Package ‘fitlandr’

February 10, 2023

Type Package

Title Fit Vector Fields and Potential Landscapes from Intensive Longitudinal Data

Version 0.1.0


License GPL (>= 3)


BugReports https://github.com/Sciurus365/fitlandr/issues

Imports cli, dplyr, furrr, future.apply, ggplot2, glue, grDevices, grid, magrittr, MASS, numDeriv, plotly, R.utils, Rfast, rlang, rootSolve, simlandr (>= 0.3.0), SparseVFC, tidyr

Suggests akima, colorRamps, future

Encoding UTF-8

RoxygenNote 7.2.2

NeedsCompilation no

Author Jingmeng Cui [aut, cre] (<https://orcid.org/0000-0003-3421-8457>)

Maintainer Jingmeng Cui <jingmeng.cui@outlook.com>

Repository CRAN

Date/Publication 2023-02-10 10:40:02 UTC
R topics documented:

add_interp_grid .................................................. 2
find_eqs ............................................................. 3
fit_2d_vf .............................................................. 3
fit_3d_vfId ............................................................ 5
MVKE ................................................................. 6
normalize_predict_f ............................................... 7
pathB_options ......................................................... 7
plot.vectorfield ..................................................... 8
predict.vectorfield .................................................. 10
reorder_output ....................................................... 11
simlandr_options ................................................... 11
sim_vf ................................................................. 12
sim_vf_options ......................................................... 13

Index 15

---

**add_interp_grid**  
*Add a grid to a vectorfield object to enable linear interpolation*

**Description**

Add a grid to a vectorfield object to enable linear interpolation

**Usage**

```r
add_interp_grid(vf, lims = vf$lims, n = vf$n)
```

**Arguments**

- `vf`  
  A vectorfield object estimated by `fit_2d_vf()`.

- `lims`  
  The limits of the range for the vector field estimation as c(<xl>, <xu>, <yl>, <yu>). If missing, the range of the data extended by 10% for both sides will be used.

- `n`  
  The number of equally spaced points in each axis, at which the vectors are to be estimated.

**Value**

A vectorfield project with an interp_grid field.
find_eqs

Find equilibrium points for a vector field

Description
Find equilibrium points for a vector field

Usage
find_eqs(vf, starts, jacobian_params = list(), ...)

Arguments
vf
A vectorfield object estimated by fit_2d_vf().

starts
A vector indicating the starting value for solving the equilibrium point, or a list
of vectors providing multiple starting values together.

jacobian_params
Parameters passed to numDeriv::jacobian().

...
Parameters passed to rootSolve::multiroot().

Value
A list of equilibrium points and their details. Use print.vectorfield_eqs() to inspect it.

fit_2d_vf
Estimate a 2D vector field

Description
Estimate a 2D vector field from intensive longitudinal data. Two methods can be used: Multi-
variate Vector Field Kernel Estimator (MVKE, using MVKE()), or Sparse Vector Field Consensus
(SparseVFC, using SparseVFC::SparseVFC()). Note that the input data are automatically nor-
malized before being sent to the estimation engines to make sure the default parameter settings are
close to the optimal. Therefore, you do not need to scale up or down the parameters of MVKE()
or SparseVFC::SparseVFC(). We suggest the MVKE method to be used for psychological data
because it has more realistic assumptions and produces more reasonable output.

Usage
fit_2d_vf(
  data,
  x,
  y,
  lims,
  n = 20,
```r
vector_position = "start",
na_action = "omit_data_points",
method = c("MVKE", "MVKE"),
...
)

Arguments

data The data set used for estimating the vector field. Should be a data frame or a matrix.
x, y Characters to indicate the name of the two variables.
lims The limits of the range for the vector field estimation as `c(<xl>, <xu>, <yl>, <yu>)`. If missing, the range of the data extended by 10% for both sides will be used.
n The number of equally spaced points in each axis, at which the vectors are to be estimated.
vector_position Only useful if `method == "VFC"`. One of "start", "middle", or "end", representing the position of the vectors. If "start", for example, the starting point of a vector is regarded as the position of the vector.
na_action One of "omit_data_points" or "omit_vectors". If using "omit_data_points", then only the NA points are omitted, and the points before and after an NA will form a vector. If using "omit_vectors", then the vectors will be omitted if either of its points is NA.
method One of "MVKE" or "VFC".
... Other parameters to be passed to `MVKE()` or `SparseVFC::SparseVFC()`.

Value

A `vectorfield` object.

See Also

`plot.vectorfield()`

Examples

# generate data
single_output_grad <- simlandr::sim_fun_grad(length = 200, seed = 1614)
# fit the vector field
v2 <- fit_2d_vf(single_output_grad, x = "x", y = "y", method = "MVKE")
plot(v2)
```
Description

Two methods are available: method = "pathB" and method = "simlandr". See Details section.

