Package 'panelPomp'

March 29, 2023

Type Package

Title Inference for Panel Partially Observed Markov Processes

Version 1.1

Date 2023-03-28

Description Data analysis based on panel partially-observed Markov process (PanelPOMP) models. To implement such models, simulate them and fit them to panel data, 'panelPomp' extends some of the facilities provided for time series data by the 'pomp' package. Implemented methods include filtering (panel particle filtering) and maximum likelihood estimation (Panel Iterated Filtering) as proposed in Breto, Ionides and King (2020) ``Panel Data Analysis via Mechanistic Models'' <doi:10.1080/01621459.2019.1604367>.

License GPL-3

Depends R(>= 4.1.0), pomp(>= 4.5.2)

Imports methods

RoxygenNote 7.2.3

Encoding UTF-8

Collate 'package.R' 'aaa.R' 'contacts.R' 'generics.R' 'get_col_row.R' 'panel_loglik.R' 'panel_logmeanexp.R' 'panelPomp.R' 'panelPomp_methods.R' 'params.R' 'pfilter.R' 'pfilter_methods.R' 'mif2.R' 'mif2_methods.R' 'panelGompertz.R' 'panelGompertzLikelihood.R' 'panelRandomWalk.R' 'plot.R' 'simulate.R'

NeedsCompilation no

Author Carles Breto [aut, cre] (<https://orcid.org/0000-0003-4695-4902>), Edward L. Ionides [aut] (<https://orcid.org/0000-0002-4190-0174>), Aaron A. King [aut] (<https://orcid.org/0000-0001-6159-3207>)

Maintainer Carles Breto <carles.breto@uv.es>

Repository CRAN

Date/Publication 2023-03-29 08:00:02 UTC

R topics documented:

`panelPomp-package` *Inference for PanelPOMPs (Panel Partially Observed Markov Processes)*

Description

The **panelPomp** package provides facilities for inference on panel data using panel partially-observed Markov process (PANELPOMP) models. To do so, it relies on and extends a number of facilities that the **pomp** package provides for inference on time series data using partially-observed Markov process (POMP) models.

The **panelPomp** package extends to panel data some of the capabilities of the **pomp** package to fit nonlinear, non-Gaussian dynamic models. This is done accomodating both fixed and random effects. Currently, the focus is on likelihood-based approaches. In addition to these likelihood-based tools, **panelPomp** also provides a framework under which alternative statistical methods for PANELPOMP models can be developed (very much like **pomp** provides a platform upon which statistical inference methods for POMP models can be implemented).

Data analysis using panelPomp

The first step in using **panelPomp** is to encode one's model(s) and data in objects of class `panelPomp`. One does this via a call to the panelPomp constructor function.

panelPomp version 1.1 provides algorithms for

1. particle filtering of panel data (AKA sequential Monte Carlo or sequential importance sampling), as proposed in Bretó, Ionides and King (2020). This reference provides the fundamental theoretical support for the averaging of Monte Carlo replicates of panel unit likelihoods as implemented in **panelPomp**; see `pfilter`

2. the panel iterated filtering method of Bretó, Ionides and King (2020). This reference provides the fundamental theoretical support for the extensions of the iterated filtering ideas of Ionides et al. (2006, 2011, 2015) to panel data as implemented in **panelPomp**; see `mif2`

The package also provides various tools for handling and extracting information on models and data.

Extending the pomp platform for developing inference tools

panelPomp extends to panel data the general interface to the components of POMP models provided by **pomp**. In doing so, it contributes to the goal of the **pomp** project of facilitating the development of new algorithms in an environment where they can be tested and compared on a growing body of models and datasets.

Comments, bug reports, and requests

Contributions are welcome, as are suggestions for improvement, feature requests, and bug reports. Please submit these via the panelPomp issues page. We particularly welcome minimal working examples displaying uninformative, misleading or inacurate error messages. We also welcome suggestions for clarifying obscure passages in the documentation. Help requests are welcome, but please consider before sending requests whether they are regarding the use of **panelPomp** or that of **pomp**. For help with **pomp**, please visit **pomp**'s FAQ.

Documentation

Examples are provided via the `contacts()`, `panelGompertz()` and `panelRandomWalk()` functions.

