

# Package ‘deepspat’

September 16, 2026

**Type** Package

**Title** Deep Compositional Spatial Models

**Date** 2026-09-16

**Version** 0.3.3

**Maintainer** Quan Vu <quanvustats@gmail.com>

**Description** Deep compositional spatial models are standard spatial covariance models coupled with an injective warping function of the spatial domain. The warping function is constructed through a composition of multiple elemental injective functions in a deep-learning framework. The package implements two cases for the univariate setting; first, when these warping functions are known up to some weights that need to be estimated, and, second, when the weights in each layer are random. In the multivariate setting only the former case is available. Estimation and inference is done using `tensorflow`, which makes use of graphics processing units. For more details see Zammit-Mangion et al. (2022) <[doi:10.1080/01621459.2021.1887741](https://doi.org/10.1080/01621459.2021.1887741)>, Vu et al. (2022) <[doi:10.5705/ss.202020.0156](https://doi.org/10.5705/ss.202020.0156)>, Vu et al. (2023) <[doi:10.1016/j.spasta.2023.100742](https://doi.org/10.1016/j.spasta.2023.100742)>, and Shao et al. (2025) <[doi:10.48550/arXiv.2505.12548](https://doi.org/10.48550/arXiv.2505.12548)>.

**License** Apache License 2.0

**Imports** data.table, dplyr, Matrix, methods, rlang, reticulate, keras, tensorflow, tfprobability, evd, fields

**SystemRequirements** TensorFlow (<https://www.tensorflow.org/>),

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**NeedsCompilation** no

**URL** <https://cran.r-project.org/package=deepspat>

**BugReports** <https://github.com/andrewzm/deepspat/issues>

**Author** Andrew Zammit-Mangion [aut],  
Quan Vu [aut, cre],  
Xuanjie Shao [aut]

**Repository** CRAN  
**Date/Publication** 2026-09-16 07:30:08 UTC

**Contents**

|                                      |           |
|--------------------------------------|-----------|
| AFF_1D . . . . .                     | 2         |
| AFF_2D . . . . .                     | 3         |
| AWU . . . . .                        | 4         |
| bisquares1D . . . . .                | 5         |
| bisquares2D . . . . .                | 6         |
| deepspat . . . . .                   | 6         |
| deepspat_bivar_GP . . . . .          | 9         |
| deepspat_GP . . . . .                | 11        |
| deepspat_MSP . . . . .               | 13        |
| deepspat_nn_GP . . . . .             | 15        |
| deepspat_nn_ST_GP . . . . .          | 17        |
| deepspat_rPP . . . . .               | 19        |
| deepspat_trivar_GP . . . . .         | 21        |
| initvars . . . . .                   | 24        |
| init_learn_rates . . . . .           | 25        |
| LFT . . . . .                        | 26        |
| plot.deepspat . . . . .              | 27        |
| predict.deepspat . . . . .           | 29        |
| predict.deepspat_bivar_GP . . . . .  | 30        |
| predict.deepspat_GP . . . . .        | 31        |
| predict.deepspat_MSP . . . . .       | 31        |
| predict.deepspat_nn_GP . . . . .     | 32        |
| predict.deepspat_nn_ST_GP . . . . .  | 33        |
| predict.deepspat_rPP . . . . .       | 34        |
| predict.deepspat_trivar_GP . . . . . | 35        |
| print.deepspat . . . . .             | 35        |
| RBF_block . . . . .                  | 36        |
| set_deepspat_seed . . . . .          | 37        |
| sim_data . . . . .                   | 38        |
| summary.deepspat . . . . .           | 39        |
| <b>Index</b>                         | <b>41</b> |

---

|        |                                             |
|--------|---------------------------------------------|
| AFF_1D | <i>Affine transformation on a 1D domain</i> |
|--------|---------------------------------------------|

---

**Description**

Sets up an affine transformation on a 1D domain

**Usage**

```
AFF_1D(a = c(0, 1), dtype = "float32")
```

**Arguments**

**a** vector of two real numbers describing an affine transformation on a 1D domain

**dtype** data type

**Value**

AFF\_1D returns a list containing a list with the following components:

**"f"** An encapsulated function that takes an input and evaluates the affine transformation using TensorFlow

**"fR"** Same as f but uses R

**"r"** The number of basis functions (fixed to 1 in this case)

**"trans"** The transformation applied to the weights before estimation (in this case the identity)

**"fix\_weights"** Flag indicating whether the weights are fixed or not (TRUE in this case)

**"name"** Name of layer

**"pars"** List of parameters describing the affine transformation as TensorFlow objects

---

AFF\_2D

*Affine transformation on a 2D domain*

---

**Description**

Sets up an affine transformation on a 2D domain

**Usage**

```
AFF_2D(a = c(0, 1, 0, 0, 0, 1), dtype = "float32")
```

**Arguments**

**a** vector of six real numbers describing an affine transformation on a 2D domain

**dtype** data type

**Value**

AFF\_2D returns a list containing a list with the following components:

- "f"** An encapsulated function that takes an input and evaluates the affine transformation using TensorFlow
- "fR"** Same as f but uses R
- "r"** The number of basis functions (fixed to 1 in this case)
- "trans"** The transformation applied to the weights before estimation (in this case the identity)
- "fix\_weights"** Flag indicating whether the weights are fixed or not (TRUE in this case)
- "name"** Name of layer
- "pars"** List of parameters describing the affine transformation as TensorFlow objects

---

AWU

*Axial Warping Unit*


---

**Description**

Sets up an axial warping unit (AWU) for used in a deep compositional spatial model. The function sets up sigmoids on a prescribed domain at regular intervals, with 'steepness' indicated by the user. It returns a list of length 1 containing an axial warping unit (AWU) and several encapsulated functions that evaluate the AWU over inputs of different types. See Value for more details.

**Usage**

```
AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5), dtype = "float32")
```

**Arguments**

- |       |                                                       |
|-------|-------------------------------------------------------|
| r     | number of basis functions                             |
| dim   | dimension to warp                                     |
| grad  | steepness of the sigmoid functions                    |
| lims  | the bounded 1D domain on which to set up the sigmoids |
| dtype | data type                                             |

**Value**

AWU returns a list containing a list with the following components:

- "f"** An encapsulated function that takes an input and evaluates the sigmoids over the dim-th dimension using TensorFlow
- "fR"** Same as f but uses R
- "fMC"** Same as f but does it in parallel for several inputs index by the first dimension of the tensor
- "r"** The number of sigmoid basis functions
- "trans"** The transformation applied to the weights before estimation
- "fix\_weights"** Flag indicating whether the weights are fixed or not (FALSE for AWUs)
- "name"** Name of layer

Examples

```
if (reticulate::py_module_available("tensorflow")) {
  layer <- AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5))
}
```

---

|             |                                          |
|-------------|------------------------------------------|
| bisquares1D | <i>Bisquare functions on a 1D domain</i> |
|-------------|------------------------------------------|

---

Description

Sets up a top-layer set of bisquare basis functions on a bounded 1D domain of length 1 for modelling the process  $Y$ . It returns a list of length 1 containing the basis functions and encapsulated functions that evaluate the bisquare functions over inputs of different types. See Value for more details.

Usage

```
bisquares1D(r = 30, lims = c(-0.5, 0.5), dtype = "float32")
```

Arguments

|       |                                                                                                  |
|-------|--------------------------------------------------------------------------------------------------|
| r     | 30                                                                                               |
| lims  | the limits of one side of the square bounded 2D domain on which to set up the bisquare functions |
| dtype | data type                                                                                        |

Value

bisquares1D returns a list containing a list with the following components:

- "f" An encapsulated function that takes an input and evaluates the sigmoids over the dim-th dimension using TensorFlow
- "r" The number of sigmoid basis functions
- "knots\_tf" The centroids of the basis functions as a TensorFlow object
- "name" Name of layer

---

|             |                                          |
|-------------|------------------------------------------|
| bisquares2D | <i>Bisquare functions on a 2D domain</i> |
|-------------|------------------------------------------|

---

### Description

Sets up a top-layer set of bisquare basis functions on a square 2D domain of size 1 x 1 for modelling the process  $Y$ . It returns a list of length 1 containing the basis functions and encapsulated functions that evaluate the bisquare functions over inputs of different types. See Value for more details.