Usage

```r
fit_3d_vfld(
  vf,
  method = c("simlandr", "pathB"),
  .pathB_options = pathB_options(vf),
  .sim_vf_options = sim_vf_options(vf),
  .simlandr_options = simlandr_options(vf),
  linear_interp = FALSE
)
```

Arguments

- `vf`: A `vectorfield` object estimated by `fit_2d_vf()`.
- `method`: The method used for landscape construction. Can be `pathB` or `simlandr`.
- `.pathB_options`: Only for `method = "pathB"`. Options controlling the path-integral algorithm. Should be generated by `sim_vf_options()`.
- `.sim_vf_options`: Only for `method = "simlandr"`. Options controlling the vector field simulation. Should be generated by `sim_vf_options()`.
- `.simlandr_options`: Only for `method = "simlandr"`. Options controlling the landscape construction. Should be generated by `simlandr_options()`.
- `linear_interp`: Use linear interpolation method to estimate the drift vector (and the diffusion matrix). This can speed up the calculation. If `TRUE`, be sure that a linear grid was calculated for the vector field using `<vf> <- add_interp_grid(<vf>)`.

Details

For method = "simlandr", the landscape is constructed based on the generalized potential landscape by Wang et al. (2008), implemented by the `simlandr` package. This function is a wrapper of `sim_vf()` and `simlandr::make_3d_static()`. Use those two functions separately for more customization.

For method = "pathB", the landscape is constructed based on the deterministic path-integral quasi-potential defined by Bhattacharya et al. (2011).

We recommend the `simlandr` method for psychological data because it is more stable.

Parallel computing based on `future` is supported for both methods. Use `future::plan("multisession")` to enable this and speed up computation.
MVKE

Multivariate vector field kernel estimator

Description

See references for details.

Usage

MVKE(d, h = 0.2, kernel = c("exp", "Gaussian"))

Arguments

d
   The dataset. Should be a matrix or a data frame, with each row representing a
   random vector.

h
   The bandwidth for the kernel estimator.

kernel
   The type of kernel estimator used. "exp" by default (exp()), and if "Gaussian"
   then stats::dnorm() will be used.

Value

A function(x), which then returns the $\mu$ and $a$ estimators at the position $x$. 

Examples

# generate data
single_output_grad <- simlandr::sim_fun_grad(length = 200, seed = 1614)
# fit the vector field
v2 <- fit_2d_vf(single_output_grad, x = "x", y = "y", method = "MVKE")
plot(v2)
# fit the landscape
future::plan("multisession")
set.seed(1614)
l2 <- fit_3d_vfld(v2,
   .sim_vf_options = sim_vf_options(chains = 16, stepsize = 1, forbid_overflow = TRUE),
   .simlandr_options = simlandr_options(adjust = 5, Umax = 4))
plot(l2, 2)
future::plan("sequential")

Value

A landscape object as described in simlandr::make_3d_static(), or a 3d_static_landscape_B
object, which inherits from the landscape class and contains the following elements: dist, the dis-
tribution estimation for landscapes; plot, a 3D plot using plotly; plot_2, a 2D plot using ggplot2;
x, y, from vf.

Examples

# generate data
single_output_grad <- simlandr::sim_fun_grad(length = 200, seed = 1614)
# fit the vector field
v2 <- fit_2d_vf(single_output_grad, x = "x", y = "y", method = "MVKE")
plot(v2)
# fit the landscape
future::plan("multisession")
set.seed(1614)
l2 <- fit_3d_vfld(v2,
   .sim_vf_options = sim_vf_options(chains = 16, stepsize = 1, forbid_overflow = TRUE),
   .simlandr_options = simlandr_options(adjust = 5, Umax = 4))
plot(l2, 2)
future::plan("sequential")

MVKE

Multivariate vector field kernel estimator
References


normalize_predict_f

Return a normalized prediction function

Description

Return a normalized prediction function

Usage

normalize_predict_f(vf)

Arguments

vf A vectorfield object estimated by fit_2d_vf().

Value

A function that takes a vector x and returns a list of v, the drift part, and a, the diffusion part.

pathB_options

Options controlling the path-integral algorithm

Description

See path_integral_B(), align_pot_B() for details.

