

# Package ‘gLVInterNetworks’

December 19, 2016

**Type** Package

**Title** Inference of interaction networks based on generalised Lotka Volterra dynamics

**Version** 0.1

**Date** 2016-11-08

**Author** Lukas Hirsch, Florian Centler

**Maintainer** Lukas Hirsch <lukashirsch@gmail.com>

**Description** Inference of interaction networks based on the parameterization of generalized Lotka Volterra models on timeseries data

**Imports** deSolve, MASS, glmnet, FME, igraph, minpack.lm, coda, minqa, rootSolve

**License** GPL-2

## R topics documented:

gLVInterNetworks-package . . . . .	1
gLVInterNetworks-package . . . . .	2
generate_data . . . . .	3
gLVlinearRegression . . . . .	4
gLVnonlinearRegression . . . . .	5
plot.Sim_data . . . . .	6
plotGraph . . . . .	6
sensitivityAnalysis . . . . .	7

<b>Index</b>	<b>8</b>
--------------	----------

---

gLVInterNetworks-package

*Inference of interaction networks based on generalised Lotka Volterra dynamics*

---

## Description

Inference of interaction networks based on the parameterization of generalized Lotka Volterra models on timeseries data

**Details**

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

~~ An overview of how to use the package, including the most important functions ~~

**Author(s)**

Lukas Hirsch, Florian Centler Maintainer: Lukas Hirsch <lukashirsch@gmail.com>

**References**

~~ Literature or other references for background information ~~

**Examples**

```
library(gLVIInterNetworks)
data <- generate_data(species = 2, number_of_non_diagonal_coefficients = 2, timepoints = 100, noise = 0.01, t
## Not run: plot(data, type = "l")
lr <- gLVlinearRegression(data, regularization = TRUE, alpha = 0)
## Not run: summary(lr)
## Not run: plot(lr, type = "l")
## Not run: points(data)
nlr <- nongLVlinearRegression(data, parms0 = lr$Parms)
## Not run: summary(nlr)
## Not run: plot(nlr, type = "l")
## Not run: points(data)
## Not run: par(mfrow = c(1,2))
## Not run: plotGraph(data, vsize = 0.2, main = "Original interaction network", verbose = TRUE )
## Not run: plotGraph(nlr, vsize = 0.2, main = "Inferred interaction network", verbose = TRUE)
ident <- sensitivityAnalysis(nlr$Parms)
## Print summary of sensitivity matrix
summary(ident$sens)
## Print collinearity index for all parameters together
ident$coll[ident$coll[, "N"] == length(data$Parms),]
```

---

gLVIInterNetworks-package

*Inference of interaction networks based on generalised Lotka Volterra dynamics*

---

**Description**

Inference of interaction networks based on the parameterization of generalized Lotka Volterra models on timeseries data

**Details**

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

~~ An overview of how to use the package, including the most important functions ~~

**Author(s)**

Lukas Hirsch, Florian Centler Maintainer: Lukas Hirsch <lukashirsch@gmail.com>

**References**

~~ Literature or other references for background information ~~

**Examples**

```
library(gLVInterNetworks)
data <- generate_data(species = 2, number_of_non_diagonal_coefficients = 2, timepoints = 100, noise = 0.01, testData = 10)
## Not run: plot(data, type = "l")
lr <- gLVlinearRegression(data, regularization = TRUE, alpha = 0)
## Not run: summary(lr)
## Not run: plot(lr, type = "l")
## Not run: points(data)
nlr <- gLVnonlinearRegression(data, parms0 = lr$Parms)
## Not run: summary(nlr)
## Not run: plot(nlr, type = "l")
## Not run: points(data)
## Not run: par(mfrow = c(1,2))
## Not run: plotGraph(data, vsize = 0.2, main = "Original interaction network", verbose = TRUE )
## Not run: plotGraph(nlr, vsize = 0.2, main = "Inferred interaction network", verbose = TRUE)
```

---

generate\_data

*Generate random data for gLV fitting*

---

**Description**

Generates random data simulating time series of cell abundances governed by Lotka Volterra dynamics

**Usage**

```
generate_data(species, number_of_non_diagonal_coefficients, timepoints, noise, testData)
```

**Arguments**

species	Integer describing the number of independent cellular subcommunities
number_of_non_diagonal_coefficients	Integer describing the number of non-zero interactions present in the simulated system. These are assigned randomly between the nodes or subcommunities
timepoints	Numeric vector containing the timepoints for which to compute the solutions of the model
noise	The standard deviation of the normally distributed stochastic factor added to the solution of the model at each time step
testData	Number of observations used as test dataset for validation on untrained data, taken from the last measurements of the time series.

**Value**

Returns a matrix. The first column displays the time points, and the remaining columns correspond each to a independent variable in the system.