### Usage

```
bisquares2D(r = 30, lims = c(-0.5, 0.5), dtype = "float32")
```

### Arguments

|                    |                                                                 |
|--------------------|-----------------------------------------------------------------|
| <code>r</code>     | 30                                                              |
| <code>lims</code>  | the bounded 1D domain on which to set up the bisquare functions |
| <code>dtype</code> | data type                                                       |

### Value

bisquares1D returns a list containing a list with the following components:

- "f" An encapsulated function that takes an input and evaluates the sigmoids over the dim-th dimension using TensorFlow
- "fR" Same as f but uses R
- "r" The number of sigmoid basis functions
- "knots\_tf" The centroids of the basis functions as a TensorFlow object
- "knots" The centroids of the basis functions as an R object
- "name" Name of layer

---

|          |                                          |
|----------|------------------------------------------|
| deepspat | <i>Deep compositional spatial models</i> |
|----------|------------------------------------------|

---

### Description

Deep compositional spatial models are standard low-rank spatial models coupled with a bijective warping function of the spatial domain. The warping function is constructed through a composition of multiple elemental bijective functions in a deep-learning framework. The package implements two cases; first, when these functions are known up to some weights that need to be estimated, and, second, when the weights in each layer are random. Estimation and inference is done using TensorFlow, which makes use of graphical processing units.

Fits the original deepspat model using either SIWGP or SDSP inference.

**Usage**

```

deepspat(
  f,
  data,
  layers = NULL,
  method = c("VB", "ML"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  MC = 10L,
  nsteps
)

```

**Arguments**

|                          |                                                                                                                                                                   |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>f</code>           | formula identifying the dependent variable and the spatial inputs (RHS can only have one or two variables)                                                        |
| <code>data</code>        | data frame containing the required data                                                                                                                           |
| <code>layers</code>      | list containing the warping layers                                                                                                                                |
| <code>method</code>      | either 'ML' (for the SIWGP) or 'VB' (for the SDSP)                                                                                                                |
| <code>par_init</code>    | list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list                                                      |
| <code>learn_rates</code> | learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list                        |
| <code>MC</code>          | number of MC samples when doing stochastic variational inference                                                                                                  |
| <code>nsteps</code>      | number of steps when doing gradient descent times three (first time the weights are optimised, then the covariance-function parameters, then everything together) |

**Details**

The main model-fitting functions are grouped by model class:

- `deepspat()` fits the original spatial input-warped Gaussian process and spatial deep stochastic process models.
- `deepspat_GP()`, `deepspat_nn_GP()`, and `deepspat_nn_ST_GP()` fit univariate Gaussian process models, including nearest-neighbor and spatio-temporal variants.
- `deepspat_bivar_GP()` and `deepspat_trivar_GP()` fit multivariate Gaussian process models.
- `deepspat_MSP()` and `deepspat_rPP()` fit extreme-value models based on max-stable and r-Pareto processes.

Prediction, summary, plot, and print methods are provided for fitted Gaussian process and extreme-value models through standard S3 methods. Warping layers and basis components can be constructed with `AFF_1D()`, `AFF_2D()`, `AWU()`, `LFT()`, `RBF_block()`, `bisquares1D()`, and `bisquares2D()`.

**Value**

deepspat returns an object of class deepspat with the following items

- "Cost"** The final value of the cost (NMLL for the SIWGP and the lower bound for the SDSP, plus a constant)
- "mupost\_tf"** Posterior means of the weights in the top layer as a TensorFlow object
- "Qpost\_tf"** Posterior precision of the weights in the top layer as a TensorFlow object
- "eta\_tf"** Estimated or posterior means of the weights in the warping layers as a list of TensorFlow objects
- "precy\_tf"** Precision of measurement error, as a TensorFlow object
- "sigma2eta\_tf"** Variance of the weights in the top layer, as a TensorFlow object
- "l\_tf"** Length scale used to construct the covariance matrix of the weights in the top layer, as a TensorFlow object
- "scalings"** Minima and maxima used to scale the unscaled unit outputs for each layer, as a list of TensorFlow objects
- "method"** Either 'ML' or 'VB'
- "nlayers"** Number of layers in the model (including the top layer)
- "MC"** Number of MC samples when doing stochastic variational inference
- "run"** TensorFlow session for evaluating the TensorFlow objects
- "f"** The formula used to construct the deepspat model
- "data"** The data used to construct the deepspat model
- "negcost"** Vector of costs after each gradient-descent evaluation
- "data\_scale\_mean"** Empirical mean of the original data
- "data\_scale\_mean\_tf"** Empirical mean of the original data as a TensorFlow object

**Author(s)**

**Maintainer:** Quan Vu <quanvustats@gmail.com>

Authors:

- Andrew Zammit-Mangion
- Xuanjie Shao

**See Also**

Useful links:

- <https://cran.r-project.org/package=deepspat>
- Report bugs at <https://github.com/andrewzm/deepspat/issues>

**Examples**

```

if (reticulate::py_module_available("tensorflow") && reticulate::py_module_available("scipy")) {
df <- data.frame(s1 = rnorm(100), s2 = rnorm(100), z = rnorm(100))
layers <- c(AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5)),
            AWU(r = 50L, dim = 2L, grad = 200, lims = c(-0.5, 0.5)),
            bisquares2D(r = 100))
d <- deepspat(f = z ~ s1 + s2 - 1,
             data = df,
             layers = layers, method = "ML",
             nsteps = 10L)
}

```

deepspat\_bivar\_GP

*Fit a bivariate Gaussian process deepspat model***Description**

Fits a stationary or nonstationary bivariate Gaussian process model with symmetric or asymmetric dependence.

**Usage**

```

deepspat_bivar_GP(
  f,
  data,
  g = ~1,
  layers_asym = NULL,
  layers = NULL,
  method = "REML",
  family = c("exp_stat_symm", "exp_stat_asymm", "exp_nonstat_symm", "exp_nonstat_asymm",
             "matern_stat_symm", "matern_stat_asymm", "matern_nonstat_symm",
             "matern_nonstat_asymm"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)

```

**Arguments**

|             |                                                                                      |
|-------------|--------------------------------------------------------------------------------------|
| f           | formula identifying the dependent variables and the spatial inputs in the covariance |
| data        | data frame containing the required data                                              |
| g           | formula identifying the independent variables in the linear trend                    |
| layers_asym | list containing the aligning function layers                                         |
| layers      | list containing the nonstationary warping layers                                     |

|             |                                                                                                                                            |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| method      | identifying the method for finding the estimates                                                                                           |
| family      | identifying the family of the model constructed                                                                                            |
| par_init    | list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list                               |
| learn_rates | learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list |
| nsteps      | number of steps when doing gradient descent times two, three or five (depending on the family of model)                                    |

### Value

deepspat\_bivar\_GP returns an object of class `deepspat_bivar_GP` with the following items

**"f"** The formula used to construct the covariance model  
**"g"** The formula used to construct the linear trend model  
**"data"** The data used to construct the deepspat model  
**"X"** The model matrix of the linear trend  
**"layers"** The warping function layers in the model  
**"layers\_asym"** The aligning function layers in the model  
**"Cost"** The final value of the cost  
**"eta\_tf"** Estimated weights in the warping layers as a list of TensorFlow objects  
**"eta\_tf\_asym"** Estimated weights in the aligning layers as a list of TensorFlow objects  
**"a\_tf"** Estimated parameters in the LFT layers  
**"a\_tf\_asym"** Estimated parameters in the AFF layers of the aligning function  
**"beta"** Estimated coefficients of the linear trend  
**"precy\_tf1"** Precision of measurement error of the first process, as a TensorFlow object  
**"precy\_tf2"** Precision of measurement error of the second process, as a TensorFlow object  
**"sigma2\_tf\_1"** Variance parameter (first process) in the covariance matrix, as a TensorFlow object  
**"sigma2\_tf\_2"** Variance parameter (second process) in the covariance matrix, as a TensorFlow object  
**"sigma2\_tf\_12"** Covariance parameter in the covariance matrix, as a TensorFlow object  
**"l\_tf1"** Length scale parameter (first process) in the covariance matrix, as a TensorFlow object  
**"l\_tf2"** Length scale parameter (second process) in the covariance matrix, as a TensorFlow object  
**"l\_tf12"** Length scale parameter (cross-covariance) in the covariance matrix, as a TensorFlow object  
**"nu\_tf1"** Smoothness parameter (first process) in the covariance matrix, as a TensorFlow object  
**"nu\_tf2"** Smoothness parameter (second process) in the covariance matrix, as a TensorFlow object  
**"nu\_tf12"** Smoothness parameter (cross-covariance) in the covariance matrix, as a TensorFlow object