License

panelPomp is provided under the GPL-3 License.

Author(s)

Carles Bretó

References

Bretó, C., Ionides, E. L. and King, A. A. (2020) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**(531), 1178–1188. doi:10.1080/01621459.2019.1604367

See Also

pomp package, panelPomp

as *Coercing* panelPomp *objects as* list, pompList *or* data.frame

Description

When coercing to a data.frame, it coerces a panelPomp into a data.frame, assuming units share common variable names.

When coercing to a list, it extracts the unit.objects slot of panelPomp objects and attaches associated parameters.

When coercing to a pompList, it extracts the unit.objects slot of panelPomp objects and attaches associated parameters, converting the resulting list to a pompList to help the assignment of pomp methods.

Value

An object of class matching that specified in the second argument (to=).

Author(s)

Carles Bretó

See Also

Other panelPomp methods: panelPomp_methods

contacts *Contacts model*

Description

A panel model for dynamic variation in sexual contacts, with data from Vittinghof et al (1999). The model was developed by Romero-Severson et al (2015) and discussed by Bretó et al (2020).

Usage

```
contacts(
  params = c(mu_X = 1.75, sigma_X = 2.67, mu_D = 3.81, sigma_D = 4.42, mu_R = 0.04,
    sigma_R = 0, alpha = 0.9)
)
```

Arguments

params	parameter vector.

Value

A panelPomp object.

Author(s)

Edward L. Ionides

References

Bretó, C., Ionides, E. L. and King, A. A. (2020) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**(531), 1178–1188. doi:10.1080/01621459.2019.1604367

Romero-Severson, E.O., Volz, E., Koopman, J.S., Leitner, T. and Ionides, E.L. (2015) Dynamic variation in sexual contact rates in a cohort of HIV-negative gay men. *American journal of epidemiology*, **182**(3), 255–262. doi:10.1093/aje/kwv044

Vitinghoff, E., Douglas, J., Judon, F., McKiman, D., MacQueen, K. and Buchinder, S.P. (1999) Per-contact risk of human immunodificiency virus tramnsmision between male sexual partners. *American journal of epidemiology*, **150**(3), 306–311. doi:10.1093/oxfordjournals.aje.a010003

See Also

Other panelPomp examples: `panelGompertz()`, `panelRandomWalk()`

Examples

```
contacts()
```

---

get_dim *Get single column or row without dropping names*

Description

Subset matrix dropping dimension but without dropping dimname (as done by `` `[` `` by default).

Usage

```
get_col(matrix, rows, col)

get_row(matrix, row, cols)
```

Arguments

matrix	matrix.
rows	numeric; rows to subset; like with `[`, this argument can be left empty to designate all rows.
col	integer; single column to subset.
row	integer; single row to subset.
cols	numeric; columns to subset; like with `[`, this argument can be left empty to designate all columns.

Value

A named vector object.

Author(s)

Carles Bretó

Examples

```
m <- matrix(NA,dimnames=list('r1','c1'))
m[1,1] # = NA; R removes both names
get_col(m,rows=1,col=1) # = c(r1=NA) keeps colname
get_row(m,row=1,cols=1) # = c(c1=NA) keeps rowname
```

---

mif2 *PIF: Panel iterated filtering*

Description

Tools for applying iterated filtering algorithms to panel data. The panel iterated filtering of Bretó et al. (2020) extends to panel models the improved iterated filtering algorithm (Ionides et al., 2015) for estimating parameters of a partially observed Markov process. Iterated filtering algorithms rely on extending a partially observed Markov process model of interest by introducing random perturbations to the model parameters. The space where the original parameters live is then explored at each iteration by running a particle filter. Convergence to a maximum likelihood estimate has been established for appropriately constructed procedures that iterate this search over the parameter space while diminishing the intensity of perturbations (Ionides et al. 2006, 2011, 2015).