Usage

pathB_options(
  vf,
  lims = rlang::expr(vf$lims),
  n_path_int = 20,
  stepsize = 0.01,
  tol = 0.01,
  numTimeSteps = 1400,
  n = 200,
  digits = 2,
  linear = TRUE,
  ...
)
Arguments

- **vf**: A *vectorfield* object estimated by `fit_2d_vf()`.
- **lims**: The limits of the range for the estimation as `c(xl, xu, yl, yu)`.
- **n_path_int**: The number of equally spaced points in each axis, at which the path integrals is to be calculated.
- **stepsize**: The stepsize for Euler–Maruyama simulation of the system.
- **tol**: The tolerance to test convergence.
- **numTimeSteps**: Number of time steps for integrating along each path (to ensure uniform arrays). Choose high-enough number for convergence with given stepsize.
- **n**: The number of equally spaced points in each axis, at which the landscape is to be estimated.
- **digits**: Currently, the raw sample points in some regions are too dense that may crashes interpolation. To avoid this problem, only one point of all with the same first several digits is kept. Use this parameter to indicate how many digits are considered. Note that this is a temporary solution and might be changed in the near future.
- **linear**: logical – indicating whether linear or spline interpolation should be used.
- **...**: Not in use.

Value

A list containing the parameters of the corresponding function. Only intended to be used within `fit_3d_vfld()`

---

**plot.vectorfield**

*Plot a 2D vector field*

Description

Plot a 2D vector field estimated by `fit_2d_vf()`. Powered by `ggplot2::ggplot()`.

Usage

```r
# S3 method for class 'vectorfield'
plot(
x,
arrow = grid::arrow(length = grid::unit(0.1, "cm")),
show_estimated_vector = TRUE,
estimated_vector_enlarge = 1,
estimated_vector_options = list(),
show_point = TRUE,
point_options = list(size = 0.5),
show_original_vector = FALSE,
original_vector_enlarge = 1,
```
original_vector_options = list(),
show_used_vector = FALSE,
used_vector_options = list(color = "red"),
show_v_norm = FALSE,
v_norm_options = list(),
...
)

Arguments

x  A vectorfield object estimated by fit_2d_vf().
arrow  The description of the arrow heads of the vectors on the plot (representing the vector field). Generated by grid::arrow(). Also see the arrow parameter of ggplot2::geom_segment().
show_estimated_vector  Show the vectors from the estimated model? TRUE by default.
estimated_vector_enlarge  A number. How many times should the vectors (representing the estimated vector field) be enlarged on the plot? This can be useful when the estimated vector field is too strong or too weak.
estimated_vector_options  A list passing other customized parameters to ggplot2::geom_segment() to control the vectors representing the estimated vector field.
show_point  Show the original data points? TRUE by default.
point_options  A list passing other customized parameters to ggplot2::geom_point() to control the points representing the original data point.
show_original_vector  Show the original vectors (i.e., the vectors between data points)? FALSE by default.
original_vector_enlarge  A number. How many times should the original vectors be enlarged on the plot?
original_vector_options  A list passing other customized parameters to ggplot2::geom_segment() to control the vectors representing the original data.
show_used_vector  Only for vector fields estimated by the "VFC" method. Should the vectors from the original data that are considered inliers be specially marked? FALSE by default.
used_vector_options  Only for vector fields estimated by the "VFC" method. A list passing other customized parameters to ggplot2::geom_segment() to control the vectors representing the inliers. Red by default.
show_v_norm  Show the norm of the estimated vectors (the strength of the vector field)? FALSE by default.
v_norm_options  A list passing other customized parameters to ggplot2::geom_raster() to control the layer representing the norm of the estimated vectors.
...  Not in use.
predict.vectorfield

Calculate the vector value at a given position

Description

Calculate the vector value at a given position

Usage

## S3 method for class 'vectorfield'
predict(object, pos, linear_interp = FALSE, calculate_a = TRUE, ...)

Arguments

- `object`: A vectorfield project generated by `fit_2d_vf()`.
- `pos`: A vector, the position of the vector.
- `linear_interp`: Use linear interpolation method to estimate the drift vector (and the diffusion matrix). This can speed up the calculation. If TRUE, be sure that a linear grid was calculated for the vector field using `<vf> <- add_interp_grid(<vf>)`.
- `calculate_a`: Effective when `linear_interp == TRUE`. Do you want to calculate the diffusion matrix? Use FALSE can save some time.
- `...`: Not in use.