"**scalings**" Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of TensorFlow objects

"**scalings\_asym**" Minima and maxima used to scale the unscaled unit outputs for each aligning layer, as a list of TensorFlow objects

"**method**" Method used for inference

"**nlayers**" Number of warping layers in the model

"**nlayers\_asym**" Number of aligning layers in the model

"**swarped\_tf1**" Spatial locations of the first process on the warped domain

"**swarped\_tf2**" Spatial locations of the second process on the warped domain

"**negcost**" Vector of costs after each gradient-descent evaluation

"**z\_tf\_1**" Data of the first process

"**z\_tf\_2**" Data of the second process

"**family**" Family of the model

### Examples

```
if (reticulate::py_module_available("tensorflow") && reticulate::py_module_available("scipy")) {
df <- data.frame(s1 = rnorm(100), s2 = rnorm(100), z1 = rnorm(100), z2 = rnorm(100))
layers <- c(AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5)),
            AWU(r = 50L, dim = 2L, grad = 200, lims = c(-0.5, 0.5)))
d <- deepspat_bivar_GP(f = z1 + z2 ~ s1 + s2 - 1,
                      data = df, g = ~ 1,
                      layers = layers, method = "REML",
                      family = "matern_nonstat_symm",
                      nsteps = 10L)
}
```

---

deepspat\_GP

*Fit a univariate Gaussian process deepspat model*

---

### Description

Fits a stationary or nonstationary univariate Gaussian process model with optional spatial warping.

### Usage

```
deepspat_GP(
  f,
  data,
  g = ~1,
  layers = NULL,
  method = c("REML"),
  family = c("exp_stat", "exp_nonstat", "matern_stat", "matern_nonstat"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

**Arguments**

|                          |                                                                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <code>f</code>           | formula identifying the dependent variables and the spatial inputs in the covariance                                                       |
| <code>data</code>        | data frame containing the required data                                                                                                    |
| <code>g</code>           | formula identifying the independent variables in the linear trend                                                                          |
| <code>layers</code>      | list containing the nonstationary warping layers                                                                                           |
| <code>method</code>      | identifying the method for finding the estimates                                                                                           |
| <code>family</code>      | identifying the family of the model constructed                                                                                            |
| <code>par_init</code>    | list of initial parameter values. Call the function <code>init_tvars()</code> to see the structure of the list                             |
| <code>learn_rates</code> | learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list |
| <code>nsteps</code>      | number of steps when doing gradient descent times two or three (depending on the family of model)                                          |

**Value**

deepspat\_GP returns an object of class `deepspat_GP` with the following items

**"f"** The formula used to construct the covariance model  
**"g"** The formula used to construct the linear trend model  
**"data"** The data used to construct the deepspat model  
**"X"** The model matrix of the linear trend  
**"layers"** The warping function layers in the model  
**"Cost"** The final value of the cost  
**"eta\_tf"** Estimated weights in the warping layers as a list of TensorFlow objects  
**"a\_tf"** Estimated parameters in the LFT layers  
**"beta"** Estimated coefficients of the linear trend  
**"precy\_tf"** Precision of measurement error, as a TensorFlow object  
**"sigma2\_tf"** Variance parameter in the covariance matrix, as a TensorFlow object  
**"l\_tf"** Length scale parameter in the covariance matrix, as a TensorFlow object  
**"nu\_tf"** Smoothness parameter in the covariance matrix, as a TensorFlow object  
**"scalings"** Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of TensorFlow objects  
**"method"** Method used for inference  
**"nlayers"** Number of warping layers in the model  
**"swarped\_tf"** Spatial locations on the warped domain  
**"negcost"** Vector of costs after each gradient-descent evaluation  
**"z\_tf"** Data of the process  
**"family"** Family of the model

## Examples

```
if (reticulate::py_module_available("tensorflow") && reticulate::py_module_available("scipy")) {
  df <- data.frame(s1 = rnorm(100), s2 = rnorm(100), z = rnorm(100))
  layers <- c(AWU(r = 50L, dim = 1L, grad = 200, lims = c(-0.5, 0.5)),
             AWU(r = 50L, dim = 2L, grad = 200, lims = c(-0.5, 0.5)))
  d <- deepspat_GP(f = z ~ s1 + s2 - 1,
                  data = df, g = ~ 1,
                  layers = layers, method = "REML",
                  family = "matern_nonstat",
                  nsteps = 10L)
}
```

---

 deepspat\_MSP

*Fit a max-stable Brown-Resnick deepspat model*


---

## Description

Constructs an extended deep compositional spatial model that supports different estimation methods ("MPL", "MRPL", or "WLS") and spatial dependence families (stationary or non-stationary). This function extends the basic deepspat model by incorporating additional dependence modeling and pre-training steps for the warping layers.

## Usage

```
deepspat_MSP(
  f,
  data,
  layers = NULL,
  method = c("MPL", "MRPL", "WLS"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  family = c("power_stat", "power_nonstat"),
  dtype = "float64",
  nsteps = 100L,
  nsteps_pre = 100L,
  edm_emp = NULL,
  p = c(0, 1),
  pen_coef = 0,
  show = TRUE,
  ...
)
```

## Arguments

**f** A formula identifying the dependent variable(s) and the spatial inputs. Use `get_depvars_multivar3` to extract the dependent variable names.

|                          |                                                                                                                                                                                                              |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>data</code>        | A data frame containing the required data.                                                                                                                                                                   |
| <code>layers</code>      | A list containing the warping layers; required for non-stationary models (i.e., when <code>family = "power_nonstat"</code> ).                                                                                |
| <code>method</code>      | A character string specifying the estimation method. Must be one of "MPL", "MRPL", or "WLS" for max-stable Brown-Resnick processes                                                                           |
| <code>par_init</code>    | A list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list.                                                                                              |
| <code>learn_rates</code> | A list of learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list.                                                        |
| <code>family</code>      | A character string specifying the spatial dependence model. Use "power_nonstat" for non-stationary models and "sta" for stationary models.                                                                   |
| <code>dtype</code>       | A character string indicating the data type for TensorFlow computations ("float32" or "float64"). Default is "float32"                                                                                       |
| <code>nsteps</code>      | An integer specifying the number of training steps for dependence parameter learning.                                                                                                                        |
| <code>nsteps_pre</code>  | An integer specifying the number of pre-training steps for warping layer parameters.                                                                                                                         |
| <code>edm_emp</code>     | For the WLS method, a numeric vector or matrix providing an empirical extremal coefficients.                                                                                                                 |
| <code>p</code>           | For pairwise likelihood based methods, <code>p</code> is used to specify the size of pair subset for pairwise likelihood, or the probability parameter of Bernoulli r.v. for randomized pairwise likelihood. |
| <code>pen_coef</code>    | A penalty parameter for weights of SR-RBF(2) to relieve overfitting.                                                                                                                                         |
| <code>show</code>        | Logical; if TRUE progress information is printed during training.                                                                                                                                            |
| <code>...</code>         | Currently unused.                                                                                                                                                                                            |

### Value

`deepspat_MSP` returns an object of class `deepspat_MSP` which is a list containing the following components:

`layers` The list of warping layers used in the model.

`Cost` The final cost value after training (e.g., negative log-likelihood, least squares, or gradient score).

`transeta_tf` TensorFlow objects for the transformed dependence parameters in the warping layers.

`eta_tf` TensorFlow objects for the warped dependence parameters.

`a_tf` TensorFlow object for the parameters of the LFT layers (if applicable).

`logphi_tf` TensorFlow variable representing the logarithm of the spatial range parameter.

`logitkappa_tf` TensorFlow variable representing the logit-transformed degrees of freedom.

`scalings` A list of scaling limits (minima and maxima) for the input and warped spatial coordinates.

`s_tf` TensorFlow object for the scaled spatial coordinates.

`z_tf` TensorFlow object for the observed response values.  
`swarped_tf` List of TensorFlow objects representing the warped spatial coordinates at each layer.  
`swarped` Matrix of final warped spatial coordinates.  
`method` The estimation method used ("MPL", "MRPL", or "WLS").  
`family` The spatial dependence family ("power\_stat" or "power\_nonstat").  
`dtype` The data type used in TensorFlow computations.  
`nlayers` Number of warping layers (for non-stationary models).  
`f` The model formula.  
`data` The data frame used for model fitting.  
`ndata` Number of observations in data.  
`negcost` Vector of cost values recorded during training.  
`pairs_tf` TensorFlow variable representing the spatial location pairs (and, for MRPL, the replicate indices) used in the pairwise / randomized pairwise likelihood or WLS objective..  
`p` The parameter of Bernoulli r.v. for pairwise likelihood and randomized pairwise likelihood.  
`time` Elapsed time for model fitting.

---

 deepspat\_nn\_GP

*Fit a nearest-neighbor Gaussian process deepspat model*


---

## Description

Fits a stationary or nonstationary Gaussian process model using a nearest-neighbor approximation.

## Usage

