Package ‘resde’

May 19, 2023

Title Estimation in Reducible Stochastic Differential Equations

Version 1.1

Description Maximum likelihood estimation for univariate reducible stochastic differential equation models. Discrete, possibly noisy observations, not necessarily evenly spaced in time. Can fit multiple individuals/units with global and local parameters, by fixed-effects or mixed-effects methods. Ref.: Garcia, O. (2019) ``Estimating reducible stochastic differential equations by conversion to a least-squares problem", Computational Statistics 34(1): 23-46, <doi:10.1007/s00180-018-0837-4>.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.2.3

Imports stats, Deriv, nlme, methods

Suggests knitr

VignetteBuilder knitr

URL `https://github.com/ogarciav/resde/`

BugReports `https://github.com/ogarciav/resde/issues`

NeedsCompilation no

Author Oscar Garcia [aut, cre] (<`https://orcid.org/0000-0002-8995-1341`>)

Maintainer Oscar Garcia <`garcia@dasometrics.net`>

Repository CRAN

Date/Publication 2023-05-19 17:20:09 UTC

R topics documented:

resde-package **resde** - *Parameter estimation in reducible SDE models.*

Description

The main functions for model fitting are `sdemodel()` and `sdefit()`. First, specify the model structure in `sdemodel()`, including the variable transformation, any re-parameterizations, initial condition, and the presence or not of process, measurement, and initial condition noise. Then, fit the model with `sdefit()`, indicating the data to be used and starting parameter values for the iterations. For hierarchical models, one must also indicate which are the global and local parameters, and if fixed locals or a mixed effects method should be used.

Some auxilliary functions include the Box-Cox transformation `bc()`, and the *unified transformation* `unitran()`.

For detailed usage see the vignette: `vignette("resde-vignette", package="resde")`.

Author(s)

Maintainer: Oscar Garcia <`garcia@dasometrics.net`> (ORCID)

References

Garcia, O. (2019) "Estimating reducible stochastic differential equations by conversion to a least-squares problem". *Computational Statistics 34*(1), 23-46. doi:10.1007/s0018001808374

See Also

Useful links:

- `https://github.com/ogarciav/resde/`
- Report bugs at `https://github.com/ogarciav/resde/issues`

Examples

```
# Richards model  dH^c = b(a^c - H^c) dt + s dW for tree heights
tree1 <- subset(Loblolly, Seed == Seed[1]) # first tree
m <- sdemodel(~x^c, beta0=~b*a^c, beta1=~-b, mum=0) # no measurement error
sdefit(m, x="height", t="age", data=tree1, start=c(a=70, b=0.1, c=0.5))
```

bc *Box-Cox transformation*

Description

These functions calculate the Box-Cox transformation, its inverse, and derivative.

Usage

```
bc(x, lambda)

bc_inv(y, lambda)

bc_prime(y, lambda)
```

Arguments

`x, y`	Numeric vector (`x` must be >= 0).
`lambda`	Numeric scalar, power parameter.

Details

`bc()` uses `expm1()`, wich is more accurate for small `lambda` than a more "obvious" alternative like

```
  if (abs(lambda) < 6e-9) log(y)
 else (y^lambda - 1) / lambda
```

The difference might be important in optimization applications. See example below. Similarly, `bc_inv()` uses `log1p()`.

Value

`bc()`: Returns the transform value(s).

`bc_inv()`: Computes the inverse of `bc()`.

`bc_prime()`: Gives the derivative of `bc()` with respect to `y`.

Functions

- `bc()`: The Box-Cox transformation
- `bc_inv()`: Inverse of the Box-Cox transformation
- `bc_prime()`: Derivative of the Box-Cox transformation

Examples

```
bc(0.5, 1.5)
bc(1, 0)
obvious <- function(lambda){(0.6^lambda - 1) / lambda} # at y = 0.6
plot(obvious, xlab="lambda", xlim=c(1e-6, 1e-9), log="x")

bc_inv(-0.4, 1.5)
bc_inv(0, 0)

bc_prime(0.5, 1.5)
bc_prime(1, 0)
```

`sdefit` *Fit SDE model*

Description

ML estimation of parameters for a reducible SDE

Usage

```
sdefit(model, x, t, unit=NULL, data=NULL, start=NULL,
  global=NULL, local=NULL, known=NULL, method="nls",
  control=NULL, phi=NULL, phiprime=NULL)
```

Arguments

`model` Model specification, as produced by `sdemodel()`.

