Package 'ifit'

May 8, 2026

Type Package

Title Simulation-Based Fitting of Parametric Models with Minimum Prior Information

Version 1.0.0

Description Implements an algorithm for fitting a generative model with an intractable likelihood using only box constraints on the parameters. The implemented algorithm consists of two phases. The first phase (global search) aims to identify the region containing the best solution, while the second phase (local search) refines this solution using a trust-region version of the Fisher scoring method to solve a quasi-likelihood equation. See Guido Masarotto (2025) <doi:10.48550/arXiv.2511.08180> for the details of the algorithm and supporting results.

License MIT + file LICENSE

Encoding UTF-8

LinkingTo Rcpp

Imports Rcpp, lpSolve, parallel, splines, stats, graphics

Suggests lattice

Depends R (>= 3.5.0)

LazyData TRUE

RoxygenNote 7.3.3

NeedsCompilation yes

Author Guido Masarotto [aut, cre] (ORCID: <https://orcid.org/0000-0003-4697-1606>)

Maintainer Guido Masarotto <guido.masarotto@unipd.it>

Repository CRAN

Date/Publication 2025-11-20 15:30:02 UTC

Contents

`ifit-package` *ifit-package*

Description

The main package function `ifit` can be used to fit complex parametric models from which observations can be simulated. The implemented algorithm integrates a global search phase and a local search phase. It can hence be used when only scarce prior information is available.

Author(s)

Maintainer: Guido Masarotto <guido.masarotto@unipd.it> (ORCID)

References

Guido Masarotto (2015) 'Simulation-Based Fitting of Intractable Models via Sequential Sampling and Local Smoothing', arXiv eprint 2511.08180, pp. 1-23, doi:10.48550/arXiv.2511.08180

`enzyme-model` *Michaelis-Menten enzyme kinetics*

Description

Function `enzymeSim` (written in C++) simulates a realization from the stochastic Michaelis-Menten kinetics model; function `enzymeStat` computes a possible summary statistics.

Usage

```
enzymeSim(theta, n = 50L, E0 = 100L, S0 = 100L)

enzymeStat(y, degree = 2, knots = 0.2)
```

Arguments

`theta`	a vector of length 3 containing the model parameters
`n`	the process is sampled on n equispaced points on the unit interval (default 50)
`E0`, `S0`	initial state of the process (default E0=100, S0=100)
`y`	a `n x 4` matrix containing the observed data
`degree`	the degree of the B-splines basis (default 2)
`knots`	the knots of the B-splines basis (default: only one knot equal to 0.2)

Details

The model describes a simple case of enzyme kinetics. It involves an enzyme E binding to a substrate S to form a complex C. This complex C then releases a product P while simultaneously regenerating the original enzyme E. The possible reactions are `E + S -> C`, `C -> E + S`, and `C -> P + E`, with rates that constitute the three model parameters.

In probabilistic terms, the integer-valued vector $(E_t, S_t, C_t, P_t)'$ is generated by a continuous-time Markov process such that:

$$\begin{aligned}
&Pr\{E_{t+\delta} = E_t - 1, S_{t+\delta} = S_t - 1, C_{t+\delta} = C_t + 1, P_{t+\delta} = P_t | E_t, S_t, C_t, P_t\} \\
&\quad = \theta_1 E_t S_t \delta + o(\delta), \\
&Pr\{E_{t+\delta} = E_t + 1, S_{t+\delta} = S_t + 1, C_{t+\delta} = C_t - 1, P_{t+\delta} = P_t | E_t, S_t, C_t, P_t\} \\
&\quad = \theta_2 C_t \delta + o(\delta), \\
&Pr\{E_{t+\delta} = E_t + 1, S_{t+\delta} = S_t, C_{t+\delta} = C_t - 1, P_{t+\delta} = P_t + 1 | E_t, S_t, C_t, P_t\} \\
&\quad = \theta_3 C_t \delta + o(\delta).
\end{aligned}$$

The initial state is $(E_0, S_0, C_0, P_0)' = (E0, S0, 0, 0)'$. Process is simulated using Gillespie exact algorithm.

Value

`enzymeSim` returns a 'n x 4' integer matrix; the ith row contains the simulated values for E, S, C and P at time t=(i-1)/(n-1).

`enzymeStat` returns a numeric vector of length 2*(degree+1+length(knots)) containing the coefficents of the B-splines curves (with degree and knots given by the second and third arguments) fitted by least squares to the complex and product trajectories (i.e., to the last two components of the process's state).