```

deepspat_nn_GP(
  f,
  data,
  g = ~1,
  layers = NULL,
  m = 25L,
  order_id,
  nn_id,
  method = c("REML"),
  family = c("exp_stat", "exp_nonstat"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)

```

**Arguments**

|                          |                                                                                                                                            |
|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <code>f</code>           | formula identifying the dependent variables and the spatial inputs in the covariance                                                       |
| <code>data</code>        | data frame containing the required data                                                                                                    |
| <code>g</code>           | formula identifying the independent variables in the linear trend                                                                          |
| <code>layers</code>      | list containing the nonstationary warping layers                                                                                           |
| <code>m</code>           | number of nearest neighbors                                                                                                                |
| <code>order_id</code>    | indices of the order of the observations                                                                                                   |
| <code>nn_id</code>       | indices of the nearest neighbors of the ordered observations                                                                               |
| <code>method</code>      | identifying the method for finding the estimates                                                                                           |
| <code>family</code>      | identifying the family of the model constructed                                                                                            |
| <code>par_init</code>    | list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list                               |
| <code>learn_rates</code> | learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list |
| <code>nsteps</code>      | number of steps when doing gradient descent times two or three (depending on the family of model)                                          |

**Value**

`deepspat_nn_GP` returns an object of class `deepspat_nn_GP` with the following items

**"f"** The formula used to construct the covariance model  
**"g"** The formula used to construct the linear trend model  
**"data"** The data used to construct the deepspat model  
**"X"** The model matrix of the linear trend  
**"layers"** The warping function layers in the model  
**"Cost"** The final value of the cost  
**"eta\_tf"** Estimated weights in the warping layers as a list of TensorFlow objects  
**"a\_tf"** Estimated parameters in the LFT layers  
**"beta"** Estimated coefficients of the linear trend  
**"precy\_tf"** Precision of measurement error, as a TensorFlow object  
**"sigma2\_tf"** Variance parameter in the covariance matrix, as a TensorFlow object  
**"l\_tf"** Length scale parameter in the covariance matrix, as a TensorFlow object  
**"scalings"** Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of TensorFlow objects  
**"method"** Method used for inference  
**"nlayers"** Number of warping layers in the model  
**"swarped\_tf"** Spatial locations on the warped domain  
**"negcost"** Vector of costs after each gradient-descent evaluation  
**"z\_tf"** Data of the process  
**"m"** The number of nearest neighbors  
**"family"** Family of the model

---

|                   |                                                                               |
|-------------------|-------------------------------------------------------------------------------|
| deepspat_nn_ST_GP | <i>Fit a nearest-neighbor spatio-temporal Gaussian process deepspat model</i> |
|-------------------|-------------------------------------------------------------------------------|

---

## Description

Fits a stationary or nonstationary spatio-temporal Gaussian process model using a nearest-neighbor approximation.

## Usage

```
deepspat_nn_ST_GP(
  f,
  data,
  g = ~1,
  layers_spat = NULL,
  layers_temp = NULL,
  m = 25L,
  order_id,
  nn_id,
  method = c("REML"),
  family = c("exp_stat_sep", "exp_stat_asym", "exp_nonstat_sep", "exp_nonstat_asym"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)
```

## Arguments

|             |                                                                                                                                            |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| f           | formula identifying the dependent variables and the spatial inputs in the covariance                                                       |
| data        | data frame containing the required data                                                                                                    |
| g           | formula identifying the independent variables in the linear trend                                                                          |
| layers_spat | list containing the spatial warping layers                                                                                                 |
| layers_temp | list containing the temporal warping layers                                                                                                |
| m           | number of nearest neighbors                                                                                                                |
| order_id    | indices of the order of the observations                                                                                                   |
| nn_id       | indices of the nearest neighbors of the ordered observations                                                                               |
| method      | identifying the method for finding the estimates                                                                                           |
| family      | identifying the family of the model constructed                                                                                            |
| par_init    | list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list                               |
| learn_rates | learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list |
| nsteps      | number of steps when doing gradient descent times two or three (depending on the family of model)                                          |

**Value**

deepspat\_nn\_ST\_GP returns an object of class deepspat\_nn\_ST\_GP with the following items

- "f"** The formula used to construct the covariance model
- "g"** The formula used to construct the linear trend model
- "data"** The data used to construct the deepspat model
- "X"** The model matrix of the linear trend
- "layers\_spat"** The spatial warping function layers in the model
- "layers\_temp"** The temporal warping function layers in the model
- "Cost"** The final value of the cost
- "family"** Family of the model
- "eta\_tf"** Estimated weights in the spatial warping layers as a list of TensorFlow objects
- "eta\_t\_tf"** Estimated weights in the temporal warping layers as a list of TensorFlow objects
- "a\_tf"** Estimated parameters in the LFT layers
- "beta"** Estimated coefficients of the linear trend
- "precy\_tf"** Precision of measurement error, as a TensorFlow object
- "sigma2\_tf"** Variance parameter in the covariance matrix, as a TensorFlow object
- "v\_tf"** Parameters of the covariance matrix (indicating asymmetric spatio-temporal covariance)
- "l\_tf"** Length scale (for spatial dimension) parameter in the covariance matrix, as a TensorFlow object
- "l\_t\_tf"** Length scale (for temporal dimension) parameter in the covariance matrix, as a TensorFlow object
- "scalings"** Minima and maxima used to scale the unscaled unit outputs for each spatial warping layer, as a list of TensorFlow objects
- "scalings\_t"** Minima and maxima used to scale the unscaled unit outputs for each temporal warping layer, as a list of TensorFlow objects
- "method"** Method used for inference
- "nlayers\_spat"** Number of spatial warping layers in the model
- "nlayers\_temp"** Number of temporal warping layers in the model
- "swarped\_tf"** Spatial locations on the warped domain
- "twarped\_tf"** Temporal locations on the warped domain
- "negcost"** Vector of costs after each gradient-descent evaluation
- "z\_tf"** Data of the process
- "m"** The number of nearest neighbors

---

deepspat\_rPP

*Fit an r-Pareto process deepspat model*


---

### Description

Constructs an extended deep compositional spatial model that supports different estimation methods ("GSM" or "WLS") and spatial dependence families (stationary or non-stationary). This function extends the basic deepspat model by incorporating additional dependence modeling and pre-training steps for the warping layers.

### Usage