`x, t` Vectors with variables, or names of columns in data frame.

`unit` If applicable, unit id vector, or name of its column in data frame.

`data` Data frame, if data not given directly in `x`, `t`, `unit`.

`start` Named vector or named list with starting parameter values for non-hierarchical models. They can also be given in global.

`global` Named vector or list of global parameters and their starting values for hierarchical models. Can also contain starting values for non-hierarchical models.

`local` Named vector or list of local parameters and their starting values for hierarchical models. The value can be a vector with values for each unit, or a single scalar that applies to all the units.

`known` Named vector or list with any parameters that should be fixed at given values.

`method` `'nls'` for non-hierarchical models (default). For hierarchical models it can be `'nls'`, for fixed locals, or `'nlme'` for mixed effects.

`control` Optional control list for `nls()` or `nlme()`.

`phi` Optional transformation function. If `NULL` (default), it is automatically generated.

`phiprime` Optional derivative function. If `NULL` (default), it is automatically generated.

Value

List with two components: a list `fit` containing the output from the optimizer (`nls` or `nlme`), and a list `more` containing sigma estimates, log-likelihood, AIC and BIC. Note that in `fit`, "residual sum-of-squares" corresponds to `uvector`, not to `x` or `y`. Same for `nls` and `nlme` methods like `fitted` or `residuals` applied to `fit`.

Examples

```
m <- sdemodel(phi=~x^c, beta0=~b*a^c, beta1=~-b)
mod1 <- sdefit(m, "height", "age", data=Loblolly[Loblolly$Seed=="301",],
               start=c(a=70, b=0.1, c=1))
mod2 <- sdefit(m, "height", "age", "Seed", Loblolly, global=c(b=0.1, c=0.5),
               local=c(a=72))
```

`sdemodel` *Model specification*

Description

Specify transformation and re-parametrizations for reducible SDE model.

Usage

```
sdemodel(phi=~x, phiprime=NULL, beta0=~beta0, beta1=~beta1,
   t0=0, x0=0, mu0=0, mup=1, mum=1)
```

Arguments

`phi` Transformation formula $y = \varphi(x, parameters)$.

`phiprime` Optional formula for derivative of `phi`.

`beta0, beta1` Optional formulas or constants, possibly giving a re-parameterization,.

`t0, x0` Formulas or constants for the initial condition.

`mu0` Formula or constant for the initial condition σ_0 multiplier.

`mup, mum` Formulas or constants for the process and measurement σ multipliers.

Value

List with model specification, to be used by `sdefit()`.

Examples

```
   richards <- sdemodel(phi=~x^c, beta0=~b*a^c, beta1=~-b, mum=0)
```

`sdemodel_display` *Display the model specification*

Description

Display the model specification

Usage

```
sdemodel_display(model)
```

Arguments

`model`	SDE model specification, as produced by sdemodel()

Value

Invisibly returns its argument

Examples

```
mod <- sdemodel(); sdemodel_display(mod)
```

`str2fun_theta` *String to function, with parameters in theta*

Description

Normally not called by the user directly, used by `sdefit()`. Converts an expression, in a character string, to a function.

Usage

```
str2fun_theta(s)
```

Arguments

`s`	String representation of a function of `x` and parameters

Value

Function of `x` and `theta`, `theta` being a named vector or list of parameters.

Examples

```
str2fun_theta("x^c / a")
```

`unitran` *Unified transformation*

Description

Calculates a variable transformation that produces various growth curve models, depending on the values of two shape parameters, `alpha` and `beta`. Models can also be specified by name. Uses `bc()`, `bc_inv()`, `bc_prime()`.

Usage

```
unitran(x, name=NULL, par=NULL, alpha=NULL, beta=NULL, reverse="auto")

unitran_inv(y, name=NULL, par=NULL, alpha=NULL, beta=NULL, reverse="auto")

unitran_prime(x, name=NULL, par=NULL, alpha=NULL, beta=NULL, reverse="auto")
```

Arguments

`x, y` Variable to be transformed, `x` must be between 0 and 1.

`name` Optional model name, case-insensitive, in quotes. One of `Richards`, `monomolecular`, `Mitscherlich`, `Bertalanffy`, `Gompertz`, `logistic`, `Levacovic`, `Weibull`, `Korf`, `exponential`, `Schumacher`, `Hosfeld`.