Usage

```
## S4 method for signature 'panelPomp'
mif2(
  data,
  Nmif = 1,
  shared.start,
  specific.start,
```

```
  start,
  Np,
  rw.sd,
  cooling.type = c("hyperbolic", "geometric"),
  cooling.fraction.50,
  block = FALSE,
  verbose = getOption("verbose"),
  ...
)

## S4 method for signature 'mif2d.ppomp'
mif2(
  data,
  Nmif,
  shared.start,
  specific.start,
  start,
  Np,
  rw.sd,
  cooling.type,
  cooling.fraction.50,
  block,
  ...
)

## S4 method for signature 'mif2d.ppomp'
traces(object, pars, ...)
```

Arguments

data	An object of class panelPomp or inheriting class.
Nmif	The number of filtering iterations to perform.
shared.start	named numerical vector; the starting guess of the shared parameters.
specific.start	matrix with row parameter names and column unit names; the starting guess of the specific parameters.
start	A named numeric vector of parameters at which to start the IF2 procedure.
Np	the number of particles to use. This may be specified as a single positive integer, in which case the same number of particles will be used at each timestep. Alternatively, if one wishes the number of particles to vary across timesteps, one may specify Np either as a vector of positive integers of length length(time(object,t0=TRUE)) or as a function taking a positive integer argument. In the latter case, Np(k) must be a single positive integer, representing the number of particles to be used at the k-th timestep: Np(0) is the number of particles to use going from timezero(object) to time(object)[1], Np(1), from timezero(object) to time(object)[1], and so on, while when T=length(time(object)), Np(T) is the number of particles to sample at the end of the time-series.

`rw.sd` An unevaluated expression of the form `quote(rw.sd())` to be used for all panel units. If a `list` of such expressions of the same length as the `object` argument is provided, each list element will be used for the corresponding panel unit.

`cooling.type, cooling.fraction.50` specifications for the cooling schedule, i.e., the manner and rate with which the intensity of the parameter perturbations is reduced with successive filtering iterations. `cooling.type` specifies the nature of the cooling schedule. See below (under "Specifying the perturbations") for more detail.

`block` A logical variable determining whther to carry out block resampling of unit-specific parameters.

`verbose` logical; if `TRUE`, diagnostic messages will be printed to the console.

`...` ....

`object` an object resulting from the application of IF2 (i.e., of class `mif2d.ppomp`)

`pars` names of parameters

Value

`mif2()` returns an object of class `mif2d.ppomp`.

`traces()` returns a `matrix` with estimated parameter values at different iterations of the IF2 algorithm in the natural scale. The default is to return values for all parameters but a subset of parameters can be passed via the optional argument `pars`.

Author(s)

Carles Bretó

References

Bretó, C., Ionides, E. L. and King, A. A. (2020) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**(531), 1178–1188. doi:10.1080/01621459.2019.1604367

Ionides, E. L., Bretó, C. and King, A. A. (2006) Inference for nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, **103**(49), 18438–18443. doi:10.1073/pnas.0603181103

Ionides, E. L., Bhadra, A., Atchadé, Y. and King, A. A. (2011) Iterated filtering. *Annals of Statistics*, **39**(3), 1776–1802. doi:10.1214/11AOS886

Ionides, E. L., Nguyen, D., Atchadé, Y., Stoev, S. and King, A. A. (2015) Inference via iterated, perturbed Bayes maps. *Proceedings of the National Academy of Sciences*, **112**(3), 719–724. doi:10.1073/pnas.1410597112

King, A. A., Nguyen, D. and Ionides, E. L. (2016) Statistical inference for partially observed Markov processes via the package **pomp**. *Journal of Statistical Software* **69**(12), 1–43. DOI: 10.18637/jss.v069.i12. An updated version of this paper is available on the package website.

See Also

pomp's mif2 at mif2, panel_loglik

Other panelPomp workhorse functions: `panelPomp`, `panel_loglik`, `pfilter()`

Examples

```
## start with a panelPomp object
p <- panelRandomWalk()
## specify which parameters to estimate via rw_sd() and how fast to cool
mp <- mif2(p,Np=10,rw.sd=rw_sd(X.0=0.2),cooling.fraction.50=0.5,cooling.type="geometric")
mp
## the object resulting from an initial estimation can be used as a new starting point
mmp <- mif2(mp,Np=10,rw.sd=rw_sd(X.0=0.2),cooling.fraction.50=0.5,cooling.type="geometric")
mmp
## convergence can be partly diagnosed by checking estimates and likelihoods at different iterations
traces(mmp)
```

---

panelGompertz *Panel Gompertz model*

Description

Builds a collection of independent realizations from the Gompertz model.