**Author(s)**

Lukas Hirsch

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (species, number_of_non_diagonal_coefficients, timepoints,
         noise)
{
  "requires inSilico_bio, discrete, addNoise_res"
  raw_data <- inSilico_bio(species, number_of_non_diagonal_coefficients)
  Parms <- raw_data[[2]]
  res <- raw_data[[1]]
  threshold = 3
  if (any(which(round(diff(abs(rowMeans(res[, -1])), 2), threshold) ==
    0))) {
    end <- min(which(round(diff(abs(rowMeans(res[, -1])),
      2), threshold) == 0))
  }
  else {
    end <- nrow(res)
  }
  res <- solveLV_bio(Parms, seq(0, end, by = 0.01), res[1,
    -1])
  discretization <- nrow(res)/timepoints
  res <- discrete(res, discretization)
  test_data <- res[(nrow(res) - 19):nrow(res), ]
  obs <- res[1:(nrow(res) - 20), ]
  obs <- addNoise_res(obs, noise)
  timepoints <- nrow(obs)
  dimensions <- ncol(obs[, -1])
  k <- sum(any(Parms == 0))/length(Parms)
  data <- list(species = species, timepoints = timepoints,
    Parms = Parms, noise = noise, sparsity = k, obs = obs,
    testData = test_data)
  class(data) <- "Sim_data"
  return(data)
}
```

---

gLVlinearRegression      *Parameter estimation of algebraic linear discrete gLV model*

---

**Description**

Given multivariatic time series data, this function fits a linear and discrete generalized Lotka Volterra model of the form  $\Delta x_i = \alpha_i + \sum \beta_{ij} * x_j$

**Usage**

```
glVlinearRegression(data, regularization = FALSE, alpha = 0)
```

**Arguments**

data	Matrix or table containing time series of measurements in longitudinal form where first column corresponds to the time points and subsequent columns correspond to each model variable
regularization	Boolean flag if regularization of the parameter matrix should be forced
alpha	Regularization parameter for the elastic net. It ranges from 0 (= Ridge regression) to 1 (= LASSO regression) with values in between corresponding to both L1 and L2 penalties weighted by alpha

**Details**

Some theory and formulas on elastic net

---

```
glVnonlinearRegression
```

*Parameter estimation through gradient search of continuous nonlinear gLV model*

---

**Usage**

```
glVnonlinearRegression(data, parms0 = NULL, ftol = 1e-8 , ptol = 1e-8, maxiter = 100, lowerbound =
```

**Arguments**

data	Data input containing a time series of observations in longitudinal matrix form
parms0	Optional. Starting parameter vector. Default = Zero vector
ftol	Objective function output tolerance before stopping iterative optimization
ptol	Parameter change tolerance in output of objective function
maxiter	Maximal number of iterations allowed before breaking the gradient search algorithm
lowerbound	Numerical vector of equal length as parameter vector describing lower bound for constrained parameter search
upperbound	Numerical vector of equal length as parameter vector describing upper bound for constrained parameter search
method	Method used for optimization of the objective function. Default is "Marq" for Leverberg-Marquandt

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

---

plot.Sim_data	<i>Plot function for objects returned by in silico data generation function</i>
---------------	---

---

**Usage**

```
plot.Sim_data(x, legend = FALSE, ...)
```

**Arguments**

x	Object of class Sim_data as returned by function generate_data()
legend	Set to TRUE to place a basic legend in the topright of the plot
...	

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (x, ...)
{
  matplot(x$obs[, -1])
}
```

---

plotGraph	<i>Plot interaction network</i>
-----------	---------------------------------

---

**Description**

Plots a graph representing the interaction network described by the data

**Usage**

```
plotGraph(x, vsize = 0.1, main = NULL, verbose = FALSE, keepNames = FALSE, ...)
```

**Arguments**

x	Object containing parameter matrix in form of x\$Parms
vsize	Integer inversely proportional to the size of the nodes
main	Title of the plot
verbose	Include edge values in output network
keepNames	Set to TRUE if plotted Network should keep the names of the variables as given in the observations table. Default = FALSE
...	

**Value**

igraph object

**Author(s)**

Lukas Hirsch

**Examples**

```
data <- generate_data(2,2,100,0.1, 20)
#plotGraph(data)
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

---

sensitivityAnalysis	<i>Compute parameter correlations and multicollinearity for the model output</i>
---------------------	--

---

**Description**

This function is a wrapper function for sensFun and Collin from package FME. It calculates both a sensitivity matrix  $S_{ij}$  and multicollinearity index for all parameter combinations.

**Usage**

```
sensitivityAnalysis(Parms)
```

**Arguments**

Parms	Numeric vector or matrix with the parameter coefficients to test
-------	--

**Value**

List containing:

sens	Matrix containing sensitivity output values for each parameter and each model variable. The sensitivity matrix $S_{ij}$ contains elements $dy_i/dpar_j * parscale_j / varscale_i$ . The scale used to change the value of each parameter can be seen using the summary function on the sens table, and it is set to be the same value of the parameter itself
coll	Table with collinearity index for each possible parameter subset.

**Warning**

The function needs the original data as a global variable called "data". Please make sure that when using this function to perform identifiability analysis on a set of parameters, that the observation matrix or table used to estimate the parameter set is accessible under the name "data"

**Note**

For details in output interpretation see package FME

# Index

- \*Topic **\textasciitildekw1**
  - generate\_data, [3](#)
- \*Topic **\textasciitildekw2**
  - generate\_data, [3](#)
  - glVlinearRegression, [4](#)
  - glVnonlinearRegression, [5](#)
  - plotGraph, [6](#)
  - sensitivityAnalysis, [7](#)
- \*Topic **package**
  - glVInterNetworks-package, [1](#), [2](#)

generate\_data, [3](#)  
glVInterNetworks

- (glVInterNetworks-package), [1](#), [2](#)

glVInterNetworks-package, [1](#), [2](#)  
glVlinearRegression, [4](#)  
glVnonlinearRegression, [5](#)

plot.Sim\_data, [6](#)  
plotGraph, [6](#)

sensitivityAnalysis, [7](#)