`par` Model parameter, if needed and model name supplied.

`alpha, beta` Shape parameters, if the model is not specified by name.

`reverse` Reverse `x` and `t` axes? One of `"yes"`, `"no"`, `"auto"`. With `"auto"`, axes are reversed as necessary for an upper asymptote. (i.e., if `alpha <= 0` and `beta > 0`).

Value

`unitran()`: Transformed `x`, i.e., $y = \varphi(x)$.

`unitran_inv()`: Inverse of `unitran()`, $x = \varphi^{-1}(y)$.

`unitran_prime()`: Derivative of `unitran()`, $y' = \varphi'(x)$.

Functions

- `unitran()`: Unified transformation.
- `unitran_inv()`: Inverse of unitran().
- `unitran_prime()`: Derivative of `unitran()` with respect to `x`.

Examples

```
curve(unitran(x, "Gompertz"))  # same as unitran(x, alpha=0, beta=0)
curve(unitran_inv(y, "logistic"), xname="y", from=-4, to=4)
curve(unitran_prime(x, "logistic"))
```

userphi *Examples of optional external transformation and derivative functions*

Description

Templates for user-supplied transformation and drivative functions, used by `sdefit()` if specified in parameters `phi` and/or `phiprime`. To be completed by the user.

Usage

```
userphi(x, theta)

userphiprime(x, theta)
```

Arguments

`x`	Numeric vector, variable to be transformed.
`theta`	Named list of transformation parameters

Value

Transformed variable

Transformation derivative

Functions

- `userphi()`: transformation
- `userphiprime()`: derivative

uvector *ML estimation vector for reducible SDEs*

Description

These functions are not normally called directly by the user. Function `uvector()` is used by `sdefit()`. Function `uvector_noh()` is a more limited version, maintained for documentation purposes. Function `logdet_and_v()` is used by `uvector()` and `uvector_noh()`.

Usage

```
uvector(x, t, unit = NULL, beta0, beta1, eta, eta0, x0, t0, lambda,
  mum = 1, mu0 = 1, mup = 1, sorted = FALSE, final = FALSE)

uvector_noh(x, t, beta0, beta1, eta, eta0, x0, t0, lambda, final = FALSE)

logdet.and.v(cdiag, csub = NULL, z)
```

Arguments

`x, t`	Data vectors
`unit`	Unit id vector, if any.
`beta0, beta1, eta, eta0, x0, t0`	SDE parameters or re-parameterizations.
`lambda`	Named list of parameters(s) for `phi()`, possibly local vectors.
`mum, mu0, mup`	Optional σ multipliers.
`sorted`	Data already ordered by increasing t?
`final`	Mode, see below.
`cdiag`	Vector with the diagonal elements c_{ii} of C.
`csub`	Vector with sub-diagonal $c_{i,i-1}$ for $i > 1$.
`z`	A numeric vector

Details

`uvector()` and `uvector_noh()` calculate a vector of residuals for sum of squares minimization by `nls()` or `nlme()`. The first one works both for single-unit and for bilevel hierarchical models. It is backward-compatible with `uvector_noh()`, which is only for single-unit models but simpler and easier to understand. They require a transformation function `phi(x, theta)`, and a function `phiprime(x, theta)` for the derivative dy/dx, where `theta` is a list containing the transformation parameters.

`logdet_and_v()` calculates $\log[\det(L)]$ and $v = L^{-1}z$, where $C = LL'$, with L lower-triangular.

The three functions are essentially unchanged from García (2019) <doi:10.1007/s001800180837-4>, except for a somewhat safer computation for very small `beta1`, and adding in `logdet_and_v()` a shortcut for when L is diagonal (e.g., when $\sigma_m = 0$). The transformation functions `phi` and `phiprime` can be passed as globals, as in the original, or in an environment named `trfuns`.

Value

`uvector()` and `uvector_noh()`: If `final = FALSE` (default), return a vector whose sum of squares should be minimized over the parameters to obtain maximum-likelihood estimates. If `final = TRUE`, passing the ML parameter estimates returns a list with the sigma estimates, the maximized log-likelihood, and AIC and BIC criteria..

`logdet_and_v()`: List with elements `logdet` and `v`.

Functions

- `uvector()`: Estimation vector, general
- `uvector_noh()`: Estimation vector, non-hierarchical
- `logdet.and.v()`: Logarithm of determinant, and v vector

Index