Package 'panelPomp'

April 12, 2020

Type Package

Title Statistical Inference for PanelPOMPs (Panel Partially Observed Markov Processes)

Version 0.10.0.2

Date 2020-04-07

Author Carles Breto, Edward L. Ionides, Aaron A. King

Maintainer Carles Breto <cbreto@umich.edu>

Description Tools for working with PanelPOMP models, i.e., with
Partially Observed Markov Processes (AKA state-space models, stochastic
dynamical systems) involving multiple, independent units (or individuals) for
each of which (potentially multivariate) time series data is available. The
basic idea driving panelPomp is to apply to a collection of units (or individuals)
some of the pomp package facilities for implementing POMP models, simulating
them, and fitting them to time series data. Regarding fitting, only the
iterated filtering (mif2) algorithm has currently been extended to the
panelPomp longitudinal/panel setting. (Alpha release version.)

License MIT + file LICENSE

LazyData TRUE

Depends R(>= 3.5.0),
pomp(>= 2.7.0)

Imports methods

RoxygenNote 7.1.0

Encoding UTF-8

Collate 'package.R'
'aaa.R'
'example.R'
'generics.R'
'get_col_row.R'
'mcap.R'
'panel_loglik.R'
'panel_logmeanexp.R'
'panelPomp.R'
'panelPomp_methods.R'

'params.R'
'pfilter.R'
'pfilter_methods.R'
'mif2.R'
'mif2_methods.R'

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 0.10.0.2 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 `panelPompExample` function. To see a list of the examples included in **panelPomp**, use

```
panelPompExample()
```

A list of examples included **panelPomp** can be displayed via

```
panelPomp:::pompExample()
```

License

panelPomp is provided under the MIT License.

Author(s)

Carles Breto

References

Breto, C., Ionides, E. L. and King, A. A. (2018) Panel Data Analysis via Mechanistic Models. *arXiv:1801.05695*.

See Also

pomp package, panelPomp

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

Description

Extracts the `unit.objects` slot of `panelPomp` objects.

Coerces a `panelPomp` into a data frame.

See Also

Other panelPomp methods: `panelPomp_methods`

Other panelPomp methods: `panelPomp_methods`

`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.

mcap *Monte Carlo adjusted profile*

Description

Given a collection of points maximizing the likelihood over a range of fixed values of a focal parameter, this function constructs a profile likelihood confidence interval accommodating both Monte Carlo error in the profile and statistical uncertainty present in the likelihood function.

Usage

```
mcap(lp, parameter, confidence = 0.95, lambda = 0.75, Ngrid = 1000)
```

Arguments

lp	a vector of profile likelihood evaluations.
parameter	the corresponding values of the focal parameter.
confidence	the required level of the confidence interval.
lambda	the loess parameter used to smooth the profile.
Ngrid	the number of points to evaluate the smoothed profile.

Value

mcap returns a list including the smoothed profile, a quadratic approximation, and the constructed confidence interval.

Author(s)

Edward L. Ionides

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,
  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,
  ...
)

## 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 in filtering. 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 timestep, one may specify `Np` either as a vector of positive integers (of length `length(time(object))`) or as a function taking a positive integer argument. In

the latter case, `Np(n)` must be a single positive integer, representing the number of particles to be used at the `n`-th timestep: `Np(1)` is the number of particles to use going from `timezero(object)` to `time(object)[1]`, `Np(2)`, from `time(object)[1]` to `time(object)[2]`, and so on.

`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` 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.

`cooling.fraction.50`
cooling.fraction.50 (seems to cause an error if documentation inherited from 'pomp' package).

`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. (2018) Panel Data Analysis via Mechanistic Models. *arXiv:1801.05695*.

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()`

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 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.

References

Breto, C., Ionides, E. L. and King, A. A. (2018) Panel Data Analysis via Mechanistic Models. *arXiv:1801.05695*.

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()`

`panelPompExample` *Construct* **panelPomp** *examples*

Description

`panelPompExample` constructs `panelPomp` objects that come with the **panelPomp** package.

Usage

```
panelPompExample(example)
```

Arguments

`example`	a `character` object specifying one of the examples that come with the **panelPomp** package. These examples can be listed using `panelPompExamples()`.

Details

`panelPompExample` is related but different than its previous (and eliminated as of version 2) **pomp** counterpart `pompExample`. The '`examples`' directory in the installed package has some example files that can be listed using `panelPompExamples()`.

Value

By default, a `panelPomp` object. If `envir` is an environment, this `panelPomp` object is created in that environment and named `example`.

Author(s)

Carles Breto and Aaron A. King

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()`

`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)

toVectorPparams(pParams)
```

Arguments

vec_pars	A one-row data.frame with format matching that of the output of toVectorPparams.
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	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.

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 in filtering. 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 timestep, one may specify Np either as a vector of positive integers (of length length(time(object))) or as a function taking a positive integer argument. In the latter case, Np(n) must be a single positive integer, representing the number of particles to be used at the n-th timestep: Np(1) is the number of particles to use going from timezero(object) to time(object)[1], Np(2), from time(object)[1] to time(object)[2], and so on.
verbose	logical; if TRUE, diagnostic messages will be printed to the console.
...	additional arguments, passed to the pfilter method of **pomp**.
object	An object of class panelPomp or inheriting class panelPomp.

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. (2018) Panel Data Analysis via Mechanistic Models. *arXiv:1801.05695*.

See Also

pomp's pfilter at pfilter, panel_loglik

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

Index