Usage

```
panelGompertz(
  N = 100,
  U = 50,
  params = c(K = 1, r = 0.1, sigma = 0.1, tau = 0.1, X.0 = 1),
  seed = 12345678
)
```

Arguments

N	number of observations for each unit.
U	number of units.
params	parameter vector, assuming all units have the same parameters.
seed	passed to the random number generator for simulation.

Value

A panelPomp object.

Author(s)

Edward L. Ionides, Carles Bretó

References

Bretó, C., Ionides, E. L. and King, A. A. (2020) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**(531), 1178–1188. doi:10.1080/01621459.2019.1604367

King, A. A., Nguyen, D. and Ionides, E. L. (2016) Statistical inference for partially observed Markov processes via the package **pomp**. *Journal of Statistical Software* **69**(12), 1–43. DOI: 10.18637/jss.v069.i12. An updated version of this paper is available on the package website.

See Also

Other panelPomp examples: `contacts()`, `panelRandomWalk()`

Examples

```
panelGompertz()
```

`panelGompertzLikelihood` *Likelihood for a panel Gompertz model via a Kalman filter*

Description

Evaluates the likelihood function for a panel Gompertz model, using a format convenient for maximization by optim() to obtain a maximum likelihood estimate. Specifically, estimated and fixed parameters are supplied by two different arguments.

Usage

```
panelGompertzLikelihood(x, panelPompObject, params)
```

Arguments

`x`	named vector for a subset of parameters, corresponding to those being estimated.
`panelPompObject`	a panel Gompertz model.
`params`	named vector containing all the parameters of the panel Gompertz model. Estimated parameters are overwritten by x.

Value

A `numeric` value.

Author(s)

Edward L. Ionides

Examples

```
pg <- panelGompertz(N=2,U=2)
panelGompertzLikelihood(coef(pg),pg,coef(pg))
```

panelPomp *Constructing* panelPomp *objects*

Description

This function constructs panelPomp objects, representing PanelPOMP models (as defined in Bretó et al., 2020). PanelPOMP models involve multiple units, each of which can in turn be modeled by a POMP model. Such POMP models can be encoded as a list of pomp objects, a cornerstone that the panelPomp function can use to construct the corresponding panelPomp object.

Usage

```
panelPomp(object, shared, specific, params)
```

Arguments

object	required; either (i) a list of pomp objects; or (ii) an object of class panelPomp or inheriting class panelPomp. If object is a list of pomps, the list must be named. All these pomps must either have no parameters or have the same parameter names. (This is just a format requirement. pomp codes can ignore any parameter that is irrelevant to any given panel unit.) If object is a panelPomp object, the function allows modifying the shared and unit-specific configuration of object.
shared, specific	optional; these arguments depend on the type of object. If object is a list of pomps, shared must be a numeric vector specifying parameter values shared among panel units. specific must be a matrix with parameter values that are unit-specific with rows naming parameters and columns naming units (these names must match those of object). If no values are specified and object has parameter values, these are set to be all unit-specific. If object is a panelPomp object, these arguments can still be used as described above to modify the parameters of object. Alternatively, the parameter configuration of object can be modified providing only a character shared naming parameters of object that should be shared (with values for parameters not originally shared taken from the unit-specific parameters of the first panel unit of object). shared=NULL sets all parameters as unit-specific.
params	optional; a named numeric vector. In this case, the nature of parameters is determined via a naming convention: names ending in "[unit_name]" are assumed to denote unit-specific parameters; all other names specify shared parameters.

Value

A panelPomp object.

Author(s)

Carles Bretó

References

Bretó, C., Ionides, E. L. and King, A. A. (2020) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**(531), 1178–1188. doi:10.1080/01621459.2019.1604367

King, A. A., Nguyen, D. and Ionides, E. L. (2016) Statistical inference for partially observed Markov processes via the package **pomp**. *Journal of Statistical Software* **69**(12), 1–43. DOI: 10.18637/jss.v069.i12. An updated version of this paper is available on the package website.