Note

The summary statistics is based only on C_t and P_t. Indeed, at every time point t, $E_t = E0 - C_t$ and $S_t = S0 - C_t - P_t$.

References

Darren J. Wilkinson (2019) Stochastic Modelling for Systems Biology, Third edition, CRC Press

Examples

```
set.seed(1)
theta <- c(0.5, 2.5, 1)
y <- enzymeSim(theta)
plot(ts(y, start = 0, frequency = 50), main = "")
print(tobs <- enzymeStat(y))

# It takes some time to run
tsim <- function(theta) enzymeStat(enzymeSim(theta))
m <- ifit(tobs, tsim, l = rep(0, 3), u = rep(50, 3), trace = 1000)
m
```

```
confint(m)
diagIFIT(m)
numsimIFIT(m)
```

`ifit` *Simulation-based (indirect) fitting of parametric models*

Description

Fits an untractable parametric model by simulation

Usage

```
ifit(
  tobs,
  tsim,
  l,
  u,
  cluster = NULL,
  export = NULL,
  trace = 0,
  Ntotal = 1e+06,
  NTotglobal = 20000,
  Ninit = 1000,
  Nelite = 100,
  Aelite = 0.5,
  Tolglobal = 0.1,
  Tollocal = 1,
  Tolmodel = 1.5,
  NFitlocal = 4000,
  NAddglobal = 100,
  NAddlocal = 10,
  Lambda = 0.1,
  Rhomax = 0.1
)
```

Arguments

`tobs`	A vector containing the observed summary statistics.
`tsim`	A function implementing the model to be simulated. It must take as arguments a vector of model parameter values and it must return a vector of summary statistics.
`l, u`	Two numeric vectors of length equal to the number of the parameters containing the lower and upper bounds on the parameters.

cluster	if not null, the model will be simulated in parallel; the argument can be either a cluster object (as defined in package parallel) or a positive integer (in this case the cluster will be created and deleted at the end by ifit).
export	a vector giving the names of the global objects (tipically needed by function tsim) which should be exported to the cluster; not used if cluster is null.
trace	An integer value; if greater than zero, results will be printed (approximately) every trace model simulations.
Ntotal	The maximum number of allowed model simulations.
NTotglobal	The maximum number of simulations performed during the global search phase.
Ninit, Nelite, Aelite, Tolglobal, Tollocal, Tolmodel, NFitlocal, NAddglobal, NAddlocal, Rhomax, Lambda	Constants affecting the details of the algorithm (see the vignette describing the algorithm)

Value

An object of class ifit for which a suitable print method exists and whose elements can be accessed using the functions described in ifit-methods

References

Guido Masarotto (2015) 'Simulation-Based Fitting of Intractable Models via Sequential Sampling and Local Smoothing', arXiv eprint 2511.08180, pp. 1-23, doi:10.48550/arXiv.2511.08180

Examples

```
# A toy model
set.seed(1)
n <- 3
y <- rnorm(n, 1)
tobs <- mean(y)
tsim <- function(theta) mean(rnorm(n, theta[1]))
m <- ifit(tobs, tsim, l = -3, u = 3)
m
coef(m)
confint(m)
globalIFIT(m)
numsimIFIT(m)
vcov(m, "parameters")
vcov(m, "statistics")
jacobianIFIT(m)
diagIFIT(m, plot = FALSE)
estfunIFIT(m)


# Logit model
# It takes some time to run
n <- 100
X <- seq(-1, 1, length = n)
Z <- rnorm(n)
```

```
X <- cbind(1, X, Z, Z + rnorm(n))
logitSim <- function(theta) rbinom(n, 1, 1 / (1 + exp(-X %*% theta)))
logitStat <- function(y) as.numeric(crossprod(X, y))
theta <- c(-1, 1, 0.5, -0.5)
y <- logitSim(theta)
m <- ifit(
    tobs = logitStat(y), tsim = function(t) logitStat(logitSim(t)),
    l = rep(-5, 4), u = rep(5, 4), trace = 1000
)
m
g <- glm(y ~ X - 1, family = binomial)
rbind(
    ifit.estimates = coef(m),
    glm.estimates = coef(g),
    ifit.se = sqrt(diag(vcov(m))),
    glm.se = sqrt(diag(vcov(g)))
)
```