```
deepspat_rPP(
  f,
  data,
  layers = NULL,
  method = c("WLS", "GSM"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  family = c("power_stat", "power_nonstat"),
  dtype = "float32",
  nsteps = 100L,
  nsteps_pre = 100L,
  edm_emp = NULL,
  risk = NULL,
  thre = NULL,
  weight_fun = NULL,
  dWeight_fun = NULL,
  pen_coef = 0,
  show = TRUE,
  ...
)
```

### Arguments

|             |                                                                                                                                                         |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| f           | A formula identifying the dependent variable(s) and the spatial inputs. Use <code>get_depvars_multivar3</code> to extract the dependent variable names. |
| data        | A data frame containing the required data.                                                                                                              |
| layers      | A list containing the warping layers; required for non-stationary models (i.e., when <code>family = "power_nonstat"</code> ).                           |
| method      | A character string specifying the estimation method. Must be one of "GSM", or "WLS" for r-Pareto processes                                              |
| par_init    | A list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list.                                         |
| learn_rates | A list of learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list.   |

|             |                                                                                                                                            |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| family      | A character string specifying the spatial dependence model. Use "power_nonstat" for non-stationary models and "sta" for stationary models. |
| dtype       | A character string indicating the data type for TensorFlow computations ("float32" or "float64"). Default is "float32"                     |
| nsteps      | An integer specifying the number of training steps for dependence parameter learning.                                                      |
| nsteps_pre  | An integer specifying the number of pre-training steps for warping layer parameters.                                                       |
| edm_emp     | For the LS method, a numeric vector or matrix providing an empirical conditional exceedance probabilities.                                 |
| risk        | For the GS method, a numeric value indicating the risk parameter.                                                                          |
| thre        | A numeric threshold used in the GS method.                                                                                                 |
| weight_fun  | A function used to weight pairwise differences in the GS method.                                                                           |
| dWeight_fun | A function representing the derivative of weight_fun (used in the GS method).                                                              |
| pen_coef    | A penalty parameter for weights of SR-RBF(2) to relieve overfitting.                                                                       |
| show        | Logical; if TRUE progress information is printed during training.                                                                          |
| ...         | Currently unused.                                                                                                                          |

### Value

deepspat\_rPP returns an object of class deepspat\_rPP which is a list containing the following components:

**layers** The list of warping layers used in the model.

**Cost** The final cost value after training (e.g., negative log-likelihood, least squares, or gradient score).

**transeta\_tf** TensorFlow objects for the transformed dependence parameters in the warping layers.

**eta\_tf** TensorFlow objects for the warped dependence parameters.

**a\_tf** TensorFlow object for the parameters of the LFT layers (if applicable).

**logphi\_tf** TensorFlow variable representing the logarithm of the spatial range parameter.

**logitkappa\_tf** TensorFlow variable representing the logit-transformed degrees of freedom.

**scalings** A list of scaling limits (minima and maxima) for the input and warped spatial coordinates.

**s\_tf** TensorFlow object for the scaled spatial coordinates.

**z\_tf** TensorFlow object for the observed response values.

**u\_tf** TensorFlow object for the threshold used in the GS method (if applicable).

**swarped\_tf** List of TensorFlow objects representing the warped spatial coordinates at each layer.

**swarped** Matrix of final warped spatial coordinates.

**method** The estimation method used ("WLS" or "GSM").

**risk** The risk parameter used in the GS method (if applicable).

**family** The spatial dependence family ("power\_stat" or "power\_nonstat").  
**dtype** The data type used in TensorFlow computations.  
**nlayers** Number of warping layers (for non-stationary models).  
**weight\_fun** The weighting function used in the GS method.  
**dWeight\_fun** The derivative of the weighting function used in the GS method.  
**f** The model formula.  
**data** The data frame used for model fitting.  
**negcost** Vector of cost values recorded during training.  
**pairs\_tf** TensorFlow variable representing the spatial location pairs (and, for MRPL, the replicate indices) used in the pairwise / randomized pairwise likelihood or WLS objective..  
**pairs\_t\_tf** Tranposed pairs\_tf.  
**time** Elapsed time for model fitting.

---

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| deepspat_trivar_GP | <i>Fit a trivariate Gaussian process deepspat model</i> |
|--------------------|---------------------------------------------------------|

---

## Description

Fits a stationary or nonstationary trivariate Gaussian process model with symmetric or asymmetric dependence.

## Usage

```

deepspat_trivar_GP(
  f,
  data,
  g = ~1,
  layers_asym_2 = NULL,
  layers_asym_3 = NULL,
  layers = NULL,
  method = c("REML"),
  family = c("matern_stat_symm", "matern_stat_asymm", "matern_nonstat_symm",
    "matern_nonstat_asymm"),
  par_init = initvars(),
  learn_rates = init_learn_rates(),
  nsteps = 150L
)

```

## Arguments

|             |                                                                                      |
|-------------|--------------------------------------------------------------------------------------|
| <b>f</b>    | formula identifying the dependent variables and the spatial inputs in the covariance |
| <b>data</b> | data frame containing the required data                                              |

|                            |                                                                                                                                            |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <code>g</code>             | formula identifying the independent variables in the linear trend                                                                          |
| <code>layers_asym_2</code> | list containing the aligning function layers for the second process                                                                        |
| <code>layers_asym_3</code> | list containing the aligning function layers for the third process                                                                         |
| <code>layers</code>        | list containing the nonstationary warping layers                                                                                           |
| <code>method</code>        | identifying the method for finding the estimates                                                                                           |
| <code>family</code>        | identifying the family of the model constructed                                                                                            |
| <code>par_init</code>      | list of initial parameter values. Call the function <code>initvars()</code> to see the structure of the list                               |
| <code>learn_rates</code>   | learning rates for the various quantities in the model. Call the function <code>init_learn_rates()</code> to see the structure of the list |
| <code>nsteps</code>        | number of steps when doing gradient descent times two, three or five (depending on the family of model)                                    |

### Value

`deepspat_trivar_GP` returns an object of class `deepspat_trivar_GP` with the following items

- `"f"` The formula used to construct the covariance model
- `"g"` The formula used to construct the linear trend model
- `"data"` The data used to construct the deepspat model
- `"X"` The model matrix of the linear trend
- `"layers"` The warping function layers in the model
- `"layers_asym_2"` The aligning function layers for the second process in the model
- `"layers_asym_3"` The aligning function layers for the third process in the model
- `"Cost"` The final value of the cost
- `"eta_tf"` Estimated weights in the warping layers as a list of TensorFlow objects
- `"eta_tf_asym_2"` Estimated weights in the aligning layers for the second process as a list of TensorFlow objects
- `"eta_tf_asym_3"` Estimated weights in the aligning layers for the third process as a list of TensorFlow objects
- `"beta"` Estimated coefficients of the linear trend
- `"precy_tf1"` Precision of measurement error of the first process, as a TensorFlow object
- `"precy_tf2"` Precision of measurement error of the second process, as a TensorFlow object
- `"precy_tf3"` Precision of measurement error of the third process, as a TensorFlow object
- `"sigma2_tf_1"` Variance parameter (first process) in the covariance matrix, as a TensorFlow object
- `"sigma2_tf_2"` Variance parameter (second process) in the covariance matrix, as a TensorFlow object
- `"sigma2_tf_3"` Variance parameter (third process) in the covariance matrix, as a TensorFlow object
- `"sigma2_tf_12"` Covariance parameter (between first and second process) in the covariance matrix, as a TensorFlow object

**"sigma2\_tf\_13"** Covariance parameter (between first and third process) in the covariance matrix, as a TensorFlow object

**"sigma2\_tf\_23"** Covariance parameter (between second and third process) in the covariance matrix, as a TensorFlow object

**"l\_tf1"** Length scale parameter (first process) in the covariance matrix, as a TensorFlow object

**"l\_tf2"** Length scale parameter (second process) in the covariance matrix, as a TensorFlow object

**"l\_tf3"** Length scale parameter (third process) in the covariance matrix, as a TensorFlow object

**"l\_tf12"** Length scale parameter (cross-covariance between first and second process) in the covariance matrix, as a TensorFlow object

**"l\_tf13"** Length scale parameter (cross-covariance between first and third process) in the covariance matrix, as a TensorFlow object

**"l\_tf23"** Length scale parameter (cross-covariance between second and third process) in the covariance matrix, as a TensorFlow object

**"nu\_tf1"** Smoothness parameter (first process) in the covariance matrix, as a TensorFlow object

**"nu\_tf2"** Smoothness parameter (second process) in the covariance matrix, as a TensorFlow object

**"nu\_tf3"** Smoothness parameter (third process) in the covariance matrix, as a TensorFlow object

**"nu\_tf12"** Smoothness parameter (cross-covariance between first and second process) in the covariance matrix, as a TensorFlow object

**"nu\_tf13"** Smoothness parameter (cross-covariance between first and third process) in the covariance matrix, as a TensorFlow object

**"nu\_tf23"** Smoothness parameter (cross-covariance between second and third process) in the covariance matrix, as a TensorFlow object

**"scalings"** Minima and maxima used to scale the unscaled unit outputs for each warping layer, as a list of TensorFlow objects

**"scalings\_asym"** Minima and maxima used to scale the unscaled unit outputs for each aligning layer, as a list of TensorFlow objects

**"method"** Method used for inference

**"nlayers"** Number of warping layers in the model

**"nlayers\_asym"** Number of aligning layers in the model

**"run"** TensorFlow session for evaluating the TensorFlow objects

**"swarped\_tf1"** Spatial locations of the first process on the warped domain

**"swarped\_tf2"** Spatial locations of the second process on the warped domain

**"swarped\_tf3"** Spatial locations of the third process on the warped domain

**"negcost"** Vector of costs after each gradient-descent evaluation

**"z\_tf\_1"** Data of the first process

**"z\_tf\_2"** Data of the second process

**"z\_tf\_3"** Data of the third process

**"family"** Family of the model

---

|          |                                          |
|----------|------------------------------------------|
| initvars | <i>Initialise weights and parameters</i> |
|----------|------------------------------------------|

---

## Description

Provides utility to alter the initial weights and parameters when fitting a deepspat model

## Usage