See Also

pomp's constructor at pomp

Other panelPomp workhorse functions: `mif2()`, `panel_loglik`, `pfilter()`

Examples

```
## recreate the 'panelRandomWalk()' example
prw <- panelRandomWalk()
prw2 <- panelPomp(unitobjects(prw),params=coef(prw))
identical(prw,prw2) # TRUE
```

---

panelPomp_methods *Manipulating* panelPomp *objects*

Description

Tools for manipulating panelPomp objects.

Usage

```
## S4 method for signature 'panelPomp'
coef(object)

## S4 replacement method for signature 'panelPomp'
coef(object, ...) <- value

## S4 method for signature 'panelPomp'
length(x)

## S4 method for signature 'panelPomp'
names(x)

## S4 method for signature 'panelPomp'
pparams(object)
```

```
pParams(value)

## S4 method for signature 'panelPomp'
print(x, ...)

## S4 method for signature 'panelPomp'
show(object)

## S4 method for signature 'panelPomp'
unitobjects(object)

## S4 method for signature 'panelPomp'
window(x, start, end)

## S4 method for signature 'panelPomp'
x[i]

## S4 method for signature 'panelPomp'
x[[i]]
```

Arguments

object, x	An object of class panelPomp or inheriting class panelPomp.
...	....
value	value to be assigned.
start, end	position in original times(pomp) at which to start.
i	unit index (indices) or name (names).

Value

coef() returns a numeric vector.

length() returns an integer.

names() returns a character vector.

When given objects of class panelPomp, pparams() returns a named list with two elements: shared, which is a named numeric vector, and specific, which is a matrix with parameter names in its row names and panel unit names in its column names.

pParams() returns a list with the model parameters in list form.

When given objects of class panelPomp, unitobjects() returns a list of pomp objects.

window() returns a panelPomp object with adjusted times.

`[` returns a panelPomp object.

`[[` returns a pomp object.

Methods

coef Extracts coefficients of panelPomp objects.

coef<- Assign coefficients to panelPomp objects.

length Count the number of units in panelPomp objects.

names Get the unit names of panelPomp objects.

pparams Extracts coefficients from panelPomp objects in list form.

pParams Converts panel coefficients from vector form to list form.

window Subset panelPomp objects by changing start time and end time.

[] Take a subset of units.

[[]] Select the pomp object for a single unit.

Author(s)

Carles Bretó, Aaron A. King.

See Also

Other panelPomp methods: as()

Examples

```
## access and manipulate model parameters and other features
prw <- panelRandomWalk()
coef(prw)
# replace coefficients
coef(prw) <- c(sigmaX=2,coef(prw)[-1])
coef(prw)
length(prw)
names(prw)
# extract parameters in list form
pparams(prw)
# convert vector-form parameters to list-form parameters
pParams(coef(prw))
## summaries of objects
print(prw)
show(prw)
## access underlying pomp objects
unitobjects(prw)
## select windows of time
window(prw,start=2,end=4)
## subsetting panelPomp objects
prw[1] # panelPomp of 1 unit (first unit of prw)
prw[[2]] # pomp object corresponding to unit 2 of prw
```

panelRandomWalk *Panel random walk model*

Description

Builds a collection of independent realizations from a random walk model.

Usage

```
panelRandomWalk(
  N = 5,
  U = 2,
  params = c(sigmaY = 1, sigmaX = 1, X.0 = 1),
  seed = 3141592
)
```

Arguments

N	number of observations for each unit.
U	number of units.
params	parameter vector, assuming all units have the same parameters.
seed	passed to the random number generator for simulation.

Value

A panelPomp object.

Author(s)

Edward L. Ionides, Carles Bretó

See Also

Other panelPomp examples: contacts(), panelGompertz()

Examples

```
panelRandomWalk()
```

panel_loglik *Handling of loglikelihood replicates*

Description

Handling of loglikelihood replicates.

Usage

```
## S4 method for signature 'matrix'
logLik(object, repMargin, first = "aver", aver = "logmeanexp", se = FALSE)
```

Arguments

object	Matrix with the same number of replicated estimates for each panel unit loglikelihood.
repMargin	The margin of the matrix having the replicates (1 for rows, 2 for columns).
first	Wether to "aver"(age replicates) or "aggr"(egate units) before performing the other action.
aver	How to average: 'logmeanexp' to average on the likelihood scale before taking logs or 'mean' to average after taking logs (in which case, which action is performed first does not change the result).
se	logical; whether to give standard errors.

