate auto-detect Python
  vae_configure()

  # Or point to an existing virtual environment
  vae_configure(venv = "r-soilvae")

## End(Not run)
```

vae_encode *Extract latent embeddings (z) from VAEReg*

Description

Extract latent embeddings (z) from VAEReg

Usage

```
vae_encode(model, X)
```

Arguments

model	Python VAEReg
X	matrix

Value

Matrix of latent embeddings (samples x latent_dim).

Examples

```
## Not run:
  vae_configure()
  X <- matrix(rnorm(200), nrow = 20, ncol = 10)
  y <- rnorm(20)
  model <- vae_build(input_dim = ncol(X))
  vae_fit(model, X, y, epochs = 5L)
  z <- vae_encode(model, X)

## End(Not run)
```

---

vae_fit *Fit VAEReg*

Description

Fit VAEReg

Usage

```
vae_fit(
  model,
  X,
  y,
  X_val = NULL,
  y_val = NULL,
  epochs = 80L,
  batch_size = 64L,
  patience = 10L,
  verbose = 0L
)
```

Arguments

model	Python VAEReg object from vae_build()
X	matrix (n x p)
y	numeric vector (n)
X_val	optional matrix
y_val	optional numeric vector
epochs	integer
batch_size	integer
patience	integer for early stopping (only if validation provided)
verbose	0/1/2

Value

The fitted model, invisibly.

Examples

```
## Not run:
  vae_configure()
  X <- matrix(rnorm(200), nrow = 20, ncol = 10)
  y <- rnorm(20)
  model <- vae_build(input_dim = ncol(X))
  vae_fit(model, X, y, epochs = 5L)

## End(Not run)
```

---

vae_predict *Predict y using VAEReg (via latent z -> y_head)*

Description

Predict y using VAEReg (via latent z -> y_head)

Usage

```
vae_predict(model, X)
```

Arguments

model	Python VAEReg
X	matrix

Value

Numeric vector of predicted values.

Examples

```
## Not run:
  vae_configure()
  X <- matrix(rnorm(200), nrow = 20, ncol = 10)
  y <- rnorm(20)
  model <- vae_build(input_dim = ncol(X))
  vae_fit(model, X, y, epochs = 5L)
  preds <- vae_predict(model, X)

## End(Not run)
```

Index

∗ **datasets**
datsoilspc, 2

datsoilspc, 2

select_best_from_grid, 3

tune_vae_train_val, 4

vae_build, 5
vae_configure, 6
vae_encode, 6
vae_fit, 7
vae_predict, 8