---

ifit-methods *Methods for* ifit *objects*

Description

Methods and functions usable fo extracting information from an object of class ifit

Usage

```
## S3 method for class 'ifit'
coef(object, ...)

## S3 method for class 'ifit'
vcov(object, type = c("parameters", "statistics"), ...)

## S3 method for class 'ifit'
confint(object, parm, level = 0.95, ...)

controlIFIT(object)

globalIFIT(object)

jacobianIFIT(object)

numsimIFIT(object)

diagIFIT(object, plot = TRUE)

estfunIFIT(object)
```

Arguments

object	an object returned by ifit
...	additional arguments, currently not used
type	should the (estimated) covariance matrix of the parameters or of summary statistics returned?
parm	a specification of which parameters are to be given confidence intervals.
level	the confidence level required.
plot	if TRUE the summary statistics are plotted and the object is returned as invisible; otherwise, the statistics are not plotted and the object is returned as visible.

Value

coef returns a numeric vector with the estimated parameters

vcov returns a numeric matrix containing either the estimates or summary statistics variance-covariance matrix.

confint returns a matrix (or vector) with columns giving lower and upper confidence limits for each parameter. The intervals assume the asymptotic normality of the summary statistics.

controlIFIT return a list containing the values of the constants Ninit, NFitlocal,... used to estimate the model.

globalIFIT return the vector of the estimated parameters after the global search

jacobianIFIT returns a numeric matrix containing the estimated jacobian of the summary statistcs mean.

numsimIFIT returns an integer vector of length 2 containing the number of simulations used during the global and local search phases.

diagIFIT returns, and optionally plot, a numeric vector containing the observed summary statistics standardized with their means and standard deviances estimated at the final parameters; and, as attributes, the Sargan-Hansen test statistic, its degrees of freedom and the corresponding p-value. The class of the returned object is ifit.diag for which suitable print and plot methods are available.

estfunIFIT returns a numeric vector containing the estimating function evaluated at the final parameters; as attributes, the function also returns the estimated standard errors of the estimating function, its Mahalanobis norm and the number of degree of freedom. The class of the returned object is ifit.estfun for which a suitable print method is available.

---

toad-data *Toad data*

Description

A dataset describing the nocturnal movements of 66 Fowler's toad that were radio-tracked for 63 days.

Usage

```
toads
```

Format

A numerical matrix with 63 rows and 66 columns

Source

https://github.com/pmarchand1/fowlers-toad-move/

References

Philippe Marchand, Morgan Boenke, and David M. Green (2017) 'A Stochastic Movement Model Reproduces Patterns of Site Fidelity and Long-Distance Dispersal in a Population of Fowler's Toads (Anaxyrus Fowleri)', Ecological Modelling, 360, pp 63–69, doi:10.1016/j.ecolmodel.2017.06.025.

Examples

```
# White cells are missing data
lattice::levelplot(sweep(toads, 2, toads[1,]),
                   ylab="Toad", xlab="Day",
                   at=c(-Inf, seq(-200, 200, by=10), +Inf))
```

---

toad-model *Marchand et al.'s toad movement model*

Description

Function `toadSim` (written in C++) simulates a realization from the "model 1" suggested by Marchand et al. (2017); function `toadStat` computes a possible summary statistics.

Usage

```
toadSim(theta, data)

toadStat(
  data,
  lags = c(1, 2, 4, 8),
  q = c(0.01, seq(0.05, 0.95, by = 0.05), 0.99),
  dret = 10
)
```

Arguments

`theta`	a vector of length 3 containing the model parameters
`data`	a (number of days)x(number of toads) numerical matrix containing the locations of a set of toads. The simulated dataset will replicates the size, the NA pattern and the initial positions (first row) of this argument.
`lags`	an integer vector giving the lags used to compute the statistic
`q`	a numeric vector specifying the desired quantiles
`dret`	the distance used to classify the displacements as "returns" or "non-returns".

Details

The model describes the nocturnal movements (along a single spatial dimension) of Fowler's toads. It assumes that toads leave their refuges at night to forage and hide within sand dunes during the day. After the t-th nocturnal foraging phase, a toad is located at a displacement of Δ_t meters from its previous refuge site. The displacements Δ_t are assumed to be independent and identically distributed realizations of a symmetric, zero-centered α-stable random variable with stability parameter α and scale parameter γ. The toad then either returns to one of the previously visited sites (with probability π) or remains at its current location (with probability $1 - \pi$). In the former case, the refuge site is selected at random. The model parameter vector is $\theta = (\alpha, \gamma, \pi)'$.