```
initvars(
  sigma2y = 0.1,
  l_top_layer = 0.5,
  sigma2eta_top_layer = 1,
  nu = 1.5,
  variogram_logrange = log(0.3),
  variogram_logitdf = 0.5,
  transeta_mean_init = list(AWU = -3, RBF = -0.8068528, RBF1 = -0.8068528, RBF2 =
    -0.8068528, LFT = 1, AFF_1D = 1, AFF_2D = 1),
  transeta_mean_prior = list(AWU = -3, RBF = -0.8068528, RBF1 = -0.8068528, RBF2 =
    -0.8068528, LFT = NA),
  transeta_sd_init = list(AWU = 0.01, RBF = 0.01, RBF1 = 0.01, RBF2 = 0.01, LFT = 0.01),
  transeta_sd_prior = list(AWU = 2, RBF = 2, RBF1 = 2, RBF2 = 0.01, LFT = NA)
)
```

## Arguments

|                     |                                                                                                                                                                                                                                      |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sigma2y             | initial value for the measurement-error variance                                                                                                                                                                                     |
| l_top_layer         | initial value for the length scale at the top layer                                                                                                                                                                                  |
| sigma2eta_top_layer | initial value for the variance of the weights at the top layer                                                                                                                                                                       |
| nu                  | initial value for the smoothness parameter                                                                                                                                                                                           |
| variogram_logrange  | initial value for variogram_logrange                                                                                                                                                                                                 |
| variogram_logitdf   | initial value for variogram_logitdf                                                                                                                                                                                                  |
| transeta_mean_init  | list of initial values for the initial weights (or the initial variational means of these weights). The list contains five values, one for the AWU, one for the RBF, one for the LFT (Mobius), and two for the affine transformation |
| transeta_mean_prior | same as transeta_mean_init but for the prior mean of the weights (SDSP only)                                                                                                                                                         |
| transeta_sd_init    | same as transeta_mean_init but for the variational standard deviations (SDSP only)                                                                                                                                                   |

transeta\_sd\_prior  
 same as transeta\_mean\_init but for the prior standard deviations of the weights (SDSP only)

### Value

initvars returns a list with the initial values. Call str(initvars()) to see the structure of this list.

---

|                  |                                  |
|------------------|----------------------------------|
| init_learn_rates | <i>Initialise learning rates</i> |
|------------------|----------------------------------|

---

### Description

Provides utility to alter the learning rates when fitting a deepspat model

### Usage

```
init_learn_rates(
  sigma2y = 5e-04,
  covfun = 0.01,
  sigma2eta = 1e-04,
  eta_mean = 0.1,
  eta_mean2 = 0.1,
  eta_sd = 0.1,
  LFTpars = 0.01,
  AFFpars = 0.01,
  rho = 0.1,
  vario = 0.1
)
```

### Arguments

|           |                                                                                   |
|-----------|-----------------------------------------------------------------------------------|
| sigma2y   | learning rate for the measurement-error variance                                  |
| covfun    | learning rate for the covariance-function (or matrix) parameters at the top layer |
| sigma2eta | learning rate for the process variance                                            |
| eta_mean  | learning rate for the weight estimates or variational means                       |
| eta_mean2 | learning rate for the weight estimates or variational means                       |
| eta_sd    | learning rate for the variational standard deviations (SDSP only)                 |
| LFTpars   | learning rate for the parameters of the Mobius transformation                     |
| AFFpars   | learning rate for the parameters of the affine transformation                     |
| rho       | learning rate for the correlation parameter in the multivariate model             |
| vario     | learning rate for the parameter in the variogram                                  |

**Value**

`init_learn_rates` returns a list with the learning rates. Call `str(init_learn_rates())` to see the structure of this list.

---

LFT

*LFT (Möbius transformation)*


---

**Description**

Sets up a Möbius transformation unit

**Usage**

```
LFT(a = NULL, dtype = "float32")
```

**Arguments**

|                    |                                                                     |
|--------------------|---------------------------------------------------------------------|
| <code>a</code>     | vector of four complex numbers describing the Möbius transformation |
| <code>dtype</code> | data type                                                           |

**Value**

LFT returns a list containing a list with the following components:

**"f"** An encapsulated function that takes an input and evaluates the Möbius transformation using TensorFlow

**"fR"** Same as `f` but uses R

**"fMC"** Same as `f` but does it in parallel for several inputs index by the first dimension of the tensor

**"r"** The number of basis functions (fixed to 1 in this case)

**"trans"** The transformation applied to the weights before estimation (in this case the identity)

**"fix\_weights"** Flag indicating whether the weights are fixed or not (TRUE for LFTs)

**"name"** Name of layer

**"pars"** List of parameters describing the Möbius transformation as TensorFlow objects

**Examples**

```
if (reticulate::py_module_available("tensorflow")) {
  layer <- LFT()
}
```

---

|               |                                    |
|---------------|------------------------------------|
| plot.deepspat | <i>Plot fitted deepspat models</i> |
|---------------|------------------------------------|

---

**Description**

Plot warped space, predictions, covariance maps, and fitted extremal dependence maps.

**Usage**

```
## S3 method for class 'deepspat'
plot(x, type = c("space", "prediction"), newdata = NULL, pred = NULL, ...)

## S3 method for class 'deepspat_GP'
plot(
  x,
  type = c("space", "prediction", "covariance"),
  newdata = NULL,
  pred = NULL,
  reference = NULL,
  ...
)

## S3 method for class 'deepspat_nn_GP'
plot(
  x,
  type = c("space", "prediction", "covariance"),
  newdata = NULL,
  pred = NULL,
  reference = NULL,
  nn_id = NULL,
  ...
)

## S3 method for class 'deepspat_nn_ST_GP'
plot(
  x,
  type = c("space", "prediction", "covariance"),
  newdata = NULL,
  pred = NULL,
  reference = NULL,
  nn_id = NULL,
  ...
)

## S3 method for class 'deepspat_bivar_GP'
plot(
  x,
```

```

    type = c("space", "prediction", "covariance"),
    newdata = NULL,
    pred = NULL,
    reference = NULL,
    component = 1L,
    target_component = component,
    ...
)

## S3 method for class 'deepspat_trivar_GP'
plot(
  x,
  type = c("space", "prediction", "covariance"),
  newdata = NULL,
  pred = NULL,
  reference = NULL,
  component = 1L,
  target_component = component,
  ...
)

## S3 method for class 'deepspat_MSP'
plot(
  x,
  type = c("space", "dependence", "uncertainty"),
  newdata = NULL,
  pred = NULL,
  reference = NULL,
  se = FALSE,
  edm_emp = NULL,
  ...
)

## S3 method for class 'deepspat_rPP'
plot(
  x,
  type = c("space", "dependence", "uncertainty"),
  newdata = NULL,
  pred = NULL,
  reference = NULL,
  se = FALSE,
  edm_emp = NULL,
  uprime = NULL,
  ...
)

```

### Arguments

**x** a fitted deepspat object

|                  |                                                          |
|------------------|----------------------------------------------------------|
| type             | plot type                                                |
| newdata          | optional prediction locations                            |
| pred             | optional object returned by predict()                    |
| ...              | additional graphical arguments                           |
| reference        | reference-site index                                     |
| nn_id            | nearest-neighbor index for prediction plots              |
| component        | optional component index for multivariate GP models      |
| target_component | target component for covariance plots                    |
| se               | logical; compute uncertainty for extreme dependence maps |
| edm_emp          | empirical dependence estimates for WLS uncertainty plots |
| uprime           | threshold used by r-Pareto WLS uncertainty plots         |

**Value**

The plotted prediction object, invisibly.

---

|                  |                                             |
|------------------|---------------------------------------------|
| predict.deepspat | <i>Predict from a fitted deepspat model</i> |
|------------------|---------------------------------------------|

---

**Description**

Computes predictions and prediction intervals from a fitted deepspat object.

**Usage**