Value

A list of `v`, the drift part that is used for vector fields, and `a` (when `calculate_a == TRUE`), the diffusion part at a given position.

See Also

`add_interp_grid()`
reorder_output

Reorder a simulation output in time order

Description

Then `simlandr::check_conv()` can be used meaningfully.

Usage

`reorder_output(s, chains)`

Arguments

- `s` A simulation output, possibly generated by `sim_vf()`
- `chains` How many chains simulations should be performed?

Value

A reordered matrix of the simulation output.

simlandr_options

Options controlling the landscape construction

Description

To control the behavior of `simlandr::make_3d_static()`, but with default values accommodated for `fitlandr`. See `simlandr::make_3d_static()` for details.

Usage

```
simlandr_options(
  vf,
  x = rlang::expr(vf$x),
  y = rlang::expr(vf$y),
  lims = rlang::expr(vf$lims),
  kde_fun = c("ks", "MASS"),
  n = 200,
  adjust = 1,
  h,
  Umax = 5
)
```
Arguments

- **vf**: A vectorfield object estimated by `fit_2d_vf()`.
- **x, y**: The names of the target variables.
- **lims**: The limits of the range for the density estimator as \( c(x_l, x_u) \) for 2D landscapes, \( c(x_l, x_u, y_l, y_u) \) for 3D landscapes, \( c(x_l, x_u, y_l, y_u, z_l, z_u) \) for 4D landscapes. If missing, the range of the data extended by 10% for both sides will be used. For landscapes based on multiple simulations, the largest range of all simulations (which means the lowest lower limit and the highest upper limit) will be used by default.
- **kde_fun**: Which kernel estimator to use? Choices: "ks" `ks::kde()` (default; faster and using less memory); "base" `base::density()` (only for 2D landscapes); "MASS" `MASS::kde2d()` (only for 3D landscapes).
- **n**: The number of equally spaced points in each axis, at which the density is to be estimated.
- **adjust**: The multiplier to the bandwidth. The bandwidth used is actually \( \text{adjust} \times h \). This makes it easy to specify values like "half the default" bandwidth.
- **h**: A number, or possibly a vector for 3D and 4D landscapes, specifying the smoothing bandwidth to be used. If missing, the default value of the kernel estimator will be used (but \( \text{bw} = "SJ" \) for `base::density()`). Note that the definition of bandwidth might be different for different kernel estimators. For landscapes based on multiple simulations, the largest \( h \) of all simulations will be used by default.
- **Umax**: The maximum displayed value of potential.

Value

A list containing the parameters of the corresponding function. Only intended to be used within `fit_3d_vfld()`

---

**sim_vf**

*Simulation from vector fields*

Description

Parallel computing based on `future` is supported. Use `future::plan("multisession")` to enable this.

Usage

```r
sim_vf(
  vf,
  noise = 1,
  noise_warmup = noise,
  chains = 10,
```
Arguments

vf  A vectorfield object estimated by `fit_2d_vf()`.
noise  Relative noise of the simulation. Set this smaller when the simulation is unstable (e.g., when the elements in the diffusion matrix are not finite), and set this larger when the simulation converges too slowly.
noise_warmup  The noise used for the warming-up period.
chains  How many chains simulations should be performed?
length  The simulation length for each chain.
discard  How much of the starting part of each chain should be discarded? (Warming-up period.)
stepsize  The stepsize for Euler–Maruyama simulation of the system.
sparse  A number. How much do you want to sparse the output? When the noise is small, sparse the output may make the density estimation more efficient.
forbid_overflow  If TRUE, when the simulated system runs out of the margins specified in vf, the system will be moved back to the previous value. This can help to stabilize the simulation. FALSE by default.
linear_interp  Use linear interpolation method to estimate the drift vector (and the diffusion matrix). This can speed up the calculation. If TRUE, be sure that a linear grid was calculated for the vector field using `<vf> <- add_interp_grid(<vf>)`.
inits  The initial values of each chain.

Value

A matrix of the simulated data.

