Package 'panelPomp'

January 12, 2023

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 0.16.0 provides algorithms for

1. particle filtering of panel data (AKA sequential Monte Carlo or sequential importance sampling), as proposed in Breto, Ionides and King (2018). 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 Breto, Ionides and King (2018). 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 MIT License.

Author(s)

Carles Breto

References

Breto, C., Ionides, E. L. and King, A. A. (2019) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**, 1178–1188.

See Also

pomp package, panelPomp

`as` *Coercing* `panelPomp` *objects as a* `list`

Description

Extracts the `unit.objects` slot of `panelPomp` objects and attaches the associated parameters.

Extracts the `unit.objects` slot of `panelPomp` objects and attaches the associated parameters, converting the resulting list to a pompList to help the assignment of pomp methods.

Coerces a `panelPomp` into a data frame, assuming units share common variable names.

See Also

Other panelPomp methods: `panelPomp_methods`

Other panelPomp methods: `panelPomp_methods`

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 Breto et al (2019).

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.

Author(s)

Edward L. Ionides

References

Breto, C., Ionides, E. L. and King, A. A. (2019) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**, 1178–1188.

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.

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.

---

get_dim *Get single column or row without dropping names*

Description

Subset matrix dropping dimension but without dropping dimname (which is R's 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	numeric; single column to subset.
row	numeric; single row to subset.
cols	numeric; columns to subset; like with '[', this argument can be left empty to designate all columns.

mif2 *PIF: Panel iterated filtering*

Description

Tools for applying iterated filtering algorithms to panel data. The panel iterated filtering of Breto et al. (2018) 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

traces returns the 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.

References

Breto, C., Ionides, E. L. and King, A. A. (2019) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**, 1178–1188.

Ionides, E. L., Breto, 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., Atchade, Y. and King, A. A. (2011) Iterated filtering. *Ann. Statist.*, **39, no. 3**, 1776–1802. DOI: 10.1214/11-AOS886

Ionides, E. L., Nguyen, D., Atchade, 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 R Package pomp. *Journal of Statistical Software*, **69(12)**, 1–43. DOI: 10.18637/jss.v069.i12

See Also

pomp's mif2 at mif2, panel_loglik

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

`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.

Author(s)

Edward L. Ionides, Carles Breto

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.

Author(s)

Edward L. Ionides

panelPomp *Constructing* panelPomp *objects*

Description

This function constructs panelPomp objects, representing PanelPOMP models (as defined in Breto et al., 2018). 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 `pomp`s, the list must be named. All these `pomp`s 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 `pomp`s, `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.

References

Breto, C., Ionides, E. L. and King, A. A. (2019) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**, 1178–1188.

King, A. A., Nguyen, D. and Ionides, E. L. (2016) Statistical Inference for Partially Observed Markov Processes via the R Package pomp. *Journal of Statistical Software*, **69(12)**, 1–43. DOI: 10.18637/jss.v069.i12

See Also

pomp's constructor at pomp

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

`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).

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.

[ ] Take a subset of units.

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

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

Author(s)

Carles Breto, Aaron A. King.

See Also

Other panelPomp methods: as()

---

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.

Author(s)

Edward L. Ionides, Carles Breto

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.

See Also

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

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.

See Also

panel_loglik

---

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

```
fromVectorPparams(vec_pars)

toMatrixPparams(listPparams)

toVectorPparams(pParams)
```

Arguments

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

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.

Methods

logLik Extracts the estimated log likelihood for the entire panel.

unitlogLik Extracts the estimated log likelihood for each panel unit.

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.

Breto, C., Ionides, E. L. and King, A. A. (2019) Panel Data Analysis via Mechanistic Models. *Journal of the American Statistical Association*, **115**, 1178–1188.

See Also

pomp's pfilter at pfilter, panel_loglik

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

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

pparams pParams *generic.*

Description

pParams generic function.

Usage

```
pparams(object, ...)
```

Arguments

object	object.
...	Additional arguments.

Details

This is a generic function: methods can be defined for it.

---

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 paramters.
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

unitlogLik — unitlogLik *generic.*

Description

unitlogLik generic.

Usage

```
unitlogLik(object, ...)
```

Arguments

object	object.
...	Additional arguments.

Details

This is a generic function: methods can be defined for it.

unitobjects — unitobjects *generic.*

Description

unitobjects generic.

Usage

```
unitobjects(object, ...)
```

Arguments

object	object.
...	Additional arguments.

Details

This is a generic function: methods can be defined for it.

wQuotes *Interpret shortcuts for* sQuote()*s and* dQuote()*s in character objects*

Description

Concatenate character objects and replace singles quotes with sQuote()s and asterisks with dQuote()s: sQuote("x") and dQuote("x") can be written as just "x" and *x*.

Usage

```
wQuotes(...)
```

Arguments

...	objects to be passed to strsplit.

Examples

```
wQuotes("in ''fn'': *object* is 'a' required argument")
paste0("in ",sQuote("fn"),": ",dQuote("object")," is 'a' required argument")
```

Index