```
## S3 method for class 'deepspat'
predict(object, newdata, nsims = 100L, ...)
```

**Arguments**

|         |                                                                        |
|---------|------------------------------------------------------------------------|
| object  | the deepspat object                                                    |
| newdata | data frame containing the prediction locations                         |
| nsims   | number of simulations from the Gaussian mixture components (SDSP only) |
| ...     | currently unused                                                       |

**Value**

predict.deepspat returns a list with the two following items

**"df\_pred"** Data frame containing the predictions/prediction intervals at the prediction locations

**"allsims"** Combined simulations from the Gaussian mixtures (SDSP only)

---

predict.deepspat\_bivar\_GP

*Predict from a fitted bivariate Gaussian process model*


---

### Description

Computes predictions and prediction intervals from a fitted deepspat\_bivar\_GP object.

### Usage

```
## S3 method for class 'deepspat_bivar_GP'
predict(object, newdata, type = NULL, reference = NULL, component = NULL, ...)
```

### Arguments

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| object    | the deepspat_bivar_GP object                                                              |
| newdata   | data frame containing the prediction locations                                            |
| type      | optional prediction type: latent process, response, warped coordinates, or covariance map |
| reference | reference-site index for type = "covariance"                                              |
| component | optional component index                                                                  |
| ...       | currently unused                                                                          |

### Value

predict.deepspat\_bivar\_GP returns a list with the following item

**"df\_pred"** Data frame containing the predictions/prediction intervals at the prediction locations

**"obs\_swarped1"** Observation locations on the warped domain (for the first process)

**"obs\_swarped2"** Observation locations on the warped domain (for the second process)

**"newdata\_swarped1"** New prediction locations on the warped domain (for the first process)

**"newdata\_swarped2"** New prediction locations on the warped domain (for the second process)

---

predict.deepspat\_GP     *Predict from a fitted univariate Gaussian process model*

---

### Description

Computes predictions and prediction intervals from a fitted deepspat\_GP object.

### Usage

```
## S3 method for class 'deepspat_GP'
predict(object, newdata, type = NULL, reference = NULL, ...)
```

### Arguments

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| object    | the deepspat_GP object                                                                    |
| newdata   | data frame containing the prediction locations                                            |
| type      | optional prediction type: latent process, response, warped coordinates, or covariance map |
| reference | reference-site index for type = "covariance"                                              |
| ...       | currently unused                                                                          |

### Value

predict.deepspat\_GP returns a list with the following item

**"df\_pred"** Data frame containing the predictions/prediction intervals at the prediction locations

**"obs\_swarped"** Observation locations on the warped domain

**"newdata\_swarped"** New prediction locations on the warped domain

---

predict.deepspat\_MSP     *Predict fitted dependence from a max-stable process model*

---

### Description

Computes warped coordinates and fitted extremal dependence maps from a fitted deepspat\_MSP object.

**Usage**

```
## S3 method for class 'deepspat_MSP'
predict(
  object,
  newdata,
  type = NULL,
  reference = NULL,
  se = FALSE,
  edm_emp = NULL,
  ...
)
```

**Arguments**

|           |                                                                        |
|-----------|------------------------------------------------------------------------|
| object    | a fitted deepspat_MSP object                                           |
| newdata   | data frame containing prediction locations                             |
| type      | optional prediction type: warped coordinates or fitted dependence map  |
| reference | reference-site index for type = "dependence"                           |
| se        | logical; if TRUE, return delta-method standard errors when available   |
| edm_emp   | empirical extremal dependence estimates for WLS uncertainty assessment |
| ...       | currently unused                                                       |

**Value**

A list containing warped coordinates, fitted dependence parameters, and optionally a reference-site dependence map.

---

predict.deepspat\_nn\_GP

*Predict from a fitted nearest-neighbor Gaussian process model*

---

**Description**

Computes predictions and prediction intervals from a fitted deepspat\_nn\_GP object.

**Usage**

```
## S3 method for class 'deepspat_nn_GP'
predict(object, newdata, nn_id = NULL, type = NULL, reference = NULL, ...)
```

**Arguments**

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| object    | the deepspat_nn_GP object                                                                 |
| newdata   | data frame containing the prediction locations                                            |
| nn_id     | nearest neighbors index                                                                   |
| type      | optional prediction type: latent process, response, warped coordinates, or covariance map |
| reference | reference-site index for type = "covariance"                                              |
| ...       | currently unused                                                                          |

**Value**

predict.deepspat\_nn\_GP returns a list with the following item

**"df\_pred"** Data frame containing the predictions/prediction intervals at the prediction locations

**"obs\_swapped"** Observation locations on the warped domain

**"newdata\_swapped"** New prediction locations on the warped domain

---

predict.deepspat\_nn\_ST\_GP

*Predict from a fitted nearest-neighbor spatio-temporal Gaussian process model*

---

**Description**

Computes predictions and prediction intervals from a fitted deepspat\_nn\_ST\_GP object.

**Usage**

```
## S3 method for class 'deepspat_nn_ST_GP'
predict(object, newdata, nn_id = NULL, type = NULL, reference = NULL, ...)
```

**Arguments**

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| object    | the deepspat_nn_ST_GP object                                                              |
| newdata   | data frame containing the prediction locations                                            |
| nn_id     | nearest neighbors index                                                                   |
| type      | optional prediction type: latent process, response, warped coordinates, or covariance map |
| reference | reference row index for type = "covariance"                                               |
| ...       | currently unused                                                                          |

**Value**

predict.deepspar\_nn\_ST\_GP returns a list with the following item

**"df\_pred"** Data frame containing the predictions/prediction intervals at the prediction locations

**"obs\_swarped"** Observation locations on the spatial warped domain

**"obs\_twarped"** Observation locations on the temporal warped domain

**"newdata\_swarped"** New prediction locations on the spatial warped domain

**"newdata\_twarped"** New prediction locations on the temporal warped domain

---

predict.deepspar\_rPP    *Predict fitted dependence from an r-Pareto process model*

---

**Description**

Computes warped coordinates and fitted CEP dependence maps from a fitted deepspar\_rPP object.

**Usage**

```
## S3 method for class 'deepspar_rPP'
predict(
  object,
  newdata,
  type = NULL,
  reference = NULL,
  se = FALSE,
  edm_emp = NULL,
  uprime = NULL,
  ...
)
```

**Arguments**

|           |                                                                        |
|-----------|------------------------------------------------------------------------|
| object    | a fitted deepspar_rPP object                                           |
| newdata   | data frame containing prediction locations                             |
| type      | optional prediction type: warped coordinates or fitted dependence map  |
| reference | reference-site index for type = "dependence"                           |
| se        | logical; if TRUE, return delta-method standard errors when available   |
| edm_emp   | empirical extremal dependence estimates for WLS uncertainty assessment |
| uprime    | threshold used by the WLS CEP uncertainty assessment                   |
| ...       | currently unused                                                       |

**Value**

A list containing warped coordinates, fitted dependence parameters, and optionally a reference-site dependence map.

---

predict.deepspat\_trivar\_GP

*Predict from a fitted trivariate Gaussian process model*


---

### Description

Computes predictions and prediction intervals from a fitted deepspat\_trivar\_GP object.

### Usage

```
## S3 method for class 'deepspat_trivar_GP'
predict(object, newdata, type = NULL, reference = NULL, component = NULL, ...)
```

### Arguments

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| object    | the deepspat_trivar_GP object                                                             |
| newdata   | data frame containing the prediction locations                                            |
| type      | optional prediction type: latent process, response, warped coordinates, or covariance map |
| reference | reference-site index for type = "covariance"                                              |
| component | optional component index                                                                  |
| ...       | currently unused                                                                          |

### Value

predict.deepspat\_trivar\_GP returns a list with the following item

**"df\_pred"** Data frame containing the predictions/prediction intervals at the prediction locations

---

print.deepspat

*Print fitted deepspat models*


---

### Description

Lightweight display methods for fitted deepspat objects.

**Usage**