---

**sim_vf_options**

*Options controlling the vector field simulation*

**Description**

See `sim_vf()` for details.
Usage

```r
sim_vf_options(
  vf,
  noise = 1,
  noise_warmup = noise,
  chains = 10,
  length = 10000,
  discard = 0.3,
  stepsize = 0.01,
  sparse = 1,
  forbid_overflow = FALSE,
  inits = rlang::expr(matrix(c(stats::runif(chains, min = vf$lims[1], max = vf$lims[2]),
                              stats::runif(chains, min = vf$lims[3], max = vf$lims[4])), ncol = 2))
)
```

Arguments

- `vf`: A vectorfield object estimated by `fit_2d_vf()`.
- `noise`: Relative noise of the simulation. Set this smaller when the simulation is unstable (e.g., when the elements in the diffusion matrix are not finite), and set this larger when the simulation converges too slowly.
- `noise_warmup`: The noise used for the warming-up period.
- `chains`: How many chains simulations should be performed?
- `length`: The simulation length for each chain.
- `discard`: How much of the starting part of each chain should be discarded? (Warming-up period.)
- `stepsize`: The stepsize for Euler–Maruyama simulation of the system.
- `sparse`: A number. How much do you want to sparse the output? When the noise is small, sparse the output may make the density estimation more efficient.
- `forbid_overflow`: If TRUE, when the simulated system runs out of the margins specified in `vf`, the system will be moved back to the previous value. This can help to stabilize the simulation. FALSE by default.
- `inits`: The initial values of each chain.

Value

A list containing the parameters of the corresponding function. Only intended to be used within `fit_3d_vfld()`
## Index

<table>
<thead>
<tr>
<th>Function/Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_interp_grid()</td>
<td>2, 10</td>
</tr>
<tr>
<td>add_interp_grid()()</td>
<td></td>
</tr>
<tr>
<td>align_pot_B()</td>
<td>7</td>
</tr>
<tr>
<td>exp()</td>
<td>6</td>
</tr>
<tr>
<td>find_eqs()</td>
<td>3</td>
</tr>
<tr>
<td>fit_2d_vf()</td>
<td>2, 3, 5, 7–10, 12–14</td>
</tr>
<tr>
<td>fit_2d_vf()()</td>
<td></td>
</tr>
<tr>
<td>fit_3d_vfld()</td>
<td>5</td>
</tr>
<tr>
<td>fit_3d_vfld()()()</td>
<td>8, 12, 14</td>
</tr>
<tr>
<td>ggplot2::geom_point()</td>
<td>9</td>
</tr>
<tr>
<td>ggplot2::geom_raster()</td>
<td>9</td>
</tr>
<tr>
<td>ggplot2::geom_segment()</td>
<td>9</td>
</tr>
<tr>
<td>ggplot2::ggplot()</td>
<td>8</td>
</tr>
<tr>
<td>grid::arrow()</td>
<td>9</td>
</tr>
<tr>
<td>ks::kde()</td>
<td>12</td>
</tr>
<tr>
<td>MASS::kde2d()</td>
<td>12</td>
</tr>
<tr>
<td>MVKE()</td>
<td>6</td>
</tr>
<tr>
<td>MVKE()()</td>
<td>3, 4</td>
</tr>
<tr>
<td>normalize_predict_f()</td>
<td>7</td>
</tr>
<tr>
<td>numDeriv::jacobian()()</td>
<td>3</td>
</tr>
<tr>
<td>path_integral_B()</td>
<td>7</td>
</tr>
<tr>
<td>pathB_options()</td>
<td>7</td>
</tr>
<tr>
<td>plot.vectorfield()</td>
<td>8</td>
</tr>
<tr>
<td>plot.vectorfield()()</td>
<td>4</td>
</tr>
<tr>
<td>predict.vectorfield()()</td>
<td></td>
</tr>
<tr>
<td>predict.vectorfield()()()</td>
<td>10</td>
</tr>
<tr>
<td>reorder_output()</td>
<td>11</td>
</tr>
<tr>
<td>rootSolve::multiroot()()</td>
<td>3</td>
</tr>
<tr>
<td>sim_vf()</td>
<td>12</td>
</tr>
<tr>
<td>sim_vf()()</td>
<td>5, 11, 13</td>
</tr>
<tr>
<td>sim_vf_options()</td>
<td>13</td>
</tr>
<tr>
<td>sim_vf_options()()()</td>
<td>5</td>
</tr>
<tr>
<td>simlandr::check_conv()()</td>
<td>11</td>
</tr>
<tr>
<td>simlandr::make_3d_static()()</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>simlandr_options()</td>
<td>11</td>
</tr>
<tr>
<td>SparseVFC::SparseVFC()()</td>
<td>3, 4</td>
</tr>
<tr>
<td>stats::dnorm()</td>
<td>6</td>
</tr>
</tbody>
</table>