Details

When se = TRUE, the jackknife se's from pomp::logmeanexp are squared, summed and the squared root is taken.

Value

numeric vector with the average panel log likelihood and, when se = TRUE, the corresponding standard error.

Author(s)

Carles Bretó

See Also

Other panelPomp workhorse functions: mif2(), panelPomp, pfilter()

Examples

```
ulls <- matrix(c(1,1.1,10.1,10),nr=2)
# when combining log likelihood estimates, the order in which aggregation and
# averaging are done can make a difference: panel_logmeanexp() implements the best
logLik(ulls,repMargin=1,first="aver",aver="logmeanexp")
logLik(ulls,repMargin=1,first="aggr",aver="mean",se=TRUE)
```

`panel_logmeanexp` *Log-mean-exp for panels*

Description

`se = TRUE`, the jackknife se's from `logmeanexp` are squared, summed and the squared root is taken.

Usage

```
panel_logmeanexp(x, MARGIN, se = FALSE)
```

Arguments

`x`	Matrix with the same number of replicated estimates for each panel unit loglikelihood.
`MARGIN`	The dimension of the matrix that corresponds to a panel unit and over which averaging occurs (1 indicates rows, 2 indicates columns).
`se`	logical; whether to give standard errors.

Value

A `numeric` value with the average panel log likelihood or, when `se = TRUE`, a `numeric` vector adding the corresponding standard error.

Author(s)

Carles Bretó

See Also

panel_loglik

Examples

```
ulls <- matrix(c(1,1,10,10),nr=2)
panel_logmeanexp(ulls,MARGIN=2,se=TRUE)
```

params *Convert to and from a* panelPomp *object* pParams *slot format and a one-row* data.frame

Description

These facilitate keeping a record of evaluated log likelihoods.

Usage

```
toVectorPparams(pParams)

fromVectorPparams(vec_pars)

toMatrixPparams(listPparams)
```

Arguments

pParams	A list with the format of the pParams slot of panelPomp objects.
vec_pars	A one-row data.frame with format matching that of the output of toVectorPparams.
listPparams	PanelPomp parameters in list format

Value

`toVectorPparams()` returns an object of class `data.frame`.

`fromVectorPparams()` returns an object of class `list` with the model parameters in list form.

`toMatrixPparams()` returns an object of class `matrix` with the model parameters in matrix form.

Author(s)

Carles Bretó

Examples

```
prw <- panelRandomWalk()
toVectorPparams(pparams(prw))
fromVectorPparams(toVectorPparams(pparams(prw)))
toMatrixPparams(pparams(prw))
```

pfilter *Particle filtering for panel data*

Description

Tools for applying particle filtering algorithms to panel data.

Usage

```
## S4 method for signature 'panelPomp'
pfilter(
  data,
  shared,
  specific,
  params,
  Np,
  verbose = getOption("verbose"),
  ...
)

## S4 method for signature 'pfilterd.ppomp'
logLik(object, ...)

## S4 method for signature 'pfilterd.ppomp'
unitlogLik(object, ...)
```

Arguments

data	An object of class panelPomp or inheriting class panelPomp.
shared, specific	optional; these arguments depend on the type of object. If object is a list of pomps, shared must be a numeric vector specifying parameter values shared among panel units. specific must be a matrix with parameter values that are unit-specific with rows naming parameters and columns naming units (these names must match those of object). If no values are specified and object has parameter values, these are set to be all unit-specific. If object is a panelPomp object, these arguments can still be used as described above to modify the parameters of object. Alternatively, the parameter configuration of object can be modified providing only a character shared naming parameters of object that should be shared (with values for parameters not originally shared taken from the unit-specific parameters of the first panel unit of object). shared=NULL sets all parameters as unit-specific.
params	optional; a named numeric vector. In this case, the nature of parameters is determined via a naming convention: names ending in "[unit_name]" are assumed to denote unit-specific parameters; all other names specify shared parameters.