Value

`toadSim` return a numerical matrix of the same size of `data`

`toadStat` returns a numeric vector of length `length(lags)x(1+length(q))`. The summary statistic is computed from absolute displacements at time lags classified as "returns" or "non-returns" depending on whether they are smaller than `dret`. For each lag in `lags`, the statistic comprises the "return" frequency and the median and adjacent quantile differences of the logarithms of the "non-return" distances (the quantiles are those specified in `q`).

References

Philippe Marchand, Morgan Boenke, and David M. Green (2017) 'A Stochastic Movement Model Reproduces Patterns of Site Fidelity and Long-Distance Dispersal in a Population of Fowler's Toads (Anaxyrus Fowleri)', Ecological Modelling, 360, pp 63–69, doi:10.1016/j.ecolmodel.2017.06.025.

Examples

```
# It takes some time to run
set.seed(20251025L)
tobs <- toadStat(toads)
tsim <- function(theta) toadStat(toadSim(theta, toads))
m <- ifit(tobs, tsim, l = c(0.01, 0, 0), u = c(2, 100, 1), trace = 1000)
m
confint(m)
numsimIFIT(m)
estfunIFIT(m)
diagIFIT(m)
```

`trait-model` *Jabot's trait model*

Description

Function `traitSim` (written in C++) simulates a realization from the trait model suggested by Jabot (2010); function `traiStat` computes a possible summary statistics.

Usage

```
traitSim(theta, nspecies = 1000L, population = 500L, ngen = 5000L)

traitStat(n, q = c(0.01, seq(0.05, 0.95, by = 0.05), 0.99))
```

Arguments

theta	a vector of length 4 containing the model parameters,
nspecies	the number of different levels of the trait characteristic (default 1000)
population	number of individuals living in the community (default 500)
ngen	number of generations (death/birth cycles) after which the actual population is returned (default 5000)
n	the observed data
q	the quantiles used to summarize the trait distribution

Details

The model describes the distribution of a numeric trait in a population of size population. The trait can only assume the values $(i-1)/(nspecies-1)$ for $i = 1, ..., nspecies$. The local competitive ability of a species with trait u is proportional to

$$F(u) = 1 - \omega + \omega\phi(u; \mu, \sigma)$$

where ω, μ and σ are parameters, and $\phi(u; \mu, \sigma)$ denotes the probability density function of a normal random variable with mean μ and variance σ^2.

The traits of the initial population are randomly drawn with probability proportional to $F(u)$. Then, for ngen steps, one individual randomly chosen dies. It is replaced either by an immigrant (with probability γ) or by a descendant of another randomly choosen existing individual (with probability $1-\gamma$). In the first case (immigration), the trait of the new individual is drawn with probability proportional to $F(u)$. In the second case (reproduction), the the probability that the trait of the new individual is u is proportional to the abundance of u in the population times $F(u)$.

The vector of model parameters is $\theta = (\gamma, \mu, \sigma, \omega)'$. Note that the parametrization used in this package differs from the one originally suggested by Jabot (2010). Specifically, Jabot assumes that $\gamma = J/(J + population - 1)$ and $F(u) = 1 + 2A\pi\sigma\phi(u; \mu, \sigma)$ where J and A are alternative parameters used in place of γ and ω, respectively.

Value

traitSim returns a integer vector of length nspecies containing the trait distribution of population individuals after ngen generations

traitStat returns a numeric vector containing the species richness (i.e., the number of distinct traits in the population), the Gini index, and the quantiles of the trait characteristic corresponding to the input arguments q

References

Franck Jabot (2010) 'A Stochastic Dispersal-Limited Trait-Based Model of Community Dynamics', Journal of Theoretical Biology, 262, pp. 650–61, doi:10.1016/j.jtbi.2009.11.004.

Examples

```
set.seed(1)
theta <- c(0.2, 0.7, 0.2, 0.7)
y <- traitSim(theta)
plot(seq(0, 1, length = length(y)), y, type = "h", xlab = "trait", ylab = "frequency")
print(tobs <- traitStat(y))

# It takes some time to run
tsim <- function(theta) traitStat(traitSim(theta))
m <- ifit(tobs, tsim, l = rep(0, 4), u = rep(1, 4), trace = 1000)
m
confint(m)
diagIFIT(m)
numsimIFIT(m)
```

Index