```

## S3 method for class 'deepspat'
print(x, ...)

## S3 method for class 'deepspat_GP'
print(x, ...)

## S3 method for class 'deepspat_nn_GP'
print(x, ...)

## S3 method for class 'deepspat_nn_ST_GP'
print(x, ...)

## S3 method for class 'deepspat_bivar_GP'
print(x, ...)

## S3 method for class 'deepspat_trivar_GP'
print(x, ...)

## S3 method for class 'deepspat_MSP'
print(x, ...)

## S3 method for class 'deepspat_rPP'
print(x, ...)

## S3 method for class 'summary.deepspat'
print(x, ...)

```

**Arguments**

|     |                          |
|-----|--------------------------|
| x   | a fitted deepspat object |
| ... | currently unused         |

**Value**

The input object, invisibly.

---

RBF\_block

*Radial Basis Function Warpings*


---

**Description**

Sets up a composition of radial basis functions (RBFs) for used in a deep compositional spatial model. The function sets up RBFs on a prescribed domain on a grid at a certain resolution. It returns a list containing all the functions in the single-resolution RBF unit. See Value for more details.

**Usage**

```
RBF_block(res = 1L, lims = c(-0.5, 0.5), dtype = "float32")
```

**Arguments**

|       |                                                                            |
|-------|----------------------------------------------------------------------------|
| res   | the resolution                                                             |
| lims  | the limits of one side of the square 2D domain on which to set up the RBFs |
| dtype | data type                                                                  |

**Value**

RBF\_block returns a list containing a list for each RBF in the block with the following components:

**"f"** An encapsulated function that takes an input and evaluates the RBF over some input using TensorFlow

**"fR"** Same as f but uses R

**"fMC"** Same as f but does it in parallel for several inputs index by the first dimension of the tensor

**"r"** The number of basis functions (one for each layer)

**"trans"** The transformation applied to the weights before estimation

**"fix\_weights"** Flag indicating whether the weights are fixed or not (FALSE for RBFs)

**"name"** Name of layer

**Examples**

```
if (reticulate::py_module_available("tensorflow")) {
  layer <- RBF_block(res = 1L)
}
```

---

|                   |                            |
|-------------------|----------------------------|
| set_deepspat_seed | <i>Set TensorFlow seed</i> |
|-------------------|----------------------------|

---

**Description**

Set TensorFlow seed in deepspat package

**Usage**

```
set_deepspat_seed(seed = 1L)
```

**Arguments**

|      |          |
|------|----------|
| seed | the seed |
|------|----------|

**Value**

No return value, called for side effects.

**Examples**

```
if (reticulate::py_module_available("tensorflow")) {
  set_deepspat_seed(seed = 1L)
}
```

---

|          |                                             |
|----------|---------------------------------------------|
| sim_data | <i>Generate simulation data for testing</i> |
|----------|---------------------------------------------|

---

**Description**

Generates simulated data for use in experiments

**Usage**

```
sim_data(type = "step1D", ds = 0.001, n = 300L, sigma2y = NULL)
```

**Arguments**

|         |                                                                                                                 |
|---------|-----------------------------------------------------------------------------------------------------------------|
| type    | type of function. Can be 'step1D', 'Monterrubio1D', 'dampedwave1D', 'step2D', 'AWU_RBF_2D', or 'AWU_RBF_LFT_2D' |
| ds      | spatial grid length                                                                                             |
| n       | number of data points                                                                                           |
| sigma2y | measurement-error variance                                                                                      |

**Value**

sim\_data returns a list containing the following items:

**"s"** Process locations on a fine grid with spacing ds

**"sobs"** Observation locations

**"swarped"** The warping function (when this is also simulated)

**"f\_true"** The true process on the fine grid

**"y"** The simulated observation data

**Examples**

```
if (reticulate::py_module_available("tensorflow")) {
  sim <- sim_data(type = "step1D", ds = 0.001)
}
```

---

|                  |                                         |
|------------------|-----------------------------------------|
| summary.deepspat | <i>Summarize fitted deepspat models</i> |
|------------------|-----------------------------------------|

---

**Description**

Summarizes fitted deepspat objects without performing prediction.

**Usage**

```
## S3 method for class 'deepspat'
summary(object, vcov = FALSE, ...)

## S3 method for class 'deepspat_GP'
summary(object, vcov = FALSE, ...)

## S3 method for class 'deepspat_nn_GP'
summary(object, vcov = FALSE, ...)

## S3 method for class 'deepspat_nn_ST_GP'
summary(object, vcov = FALSE, ...)

## S3 method for class 'deepspat_bivar_GP'
summary(object, vcov = FALSE, ...)

## S3 method for class 'deepspat_trivar_GP'
summary(object, vcov = FALSE, ...)

## S3 method for class 'deepspat_MSP'
summary(
  object,
  newdata = NULL,
  vcov = FALSE,
  uncAss = NULL,
  edm_emp = NULL,
  ...
)

## S3 method for class 'deepspat_rPP'
summary(
  object,
  newdata = NULL,
  vcov = FALSE,
  uncAss = NULL,
  edm_emp = NULL,
  uprime = NULL,
  ...
)
```

**Arguments**

|         |                                                                           |
|---------|---------------------------------------------------------------------------|
| object  | a fitted deepspat object                                                  |
| vcov    | logical; include stored covariance information when available             |
| ...     | additional arguments used only for legacy compatibility                   |
| newdata | legacy prediction locations for extreme-process summaries                 |
| uncAss  | legacy uncertainty flag; use <code>predict(..., se = TRUE)</code> instead |
| edm_emp | empirical dependence estimates for legacy WLS uncertainty                 |
| uprime  | legacy threshold for r-Pareto WLS uncertainty                             |

**Value**

A `summary.deepspat` object.

# Index

AFF\_1D, [2](#)  
AFF\_2D, [3](#)  
AWU, [4](#)

bisquares1D, [5](#)  
bisquares2D, [6](#)

deepspat, [6](#)  
deepspat-package (deepspat), [6](#)  
deepspat\_bivar\_GP, [9](#)  
deepspat\_GP, [11](#)  
deepspat\_MSP, [13](#)  
deepspat\_nn\_GP, [15](#)  
deepspat\_nn\_ST\_GP, [17](#)  
deepspat\_rPP, [19](#)  
deepspat\_trivar\_GP, [21](#)

init\_learn\_rates, [25](#)  
initvars, [24](#)

LFT, [26](#)

plot.deepspat, [27](#)  
plot.deepspat\_bivar\_GP (plot.deepspat),  
[27](#)  
plot.deepspat\_GP (plot.deepspat), [27](#)  
plot.deepspat\_MSP (plot.deepspat), [27](#)  
plot.deepspat\_nn\_GP (plot.deepspat), [27](#)  
plot.deepspat\_nn\_ST\_GP (plot.deepspat),  
[27](#)  
plot.deepspat\_rPP (plot.deepspat), [27](#)  
plot.deepspat\_trivar\_GP  
(plot.deepspat), [27](#)  
predict.deepspat, [29](#)  
predict.deepspat\_bivar\_GP, [30](#)  
predict.deepspat\_GP, [31](#)  
predict.deepspat\_MSP, [31](#)  
predict.deepspat\_nn\_GP, [32](#)  
predict.deepspat\_nn\_ST\_GP, [33](#)  
predict.deepspat\_rPP, [34](#)  
predict.deepspat\_trivar\_GP, [35](#)

print.deepspat, [35](#)  
print.deepspat\_bivar\_GP  
(print.deepspat), [35](#)  
print.deepspat\_GP (print.deepspat), [35](#)  
print.deepspat\_MSP (print.deepspat), [35](#)  
print.deepspat\_nn\_GP (print.deepspat),  
[35](#)  
print.deepspat\_nn\_ST\_GP  
(print.deepspat), [35](#)  
print.deepspat\_rPP (print.deepspat), [35](#)  
print.deepspat\_trivar\_GP  
(print.deepspat), [35](#)  
print.summary.deepspat  
(print.deepspat), [35](#)

RBF\_block, [36](#)

set\_deepspat\_seed, [37](#)  
sim\_data, [38](#)  
summary.deepspat, [39](#)  
summary.deepspat\_bivar\_GP  
(summary.deepspat), [39](#)  
summary.deepspat\_GP (summary.deepspat),  
[39](#)  
summary.deepspat\_MSP  
(summary.deepspat), [39](#)  
summary.deepspat\_nn\_GP  
(summary.deepspat), [39](#)  
summary.deepspat\_nn\_ST\_GP  
(summary.deepspat), [39](#)  
summary.deepspat\_rPP  
(summary.deepspat), [39](#)  
summary.deepspat\_trivar\_GP  
(summary.deepspat), [39](#)