Np — the number of particles to use. This may be specified as a single positive integer, in which case the same number of particles will be used at each timestep. Alternatively, if one wishes the number of particles to vary across timesteps, one may specify Np either as a vector of positive integers of length

```
length(time(object,t0=TRUE))
```

or as a function taking a positive integer argument. In the latter case, Np(k) must be a single positive integer, representing the number of particles to be used at the k-th timestep: Np(0) is the number of particles to use going from timezero(object) to time(object)[1], Np(1), from timezero(object) to time(object)[1], and so on, while when T=length(time(object)), Np(T) is the number of particles to sample at the end of the time-series.

verbose — logical; if TRUE, diagnostic messages will be printed to the console.

... — additional arguments, passed to the pfilter method of **pomp**.

object — required; either (i) a list of pomp objects; or (ii) an object of class panelPomp or inheriting class panelPomp.

If object is a list of pomps, the list must be named. All these pomps must either have no parameters or have the same parameter names. (This is just a format requirement. pomp codes can ignore any parameter that is irrelevant to any given panel unit.)

If object is a panelPomp object, the function allows modifying the shared and unit-specific configuration of object.

Value

pfilter() returns an object of class pfilterd.ppomp that is also a panelPomp object (with the additional filtering details).

When applied to an object of class pfilterd.ppomp, logLik() returns a numeric value.

When given objects of class pfilterd.ppomp, unitloglik() returns a numeric vector.

Methods

logLik Extracts the estimated log likelihood for the entire panel.

unitlogLik Extracts the estimated log likelihood for each panel unit.

Author(s)

Carles Bretó

References

Arulampalam, M. S., Maskell, S., Gordon, N. and Clapp, T. (2002) A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Trans. Sig. Proc.*, **50**(2), 174–188. doi:10.1109/78.978374

Bretó, C., Ionides, E. L. and King, A. A. (2020) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**(531), 1178–1188. doi:10.1080/01621459.2019.1604367

See Also

pomp's pfilter at pfilter, panel_loglik

Other panelPomp workhorse functions: `mif2()`, `panelPomp`, `panel_loglik`

Examples

```
# filter, which generates log likelihoods
pfrw <- pfilter(panelRandomWalk(),Np=10)
class(pfrw) # "pfilterd.ppomp"
is(pfrw,"panelPomp") # TRUE
pfrw
# extract single log likelihood for the entire panel
logLik(pfrw)
# extract log likelihood for each panel unit
unitlogLik(pfrw)
```

`plot` *panelPomp plotting facilities*

Description

Diagnostic plots for each unit in a panelPomp

Usage

```
## S4 method for signature 'panelPomp_plottable'
plot(
  x,
  variables,
  panel = lines,
  nc = NULL,
  yax.flip = FALSE,
  mar = c(0, 5.1, 0, if (yax.flip) 5.1 else 2.1),
  oma = c(6, 0, 5, 0),
  axes = TRUE,
  ...
)
```

Arguments

`x`	the object to plot
`variables`	optional character; names of variables to be displayed
`panel`	function of prototype `panel(x, col, bg, pch, type, ...)` which gives the action to be carried out in each panel of the display.
`nc`	the number of columns to use. Defaults to 1 for up to 4 series, otherwise to 2.

yax.flip	logical; if TRUE, the y-axis (ticks and numbering) should flip from side 2 (left) to 4 (right) from series to series.
mar, oma	the par mar and oma settings. Modify with care!
axes	logical; indicates if x- and y- axes should be drawn
...	ignored or passed to low-level plotting functions

Value

No return value (the function returns NULL).

Author(s)

Edward L. Ionides

Examples

```
plot(panelRandomWalk())
```

---

simulate *Simulations of a panel of partially observed Markov process*

Description

simulate generates simulations of the state and measurement processes.

Usage

```
## S4 method for signature 'panelPomp'
simulate(object, nsim = 1, shared, specific)
```

Arguments

object	a panelPomp object.
nsim	The number of simulations to perform. Unlike the pomp simulate method, all simulations share the same parameters.
shared	Named vector of the shared parameters.
specific	Matrix of unit-specific parameters, with a column for each unit.

Value

A single panelPomp object (if nsim=1) or a list of panelPomp objects (if nsim>1).

Author(s)

Edward L. Ionides

Examples

```
simulate(panelRandomWalk())
```

Index